WISHBONE II COMPATIBLE

An Introduction to the WISHBONE BRIDGE

Wade D. Peterson, Silicore Corporation

First Release: 28 MAR 2012

Application Note: WBAN005

Abstract

This report describes methods for bridging the WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores. The term *bridge* is broadly defined as a way to exchange data between two or more buses which may have similar or different electrical, mechanical or logical structures. Bridges are commonly used to link traditional microcomputer buses like USB, PCI and VMEbus, as well as most networks in use today. Within the context of the WISHBONE SoC, the uppercase term *BRIDGE* specifically refers to a functional module that transfers DATUM INTERLOCKED bus cycles between two clock domain or clock sub-domain regions. This report discusses three general types of BRIDGE: the *common-clock BRIDGE*, the *synchronized BRIDGE* and the *asynchronous BRIDGE*. Untested proof-of-concept prototypes are presented to flesh out potential design problems for an enhanced WISHBONE II architecture. Each is presented as a CLASS Machine design representation which adheres to the WISHBONE II timing parameters.

Copyright Notice

Document History

28 MAR 2012 First published under the title: An Introduction to the WISHBONE BRIDGE.
WBAN005 Electronic format: < WbBridge_28MAR2012.pdf >

Credits

Drawings and text in this document were created with application software from OpenOffice.org.

Legal Disclaimer; Notice of Patent, Copyright, Trademark & Trade Secrecy Rights

Silicore Corporation is not aware that the information contained herein, nor any product or service that uses this information, causes infringement on the patent, copyright, trademark or trade secret rights of others. However, there is a possibility that such infringement may exist without its knowledge. The user of this document is responsible for determining if products or services it produces in accordance with this document infringe on the intellectual property rights of others.

All U.S. and International patent rights to any invention(s) described in this document, which might be assigned to Silicore Corporation, are hereby abandoned and injected into the public domain under the authority of 35 U.S.C. 102(c).

Silicore™ is a common law trademark of Silicore Corporation (Plymouth, MN USA).

The 'CONGA' and 'CONGA Cycle' word marks are trade dress for the combining, enclosing, compilation and de-compilation of System-on-Chip and microcomputer bus cycles in any product, software product or service related to a semiconductor chip product or associated article(s) of manufacture thereof. The 'WISHBONE Off-chip Component Interconnection (WOCI)' word mark is trade dress for an off-chip variant of WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores. The 'WISHBONE II' word mark is trade dress for enhancements to the WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.2 (change tracking document dated 30 NOV 2001). Silicore Corporation is the originator of the CONGA, CONGA Cycle, WISHBONE Off-chip Component Interconnection (WOCI) and WISHBONE II word marks, and hereby injects all right of trademark for them into the public domain so they may be used freely, on a worldwide basis, without further obligation to Silicore Corporation.

This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Su*bd. 5(ii)

The author is not aware that the information contained herein is subject to any export restrictions by the Government of the United States of America or of any other country, including (but not limited to) those established under: (a) U.S. Government Export Administration Regulations (EAR) or (b) under U.S. Title 22 CFR relating to International Traffic in Arms Regulations (ITAR).

Contents

Chapter 1 - Introduction

This report describes methods for bridging the WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores. The term *bridge* is broadly defined as a way to exchange data between two or more buses which may have similar or different electrical, mechanical or logical structures. Bridges are commonly used to link traditional microcomputer buses like USB, PCI and VMEbus, as well as most networks in use today. Within the context of the WISHBONE SoC, the uppercase term *BRIDGE* specifically refers to a functional module that transfers DATUM INTERLOCKED bus cycles between two clock domain or clock sub-domain regions. This report discusses three general types of BRIDGE: the *common-clock BRIDGE*, the *synchronized BRIDGE* and the *asynchronous BRIDGE*.

Untested proof-of-concept prototypes are presented to flesh out potential design problems for an enhanced WISHBONE II architecture[1]. Each is presented as a CLASS Machine design representation[2] which adheres to the WISHBONE II timing parameters[3].

```
Tpd(AIS), clk-su (max):  Asynchronous interflop synchronizer transfer delay.
                         Used exclusively with the trans-domain double
                         D-type synchronizer.

Tpd(DOM), clk-su (max):  Clock domain transfer delay.

                         When two domains exist, then this parameter may
                         take the alternate forms of:

                            - Primary domain:   Tpd(PRI)
                            - Secondary domain: Tpd(SEC)

Tpd(STE), clk-su (max):  Synchronized trans-domain entry transfer delay.
Tpd(STX), clk-su (max):  Synchronized trans-domain exit transfer delay.
Tpd(TDP), in-out (max):  Synchronized trans-domain clock phase lag.

Note: All propagation delays which are presented in the form: 'Tpd(), clk-su
(max)' or 'Tpd(), in-out' must be evaluated over the following limits: 0 <
Tpd()(typ) ≤ Tpd()(max).  Note that the lower limit is always an
infinitesimally small value which is greater than zero (0).
```

WISHBONE II Timing Parameters

[1] This document claims the benefit of Peterson (2010) including (but not limited to): PLATEs: 101, 103, 116, 117, 200-211 and 300.

[2] ***CLASS Machine design representation***
Software that conforms to the CLASS Machine definition.
. . .
The CLASS Machine is described by Peterson (2008), p. 57-65

[3] ***timing parameter***
A relative unit of time that is enforced in a CLASS Machine design representation, or an absolute unit of time that is enforced when the design is synthesized, placed and routed onto a target device.

Interlocked Handshaking

A BRIDGE supports fully-interlocked handshaking between MASTER and SLAVE interfaces using the DATUM INTERLOCKED[4] bus cycles. Note that the other major type of WISHBONE bus cycle – the BLOCK INTERLOCKED cycle – cannot be bridged and interlocked on a per-cycle basis. Instead, those cycles are transported on a block-by-block basis using the REPEATER[5] functional module (which is not discussed further in this report).

The BRIDGE, and the DATUM INTERLOCKED cycles it supports, are both deterministic. This means they are predictable enough for use in real-time control systems. Part of this suitability is because they rely entirely on a fully-interlocked handshaking protocol.

A short study of the WISHBONE handshaking mode efficiency for BRIDGE applications is provided in an appendix at the end of this report.

Types of BRIDGE

A BRIDGE is generally described according to the type of boundary it crosses. There are two general types of boundary considered in this report:

- *Clock sub-domain boundary*. A boundary between two *clock sub-domain* regions. A clock sub-domain is one of the set of register-transfer logic (RTL) regions operating within a clock domain, at least one of which constrains the maximum clock frequency of the WISHBONE System-on-Chip (SoC).

- *Clock trans-domain boundary*. A boundary between two clock domain regions. The two clock domains operate at a different frequency, or at the same frequency but having a phase relationship which exceeds that of a low-skew clock distribution.

These boundaries are further classified according to the three different clocking relationships which

[4] ***DATUM INTERLOCKED***
● A control transport attribute for a WISHBONE bus cycle that indicates participation in 3-Edge(AF) or 3-Edge(RF) handshaking modes, is capable of performing data transfers during every system clock cycle [CLK], and where both MASTER and the SLAVE interfaces can throttle the data transfer rate on a per-cycle basis by inserting wait states. It supports bus cycles having READ, WRITE, DUPLEX and EVENT data transfer attributes using a single or multiphase bus protocol. ● A WISHBONE II CONGA compliant cycle that supersedes the SINGLE and BLOCK cycles which were previously defined under WISHBONE Rev. B.2 (2001+).

[5] ***REPEATER***
A WISHBONE functional module for regenerating BLOCK INTERLOCKED cycles across a clock domain or clock sub-domain boundary, and which uses an internal FIFO queue as a decoupling point. ■ *Ex.* A *REPEATER* may respond to DATUM INTERLOCKED cycles for the purpose of converting them to a BLOCK INTERLOCKED cycle.

. . .

In general, DATUM INTERLOCKED cycles are more suitable for control transport applications, and BLOCK INTERLOCKED cycles are more suitable for data transport applications. This difference arises because BLOCK INTERLOCKED cycles require a minimum of two clocks per cycle, and may require data to be assembled and disassembled before and after transport. At a higher system level (higher level of abstraction), these shortcomings are often overcome by using a message passing protocol, such as the one used in the Multibus II architecture (a microcomputer bus first standardized in 1987 as IEEE-1296).

may exist between the two sides of the BRIDGE. They are:

- *Common-clock BRIDGE*. Used exclusively at a clock sub-domain boundary where both sides of the BRIDGE share the same low-skew clock distribution. They are primarily used in two situations: (1) where RTL segmentation may increase the maximum clock domain frequency, or (2) where parallelism is created by blocking (splitting) an interconnection into two cycle domains.

- *Synchronized trans-domain BRIDGE*. Used in situations that cross a synchronized clock domain boundary. These are boundaries where the two clock domains are synchronized with each other, such as when a divide-by-two counter located in a primary (first) clock domain generates the clock for a secondary (second) clock domain. This classification is further refined according to the clock divisor ratio that exists between the primary and secondary domains:

    - Unity BRIDGE. Used when the primary clock domain drives the secondary clock domain with a driver (buffer); and where a primary-to-secondary frequency ratio of 1:1 exists.

    - Non-unity BRIDGE. Used when the primary clock domain operates at a higher frequency than the secondary clock domain; where the primary and secondary clocks are separated by a counter; and where a primary-to-secondary frequency ratio equal to or greater than 2:1 exists (2:1, 3:1, 4:1, etc.).

- *Asynchronous trans-domain BRIDGE*. Used in situations where two clock domains operate from two independent clock sources, such as when they are driven by separate crystal oscillators[6].

These classifications have been presented in order of increasing clock efficiency. That is, when a portable CLASS Machine design representation is used, the common-clock BRIDGE requires the fewest number of clock cycles, followed by the synchronized and the asynchronous types.

Each of these classifications can be further refined according to the four types of data transport they support. They are:

- *READ*. A data transport attribute for a WISHBONE bus cycle that transfers data to a MASTER interface from a selected SLAVE. It cannot be used during a BROADCAST cycle.

- *WRITE*. A data transport attribute for a WISHBONE bus cycle that transfers data from a MASTER interface to a selected SLAVE interface. In the case of a BROADCAST cycle, the data is transferred to two or more selected SLAVE interfaces.

- *DUPLEX*. A data transport attribute for a WISHBONE bus cycle that simultaneously transfers data in two directions between a MASTER interface and a selected SLAVE interface. This activity is prearranged by the system integrator, as no special control transport signaling exists for this bidirectional transfer. The interfaces perform bidirectional transfers only when the

[6] Unless they are justified under the rules of the specification, two clock domains which are synchronized by a phase-lock loop are considered to have an asynchronous timing relationship.

write-enable signal [WE] is asserted. Otherwise, when write-enable [WE] is negated, a standard READ cycle is performed. A MASTER interface may ignore any read data during a bidirectional transfer. It cannot be used during a BROADCAST cycle.

- *EVENT*. A data transport attribute for a WISHBONE bus cycle that carries no data except for a single bit of information passed from a MASTER interface to a selected SLAVE interface. The information transfer is implied from the occurrence of the bus cycle itself, without the use of separate data input or output signals. The write enable input signal [WE_I] is ignored by the SLAVE interface. In the case of a BROADCAST cycle, the data is transferred to two or more selected SLAVE interfaces.

Each of these classifications can be further refined according to the additional functions they provide. These include:

- *Unidirectional BRIDGE*. A BRIDGE with one pair of MASTER-SLAVE interfaces which propagate bus cycles in one direction. The symbol shown in Figure 1 represents a unidirectional BRIDGE in a system modeling diagram.

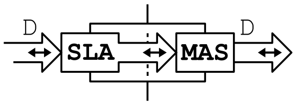


Figure 1. System modeling symbol for a unidirectional BRIDGE.

- *Bidirectional BRIDGE*. A BRIDGE with two pair of MASTER-SLAVE interfaces in opposition which propagate bus cycles in two directions. The SLAVE interfaces on a bidirectional BRIDGE also resolves a deadly embrace[7] (deadlock) condition by asserting the retry output signal [RTY_O] during simultaneous accesses from remote MASTERs on both sides. The symbols shown in Figure 2 represent bidirectional BRIDGEs in a system modeling diagram.

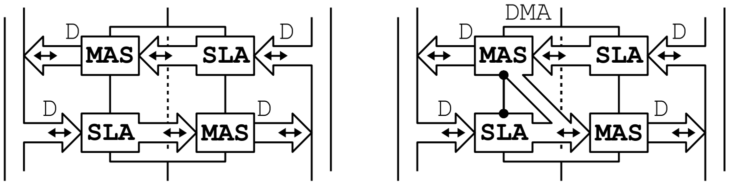


Figure 2. System modeling symbols for an automatic bidirectional BRIDGE (left); w/DMA (right).

- *Automatic BRIDGE*. A BRIDGE that responds automatically to bus cycles presented to its SLAVE interface.

- *DMA BRIDGE*. A BRIDGE that initiates bus cycles using an integral direct-memory access

[7] ***deadly embrace***
A form of system lock-up that occurs when both sides of a bidirectional BRIDGE attempt to access the other at the same time. This is prevented when a deadlock arbiter recognizes this situation and responds by allowing one side to complete its bus cycle, and terminating the other by asserting the retry output signal [RTY_O].

(DMA) unit.

Blocking Topologies

A blocking topology refers to a System-on-Chip (SoC) that uses WISHBONE BRIDGE or REPEATER functional modules to divide an interconnection into two cycle domains, both of which must be equipped with their own ARBITER and block address decoder (if needed). The major advantages to this might be:

- *Increased system clock frequency*. When a WISHBONE interconnection resides in its own register-transfer logic (RTL) segment, the technique divides the longest common combinatorial path[8] (LCCP) of the interconnection into two smaller parts. This increases the maximum clock frequency through the use of *segment balancing*[9].

- *Facilitate parallelism*. In some cases MASTER and SLAVE interfaces related by some function can be grouped into separate cycle domains. This increases the speed of the system by exploiting parallelism in the design.

- *Gateway between on-chip and off-chip interconnections*. Separates an interconnection into on-chip and off-chip blocks, such as a BRIDGE between a SoC and the WISHBONE Off-chip Component Interconnection (WOCI)[10].

The term *blocking* was derived from the discipline of operations management, where it is used in cases like the British block system of railway signaling[11]. Figure 3 shows the effects of blocking a SoC with four (4) MASTER and eight (8) SLAVE interfaces. This would have the effect of dividing the longest

[8] ***longest common combinatorial path (LCCP)***
One or more paths within the set of common combinatorial paths which has the longest time delay associated with it.

[9] ***segment balancing***
● The process of shifting combinatorial logic forward or backward in a pipeline so that each RTL segment takes the same amount of time to process data. It attempts to maximize the efficiency of a sequential process having multiple steps. A balanced pipeline will operate more efficiently than an unbalanced pipeline in terms of the rate of data processing, over a wide range of operating frequencies. The pipeline is said to be balanced when every segment executes in the same amount of relative time. When more than one pipeline exists in a clock domain, the process can be extended to all pipes operating within it. ● In a portable CLASS Machine design representation: segment balancing is performed by assuming a generic (non-specific) target device, and performing the following process upon the front end design for a unified value-added data process performed by a plurality of steps: (a) determine the number of look-up tables (LUTs) entries required by the combinatorial logic (along the longest common combinatorial path (LCCP) for each segment in the pipeline); (b) assign a representative (RMS) length for the interconnecting paths between each look-up table (LUT) in each segment; (c) assign time values to LUTs and interconnection paths based upon the FASM relative timing (ILUT) model; and (d) equalizing the propagation delay for all of the segments in the pipeline.

[10] ***WISHBONE Off-chip Component Interconnection (WOCI)***
A peripheral component interconnection that allows chips (components) to be wired together using standard WISHBONE SoC interface cores as the bus interface.

[11] In a block system of railway signaling, each block represents a different section of the same track. Thus, on a single east-west track a dispatcher might authorize an eastbound local freight in the first block to operate at the same time as a westbound local freight in the second block. Similarly, both local freights might be diverted to sidings to allow an express passenger service to pass through.

Watch a railway dispatcher's display in real-time at: 'BNSF Wayzata Morris' at: http://www.mnrails.com/atcs/bnsf-wayzata-morris.jpg combinatorial path in the interconnection. For example, if blocking reduced the size of the largest multiplexor from a four-input to a two-input type, then that part of the longest common combinatorial path (LCCP) would be reduced by about one-half (on many popular FPGA). Similarly, the two divided interconnections would use a less complex block address decoder, which would further reduce the LCCP.

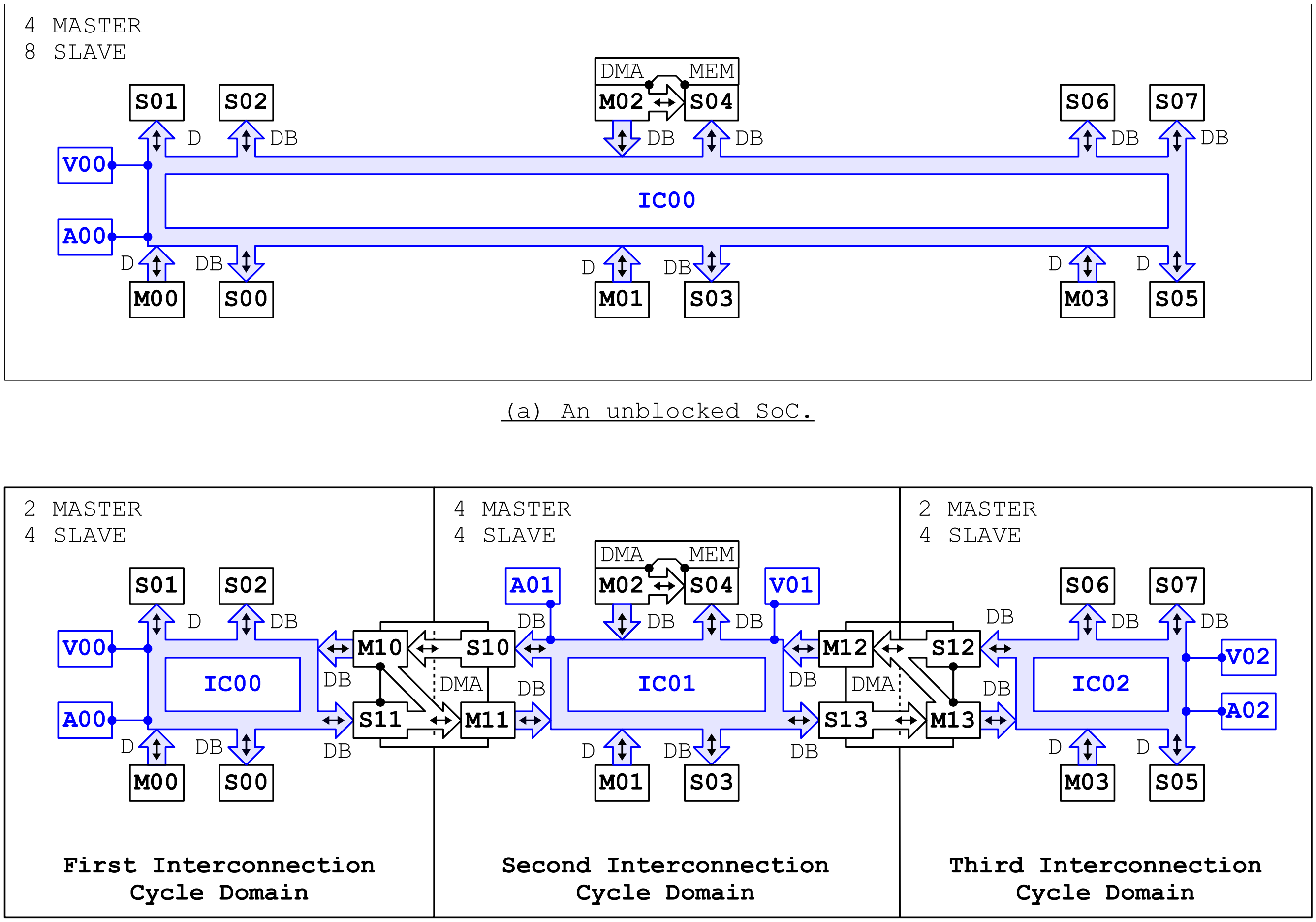
Furthermore, if blocking caused the MASTER and SLAVE interfaces to be grouped together by function, then MASTER/SLAVE pairs within all three blocks would be able to communicate with each other at the same time. This increases the parallelism of the system.

In most cases, such systems can be designed so that bus cycles outside of the block will generate automatic BRIDGE cycles into either of the other two blocks. In this mode a BRIDGE is expected to respond to an address block that covers all addresses outside of the block.

The blocked system also shows why a bidirectional BRIDGE must handle deadly embrace (deadlock) situations caused when MASTERs on both sides of the BRIDGE attempt to communicate with the other at the same time. When this happens, the BRIDGE SLAVE interface on one side would respond with the retry signal [RTY], and the SLAVE on the other side would respond with the acknowledge signal [ACK]. In WISHBONE, a MASTER must relinquish bus ownership if it receives the retry signal [RTY_I].

Another form of BRIDGE blocking is used to connect an on-chip system to an off-chip system, such as the WISHBONE Off-chip Component Interconnection (WOCI).

Self-accessible operation is sometimes required in a blocking topology. A bidirectional BRIDGE is not self-accessible, meaning that if a MASTER attempts to access its own interconnection by crossing the bidirectional BRIDGE from both sides, it would receive the retry terminal signal as a response. If self-accessible operation is needed (and supported by other aspects of the SoC), then use a pair of unidirectional BRIDGEs (which have no deadlock arbiter).



(a) An unblocked SoC.

(b) A three-way blocking topology using two common-clock BRIDGEs.

Figure 3. Effects of blocking a SoC with four (4) MASTER and eight (8) SLAVE interfaces.

Benchmarking

The various BRIDGE topologies discussed above may be compared in terms of their relative clock efficiencies[12]. This is one of the simplest and most influential ways to benchmark a CLASS Machine design representation because:

- Clock efficiency seeks to minimize the number of clock cycles required to perform a value-added data task.

- Clock efficiency is a portable benchmark, which presumes it can operate at clock speeds between D.C. and light.

[12] ***clock efficiency***
● A metric for evaluating a CLASS Machine design representation which equals the number of clock periods required to complete a process, or a ratio which compares two processes. It can be applied either to a complete a process or one iteration of a repetitive process. The metric is identified by the label 'Eff, clk', with units of 'Clk' (meaning a positive clock edge plus one clock period) or as a unitless quantity (e.g. percentage). ● The number of clock cycles needed to complete a process. Sometimes referred to as latency or clock latency.

- Clock efficiency is directly proportional to relative execution time.

- Clock efficiency is proportional to dynamic power.

- Clock efficiency influences the spectral content of EMI/RFI emissions.

Because a WISHBONE BRIDGE relies on a closed-loop control transport structure, the benchmarks are provided in terms of their closed-loop control transport clock efficiency.

Closed-loop Control Transport Clock Efficiency

Closed-loop control transport clock efficiency is a metric for benchmarking a control transport structure that measures the number of clock cycles required to complete a transaction. In the context of a BRIDGE, it measures the number of clock cycles required to complete a process that starts and ends at the same state machine. During all WISHBONE handshaking modes, a closed-loop control transport structure begins and ends at a MASTER interface. It starts with the assertion of the strobe output [STB_O], progresses through the interconnection to a selected SLAVE interface, which then generates a terminal signal that propagates back again to an input on the MASTER (e.g. [ACK_I]). Along the same path it coordinates the operation of data transport structures for READ, WRITE, DUPLEX and EVENT cycles.

In a BRIDGE application the closed-loop control transport structure includes the remote MASTER and remote SLAVE interfaces that are connected to the primary and secondary sides of the BRIDGE. Thus, it begins when the remote MASTER asserts the strobe output signal [STB_O] and ends when it receives a terminal signal (e.g. [ACK_I]).

It is assumed that the remote MASTER and SLAVE interfaces are connected to the BRIDGE using point-to-point interconnections. This approach evaluates the BRIDGE itself, rather than any intervening interconnection structures.

Test Methodology

Figure 4 shows how a bidirectional BRIDGE is arranged for the closed-loop control transport clock efficiency benchmark measurement. Note that the test excitation is provided by two identical bridge test state machines, both of which provide a representative CONGA cycle by generating two phases of a DATUM INTERLOCK cycle. Figure 5 shows the logic for these state machines. Only the control transport signals are exercised; the data transport signals do not affect the benchmark.

A two-phase DATUM INTERLOCKED CONGA test cycle allows two general types of measurement. The first is data transfer efficiency, which is the number of clock cycles needed to process one cycle phase (datum) through the BRIDGE. The second is the single cycle transfer efficiency, which is the number of clock cycles needed to process one complete cycle through the BRIDGE. The latter includes any overhead associated with the cycle signal [CYC] and the deadlock arbiter.

The benchmark for data transfer efficiency is measured in both directions, as well as for packed and unpacked interconnections on the destination side. These metrics are identical for both uni and bidirectional BRIDGEs. Note that the SLAVE interface on an automatic BRIDGE always introduces at least one wait-state, and so that end of the transaction always has a 3-Edge (RF) type waveform. However, the MASTER side of the BRIDGE might execute either 3-Edge (AF) or 3-Edge (RF) type cycles, depending upon whether the interconnection is packed or not. Since a point-to-point interconnection is used, the packing option is selected by the pack input signal [PAK_I] on the remote SLAVE interface. In actual operation a SLAVE may throttle the bus cycle by introducing any number of wait states, but the benchmark assumes that the remote SLAVE will always respond in the minimum number of cycles (i.e. in one or two clock periods).

The schematic diagram for the benchmark evaluation makes the following assumptions about the clock and reset sources for each type of BRIDGE topology which is tested:

- *Common-clock BRIDGE*. Both sides of the BRIDGE are connected to the same low-skew clock distribution network [CLKP].

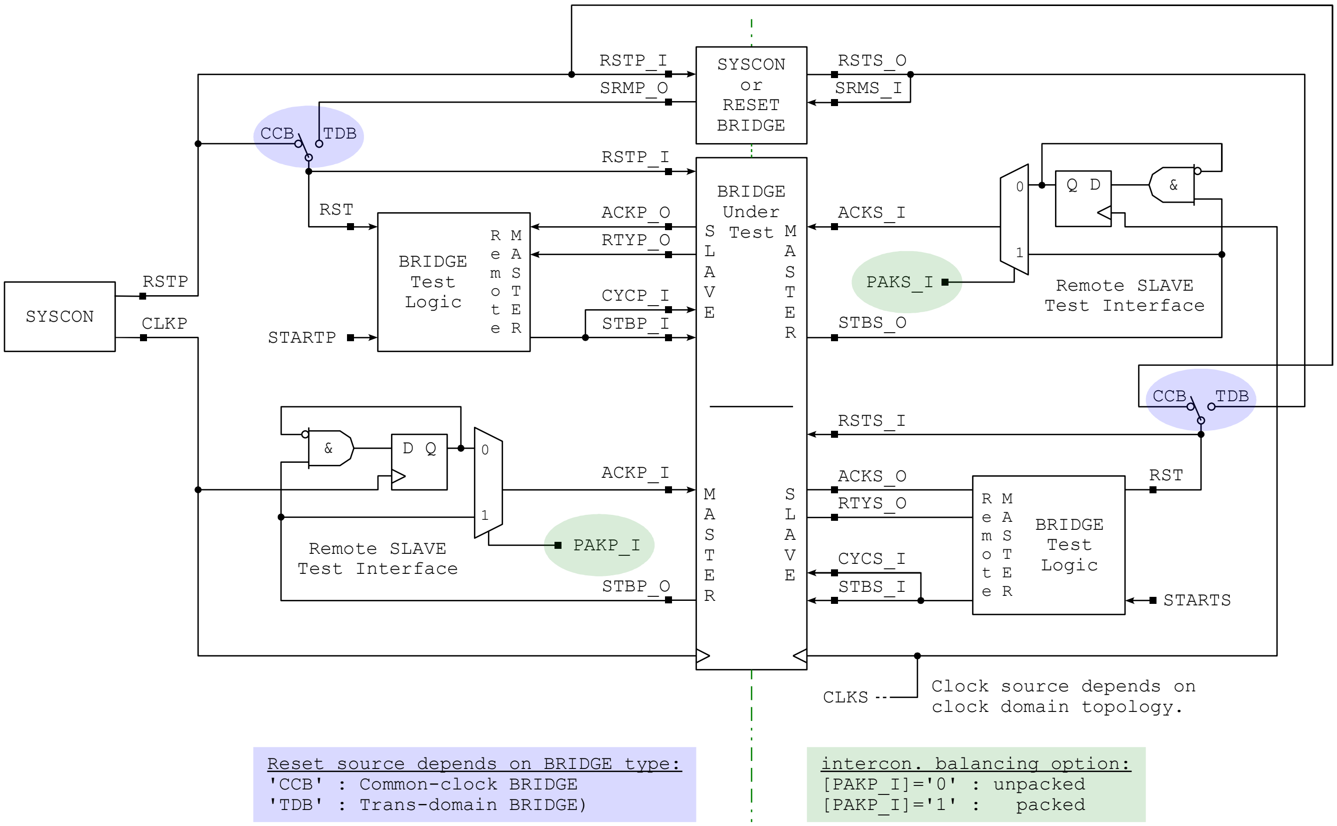
  Reset source switches are set to 'CCB', and both sides of the BRIDGE respond to the same system reset [RSTP].

- *Synchronized trans-domain BRIDGE*. The primary clock [CLKP] and secondary clock [CLKS] are connected by a synchronized non-unity trans-domain SYSCON. By definition, this also means that the primary clock [CLKP] is the source for the secondary clock [CLKS], and that a primary-to-secondary frequency divisor ratio of 2:1, 3:1, 4:1 or greater exists. In this case the metric should indicate the frequency divisor ratio.

  Reset source switches are set to 'TDB', and that the primary and secondary sides of the BRIDGE respond to secondary reset monitor signals [SRMP_O] and [SRMS_I] respectively.

- *Asynchronous trans-domain BRIDGE*. The primary clock [CLKP] and secondary clock [CLKS] are produced by independent frequency sources (e.g. two crystal oscillators). The metric should indicate the ratio between the primary and secondary clock frequencies.

  Reset source switches are set to 'TDB', and that the primary and secondary sides of the BRIDGE respond to secondary reset monitor signals [SRMP_O] and [SRMS_I] respectively.



Reset source depends on BRIDGE type:
'CCB' : Common-clock BRIDGE
'TDB' : Trans-domain BRIDGE)

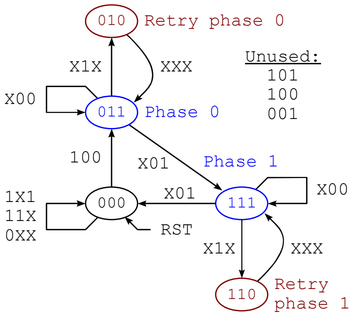
intercon. balancing option:
[PAKP_I]='0' : unpacked
[PAKP_I]='1' : packed

Acronyms

ACKP_I	WISHBONE acknowledge, primary, input
ACKP_O	WISHBONE acknowledge, primary, output
ACKS_I	WISHBONE acknowledge, secondary, input
ACKS_O	WISHBONE acknowledge, secondary, output
CCB	Common-clock BRIDGE
CLKP	WISHBONE system clock, primary, distr.
CLKP_I	WISHBONE system clock, primary, input
CLKS	WISHBONE system clock, secondary, distr.
CLKS_I	WISHBONE system clock, secondary, input
CYCP_I	WISHBONE cycle, primary, input
CYCS_I	WISHBONE cycle, secondary, intptut
PAKP_I	WISHBONE packed intercon., primary, input
PAKS_I	WISHBONE packed intercon., secondary, input
RST	Reset
RSTP	WISHBONE system reset, primary, distr.
RSTP_I	WISHBONE system reset, primary, input
RSTS	WISHBONE system reset, secondary, distr.
RSTS_I	WISHBONE system reset, secondary, input
RSTS_O	WISHBONE system reset, secondary, output
RTYP_O	WISHBONE retry, primary, output
RTYS_O	WISHBONE retry, secondary, output
STARTP	Start pulse, primary
STARTS	Start pulse, secondary
STBP_I	WISHBONE strobe, primary, input
STBP_O	WISHBONE strobe, primary, output
STBS_I	WISHBONE strobe, secondary, input
STBS_O	WISHBONE strobe, secondary, output
SRMS_I	WISHBONE sec. reset monitor, secondary, input
SRMP_O	WISHBONE sec. reset monitor, primary, output
TDB	Trans-domain BRIDGE

Figure 4. Arrangement for the BRIDGE benchmark evaluation.

State Diagram
Bridge Test Logic



```
STATES: PH1,PH0,EXC
INPUTS: START,RTY,ACK

ACK:   Acknowledge
EXI:   Excitation (CYC & STB)
PH1:   Phase 1 transfer
PH0:   Phase 0 transfer
RST:   System reset
RTY:   Retry
START: Start transfer
```

Truth Table
Bridge Test Logic

```
PPESRA PPE | PPESRA PPE
HHXTTC HHX | HHXTTC HHX
10CAYK 10C | 10CAYK 10C
   R       |    R
   T       |    T
------ --- | ------ ---
100000 XXX | 010000 011
100001 XXX | 010001 011
100011 XXX | 010011 011
100010 XXX | 010010 011

100110 XXX | 010110 011
100111 XXX | 010111 011
100101 XXX | 010101 011
100100 XXX | 010100 011

101100 XXX | 011100 011
101101 XXX | 011101 111
101111 XXX | 011111 010
101110 XXX | 011110 010

101010 XXX | 011010 010
101011 XXX | 011011 010
101001 XXX | 011001 111
101000 XXX | 011000 011

111000 111 | 001000 XXX
111001 000 | 001001 XXX
111011 110 | 001011 XXX
111010 110 | 001010 XXX

111110 110 | 001110 XXX
111111 110 | 001111 XXX
111101 000 | 001101 XXX
111100 111 | 001100 XXX

110100 111 | 000100 011
110101 111 | 000101 000
110111 111 | 000111 000
110110 111 | 000110 000

110010 111 | 000010 000
110011 111 | 000011 000
110001 111 | 000001 000
110000 111 | 000000 000
```

Karnaugh Map
Bridge Test Logic

NS: EXC

PH1,PH0,EXC \ START,RTY,ACK	100	101	111	110	010	011	001	000
100	X	X	X	X	X	X	X	X
101	X	X	X	X	X	X	X	X
111	1	0	0	0	0	0	0	1
110	1	1	1	1	1	1	1	1
010	1	1	1	1	1	1	1	1
011	1	1	0	0	0	0	1	1
001	X	X	X	X	X	X	X	X
000	1	0	0	0	0	0	0	0

NS: PH0

PH1,PH0,EXC \ START,RTY,ACK	100	101	111	110	010	011	001	000
100	X	X	X	X	X	X	X	X
101	X	X	X	X	X	X	X	X
111	1	0	1	1	1	1	0	1
110	1	1	1	1	1	1	1	1
010	1	1	1	1	1	1	1	1
011	1	1	1	1	1	1	1	1
001	X	X	X	X	X	X	X	X
000	1	0	0	0	0	0	0	0

NS: PH1

PH1,PH0,EXC \ START,RTY,ACK	100	101	111	110	010	011	001	000
100	X	X	X	X	X	X	X	X
101	X	X	X	X	X	X	X	X
111	1	0	1	1	1	1	0	1
110	1	1	1	1	1	1	1	1
010	0	0	0	0	0	0	0	0
011	0	1	0	0	0	0	1	0
001	X	X	X	X	X	X	X	X
000	0	0	0	0	0	0	0	0

```
Logic Equations (Canonical Form):                              LCCP (ILUT)
                                                               IN L1 L2 L3
EXC  :=         PH0 & /EXC                           & /RST        +2
      + /PH1 &  PH0                  & /RTY          & /RST
      +         PH0                  & /RTY & /ACK   & /RST
      +                        START & /RTY & /ACK   & /RST;

PH0  :=         PH0 & /EXC                           & /RST        +2
      + /PH1 &  PH0                                  & /RST
      +         PH0                  & /RTY & /ACK   & /RST
      +         PH0                  & /RTY          & /RST
      +                        START & /RTY & /ACK   & /RST;

PH1  :=  PH1 &  PH0 & /EXC                           & /RST        +2
      +  PH1                         & /RTY & /ACK   & /RST
      +  PH1 &  PH0                  & /RTY          & /RST
      + /PH1          &  EXC         & /RTY &  ACK   & /RST;
```

Figure 5. Internal logic for the BRIDGE benchmark evaluation.

Graphical Data Evaluation Technique

Benchmark data for each BRIDGE is evaluated using a graphical technique. This means that a scaled timing diagram is created for each timing circumstance, and that clock efficiencies are measured directly from the diagram. While somewhat cumbersome to implement, this technique allows precise evaluation of average, best case and worst case timing scenarios in complex trans-domain designs.

Figure 6 shows how the graphical measurement technique is applied at an asynchronous trans-domain boundary in an asynchronous BRIDGE. This situation is especially challenging because the signal passes from one clock domain to another. Thus, the clock efficiency of the signal will depend upon the instantaneous timing relationship between the primary and secondary clocks. This cannot be predicted exactly, and so must be done statistically.

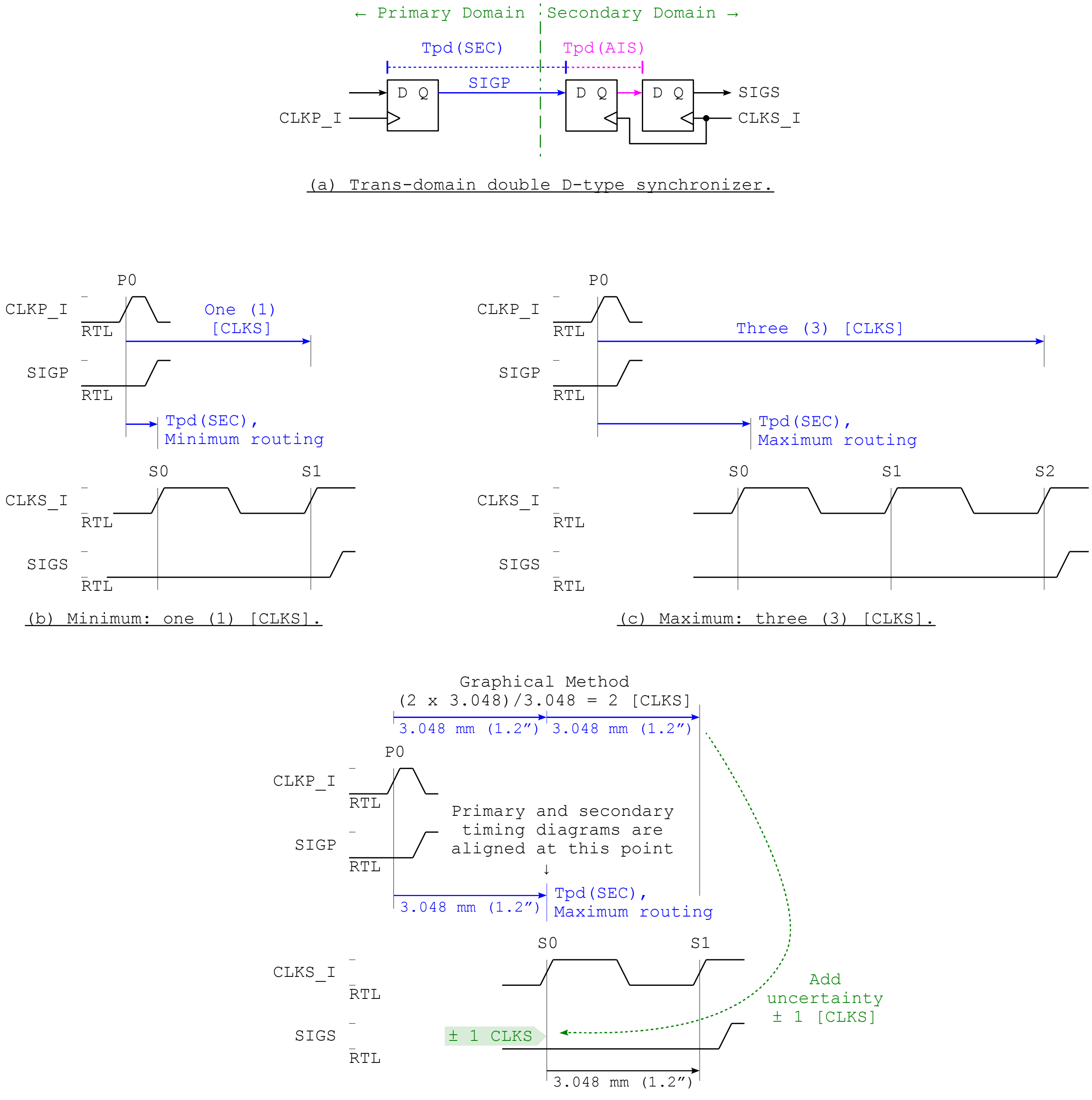
In the asynchronous BRIDGE, all signals of this type are processed through a trans-domain double D-type synchronizer like the one shown in Figure 6(a). This method is recommended for controlling synchronization, timing and metastability in a trans-domain signal that are tightly controlled (not all signals meet this criteria). It uses three flip-flops which are arranged in series along the signal path: an exit flip-flop located in a first clock domain and two synchronization flip-flops located in the second.

In this example the signal crosses a trans-domain boundary from a primary to a secondary clock domain. It is first registered by a flip-flop in the primary domain. This serves as the terminus to the clock sub-domain in the primary domain. The signal then crosses the trans-domain boundary and is registered by a first synchronization flip-flop. Timing in this segment is controlled by the trans-domain transfer delay. Because the set-up and hold times of the first synchronization flip-flop cannot be controlled, it is subject to metastability at its output. This is filtered out by the second flip-flop. Timing between the first and second synchronization flip-flops is always one destination clock period wide. The second synchronization flip-flop also serves as a terminus to the secondary clock sub-domain.

The clock efficiency benchmark for a signal processed through a trans-domain double D-type synchronizer is a complex statistical value described elsewhere in this report under the heading “Estimating Total Synchronization Time”.

The statistical range of values could be demonstrated for the entire signal synchronization in Figures 6(b) and 6(c). However, rather than drawing two sets of timing diagrams showing the minimum and maximum delay extremes, only one diagram is drawn for the average delay. This is modified by a green arrow that indicates a variability of plus/minus (±) one (1) destination clock periods. Thus, the timing diagram of Figure 6(d) replaces the other two.

The final diagram shows the graphical measurement technique used for benchmarking clock efficiency. In this case, the synchronization delay will be measured directly from the timing diagram between primary clock 'P0' and secondary clock 'S1'. The result is a total of '2 ± 1 CLKS'. This technique can be expanded and is especially useful for large timing diagrams having many signal trans-domain signal transactions.



(a) Trans-domain double D-type synchronizer.

(b) Minimum: one (1) [CLKS].

(c) Maximum: three (3) [CLKS].

(d) Metric measured graphically, and reported as: 2 ± 1 [CLKS].

Figure 6. Graphical measurement technique for an asynchronous trans-domain BRIDGE.

Chapter 2 - The Common-clock BRIDGE

The common-clock BRIDGE supports blocking within a clock domain region. This has the effect of increasing the SoC clock frequency, and in some cases can exploit parallelism in the design.

A WISHBONE interconnection uses the principles of register-transfer logic (RTL) to carry out its functions. All of the various RTL dependencies within the interconnection are combined into an entity called the clock sub-domain region.[13] A common-clock BRIDGE has the effect of dividing the interconnection – and the clock sub-domain region – into two smaller parts. If the interconnection happens to be the limiting factor in the maximum frequency of the SoC, then the BRIDGE will have the effect dividing its longest common combinatorial path (LCCP) into two smaller parts, and increasing the overall clock speed of the SoC.

In some situations where MASTER and SLAVE interfaces communicate over a shared bus interconnection, the interfaces can be grouped together in terms of how many bus cycles are shared between them. In this case the BRIDGE might exploit parallelism by separating these groups into their own cycle domains[14].

In other situations a BRIDGE might be used to separate an interconnection into on-chip and off-chip blocks, such as a BRIDGE between a chip-level system and the WISHBONE Off-chip Component Interconnection (WOCI).

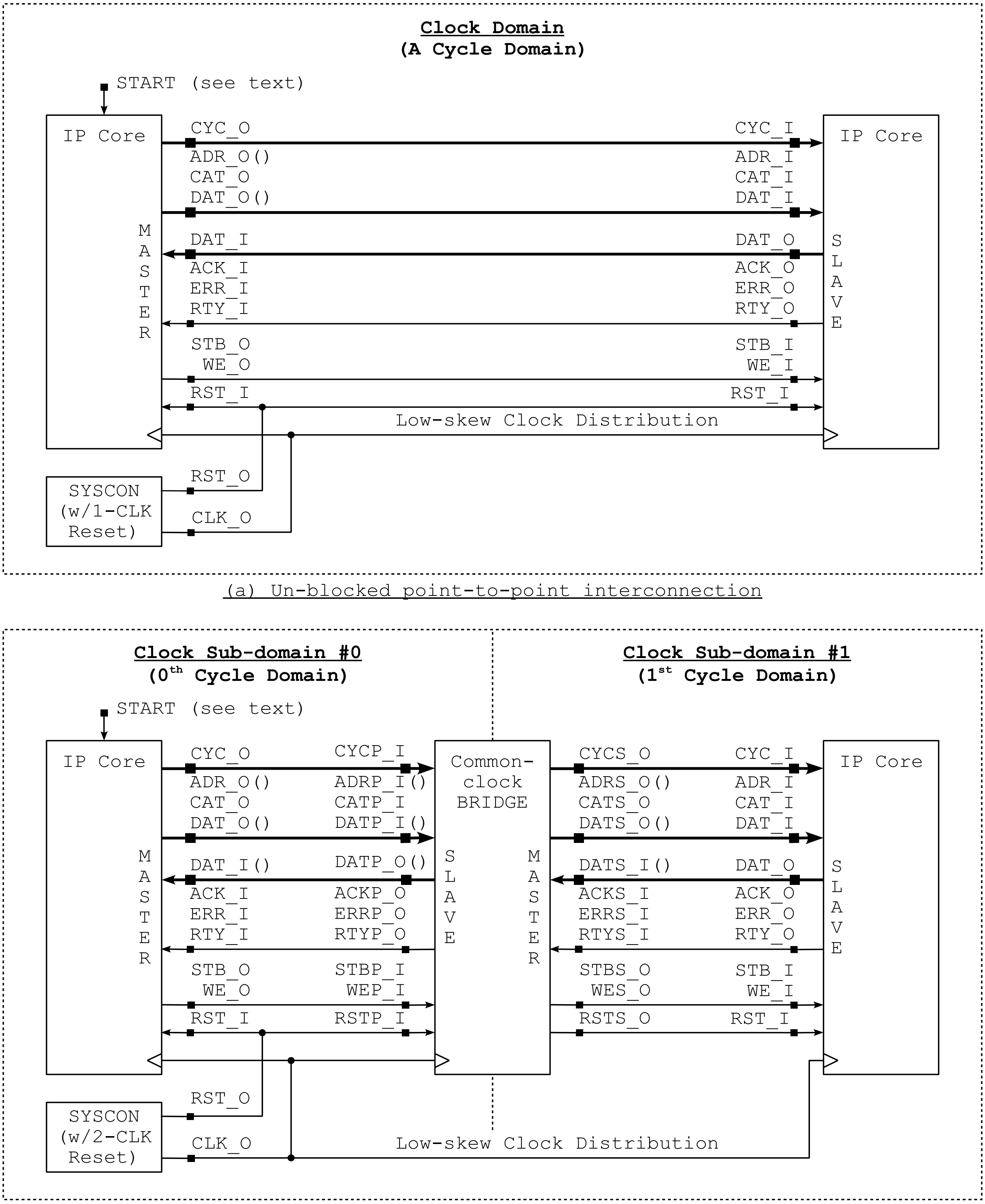
Figure 7 shows a WISHBONE point-to-point interconnection that is broken-up into two separate blocks (cycle domains) by an automatic, unidirectional, common-clock BRIDGE. This simple example is useful as a thought experiment to illustrate the operation and construction of a common-clock BRIDGE.

Reset Blocking

Reset blocking is the re-transmission of the WISHBONE system reset signal [RST] across a clock sub-domain boundary. Figure 8 shows how reset blocking is employed in the two-block system example above. Since the reset signal is distributed on a standard signal interconnection (rather than on a low-skew clock distribution), it must be blocked along with all of the other signals in the sub-domain.

[13] ***clock sub-domain***
One of the set of register-transfer logic (RTL) regions operating within a clock domain, at least one of which constrains the maximum clock frequency of the WISHBONE System-on-Chip (SoC).

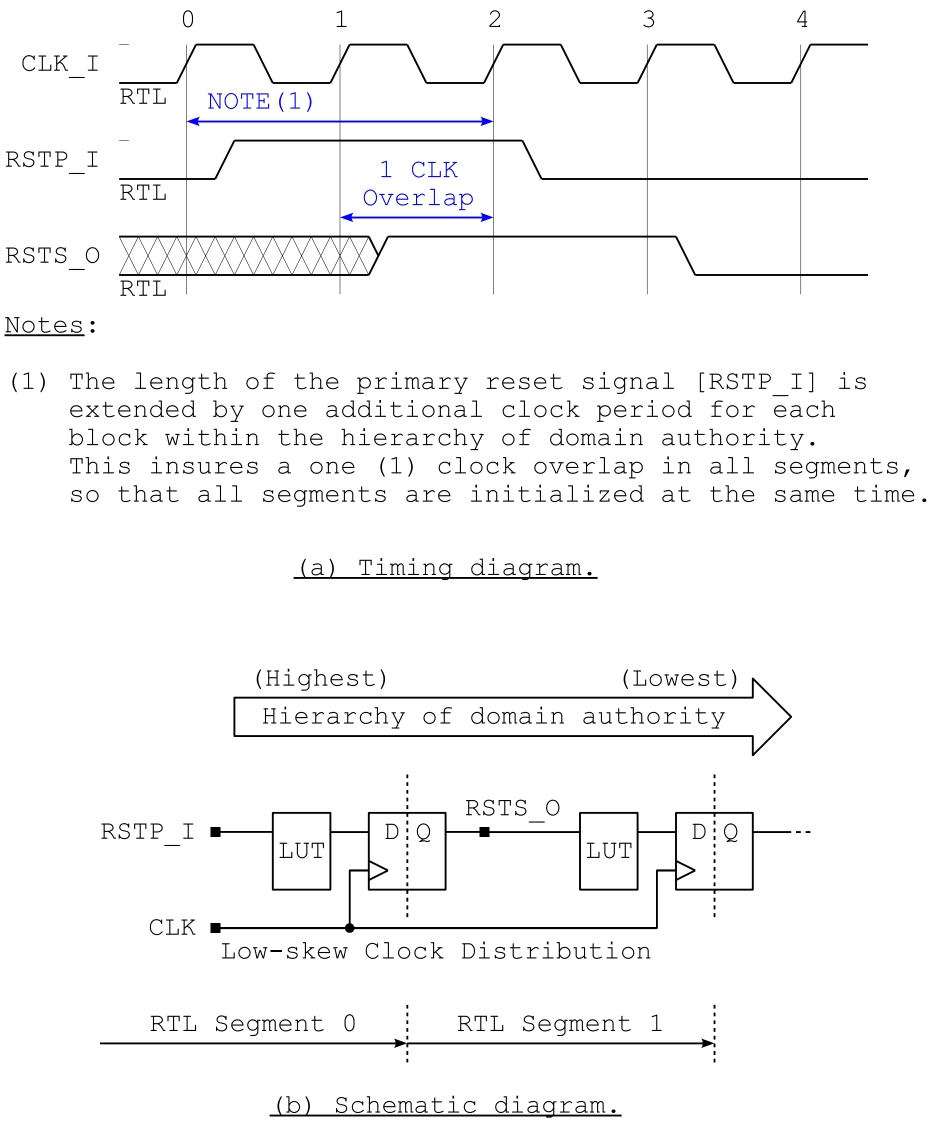
[14] ***cycle domain***
● A locus of similar bus cycle activity. ● In the WISHBONE System-on-Chip (SoC) a set MASTER and SLAVE interfaces which are attached to a common point-to-point, shared bus or crossbar switch interconnection. ■ *Ex.* Each side of a BRIDGE operates in its own *cycle domain* because each has a different timing relationship.



(a) Un-blocked point-to-point interconnection

(b) Blocked point-to-point interconnection

Figure 7. A WISHBONE point-to-point interconnection broken-up into two blocks.



Notes:

(1) The length of the primary reset signal [RSTP_I] is extended by one additional clock period for each block within the hierarchy of domain authority. This insures a one (1) clock overlap in all segments, so that all segments are initialized at the same time.

(a) Timing diagram.

(b) Schematic diagram.

Figure 8. How reset blocking is employed in the two-block example.

In a common-clock BRIDGE, reset blocking is very simple because it merely registers the reset signal with a flip-flop. This results in a hierarchy of domain authority[15], meaning that a signal in one clock domain or sub-domain region has authority over another.

To compensate for the one-clock delay between the primary and secondary sides of the BRIDGE, the length of the primary reset signal [RSTP] is extended by one additional clock period for each additional block within the hierarchy of domain authority. Thus, if the reset signal transcends two (2) blocks in

[15] ***hierarchy of domain authority***
A hierarchy with respect to the WISHBONE system clock [CLK] and reset [RST] signals that exists between two clock domain or sub-domain regions, where a primary domain or sub-domain has authority over a secondary domain or sub-domain. At a boundary between two asynchronous clock domain or two common-clock sub-domain regions, the authority is limited to the propagation of system reset. At a boundary between two synchronized clock domains the primary clock provides an excitation signal for the secondary clock, and so the authority encompasses both system reset and clock generation.

series, then the primary reset signal [RSTP] must be at least two clock periods long; if the reset signal transcends three (3) block in series it is at least three clock periods long and so forth. This insures a one-clock overlap so that all segments are initialized at the same time[16] at least once during the initialization period.

Bus Arbitration Factors

An automatic bidirectional BRIDGE has an internal deadlock arbiter to prevent system lock-ups which may be caused by a deadly embrace. During simultaneous accesses, one side will receive ownership of the destination bus, and the other side will receive the retry signal [RTY]. A remote MASTER, upon receiving the retry input signal [RTY_I], must terminate its bus cycle by negating its cycle output signal [CYC_O].

The deadlock arbiter only monitors bridge accesses, and does not attempt to influence a shared bus or crossbar switch ARBITER in any other way. The WISHBONE bus protocols guarantee that the deadlock arbiter and the interconnection ARBITER will not cause a system lock-up so long as they comply with the rules for the cycle signal [CYC] and retry signal [RTY]. In this respect the BRIDGE becomes an extension of the interconnection. For example, once a first MASTER in a first block is granted the bus in a second block, the remote interconnection is held by the first MASTER until it negates its cycle signal [CYC].

However, this does not necessarily guarantee that every MASTER in every block will obtain sufficient sufficient bandwidth across a BRIDGE. This will depend on factors associated with the SoC. Typical RRS, PRI and XBAR[17] ARBITERs are only intended to resolve bus ownership conflicts within their respective sphere of influence, but are not necessarily capable of resolving conflicts in other cycle domains.

[16] A similar technique is used to reduce the redundancy of reset inputs in multiple logic stages. Otherwise, the system reset signal would consume one (1) of every five (5) LUT inputs in designs where $LCCP_{REL}$ = +1 ILUT.

[17] ***RRS ARBITER***
● A WISHBONE functional module used with the standard or FIRECRACKER shared bus interconnections, which grants access to the INTERCON one MASTER at a time, on a rotating priority basis. It is one form of the FAIR ARBITER. ● 'RRS' is an acronym for 'Round-Robin Select'. ■ *Ex.* After an *RRS ARBITER* grants the interconnection to the MASTER on its [CYC_I(n)] input, then the next arbitration will be assigned to the MASTER on its [CYC_I(n-1)] input.

FAIR ARBITER
A WISHBONE functional module used with the standard or FIRECRACKER shared bus interconnections, which grants access to the INTERCON one MASTER at a time, on a statistically equal basis.

PRI ARBITER
● A WISHBONE functional module used with the standard or FIRECRACKER shared bus interconnections, which grants access to the INTERCON one MASTER at a time, on a priority basis. ● 'PRI' is an acronym for 'PRIority'. ■ *Ex.* A *PRI ARBITER* assigns a higher priority to the MASTER on its [CYC_I(n)] input that it does to the MASTER on its [CYC_I(n-1)] input.

XBAR ARBITER
A WISHBONE functional module used with the crossbar INTERCON, which connects pairs of MASTER and SLAVE interfaces together.

These kinds of multiple block scheduling conflicts can be resolved with a PRP ARBITER[18], which supervises local arbiters by providing a high level scheduling function.

Example: Common-clock BRIDGE

This example demonstrates an automatic, bidirectional, common-clock BRIDGE that supports all combinations of DATUM INTERLOCKED cycles, including:

```
        CONGA{
                        DATUM INTERLOCKED( READ() ),
                        DATUM INTERLOCKED( WRITE() ),
                        DATUM INTERLOCKED( DUPLEX() ),
                        DATUM INTERLOCKED( EVENT() )
        };
```

This section provides a CLASS Machine design representation for this type of BRIDGE. There are no special portability requirements needed for this design. It has a closed-loop $LCCP_{REL}$ = +2 ILUT, leading balance $LCCP_{REL}$ = +2 ILUT, with no lagging balance.

Figure 9 shows the control transport structure, Figure 10 the data transport structure, and Figures 11 and 12 the logic for the common-clock BRIDGE.

Figures 13 and 14 show timing diagrams for typical primary-to-secondary and secondary-to-primary BRIDGE unidirectional transactions (respectively). Figure 15 provides a timing diagram showing reset timing in conjunction with the earliest possible secondary-to-primary BRIDGE transaction.

[18] ***PRP ARBITER***

● A WISHBONE functional module used with the standard or FIRECRACKER shared bus interconnections, which grants access to the INTERCON on a programmable priority basis. It can also be used in a blocking system to schedule bus ownership among multiple MASTERs operating on multiple interconnections. It is capable of implementing a FAIR ARBITER. ● 'PRP' is an acronym for 'PRogrammable Priority'. ■ *Ex.* The *PRP ARBITER* changes the priority of each individual MASTER on a programmable basis. The scheduling algorithm is application dependent; any scheme can be used so long as it conforms to the basic timing requirements given in the WISHBONE specification. The priority level is configured in a RAM or ROM memory element so it can be changed at at run-time or at synthesis time, respectively.

. . .

The functions of the PRP ARBITER are analogous to a train dispatcher in the British block system of railway signaling. Two shared bus interconnections might have local RRS ARBITERs that grant equal priority to their respective local MASTERs, but the PRP ARBITER would supervise their operation for scheduling purposes. Under programmable control, it would cause one MASTER on each interconnection to take on a higher priority than the others. This situation is analogous to local freight being scheduled by each of the local RRS ARBITERs, but an express train 'passenger through-service' would be scheduled by the PRP ARBITER.

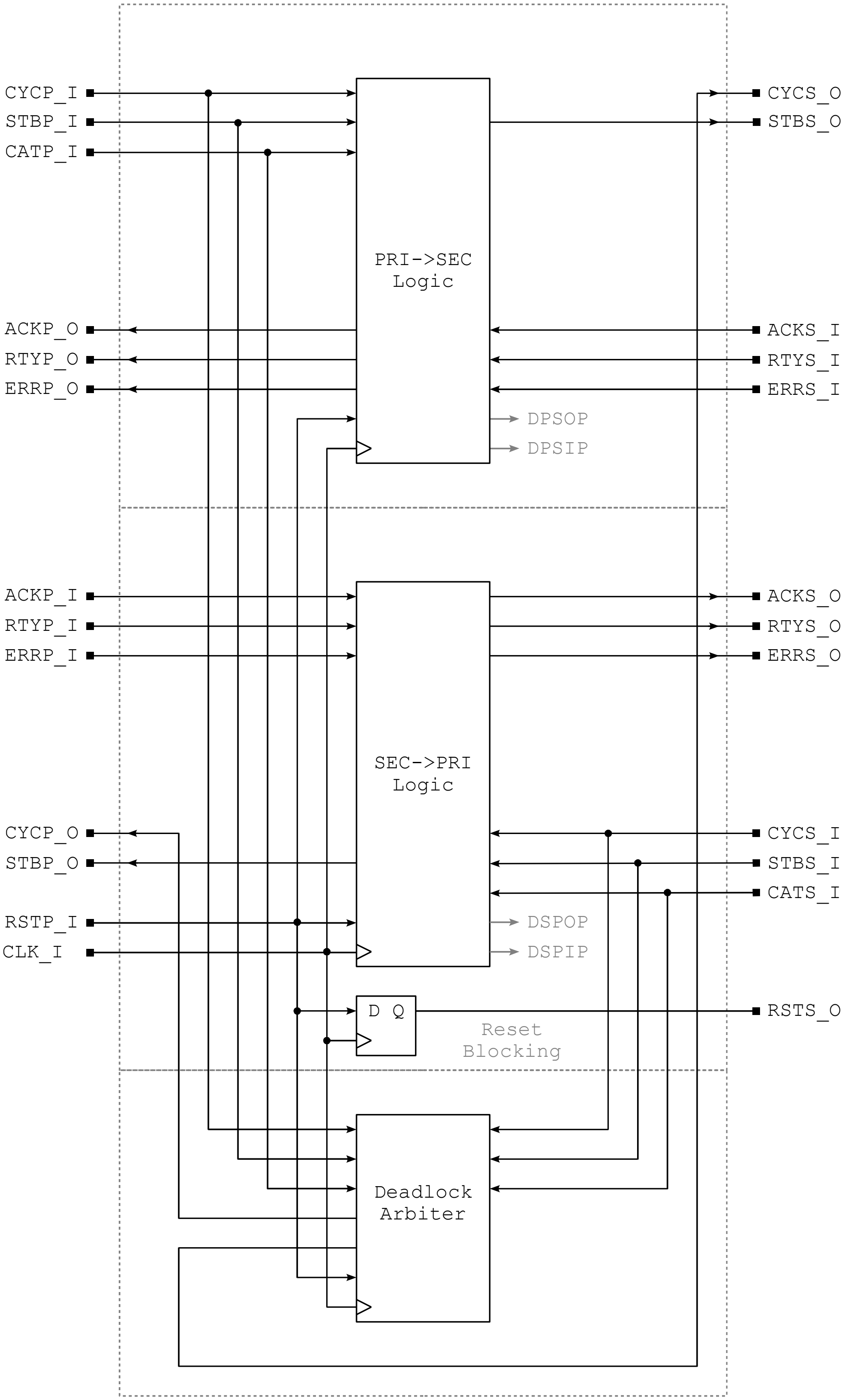
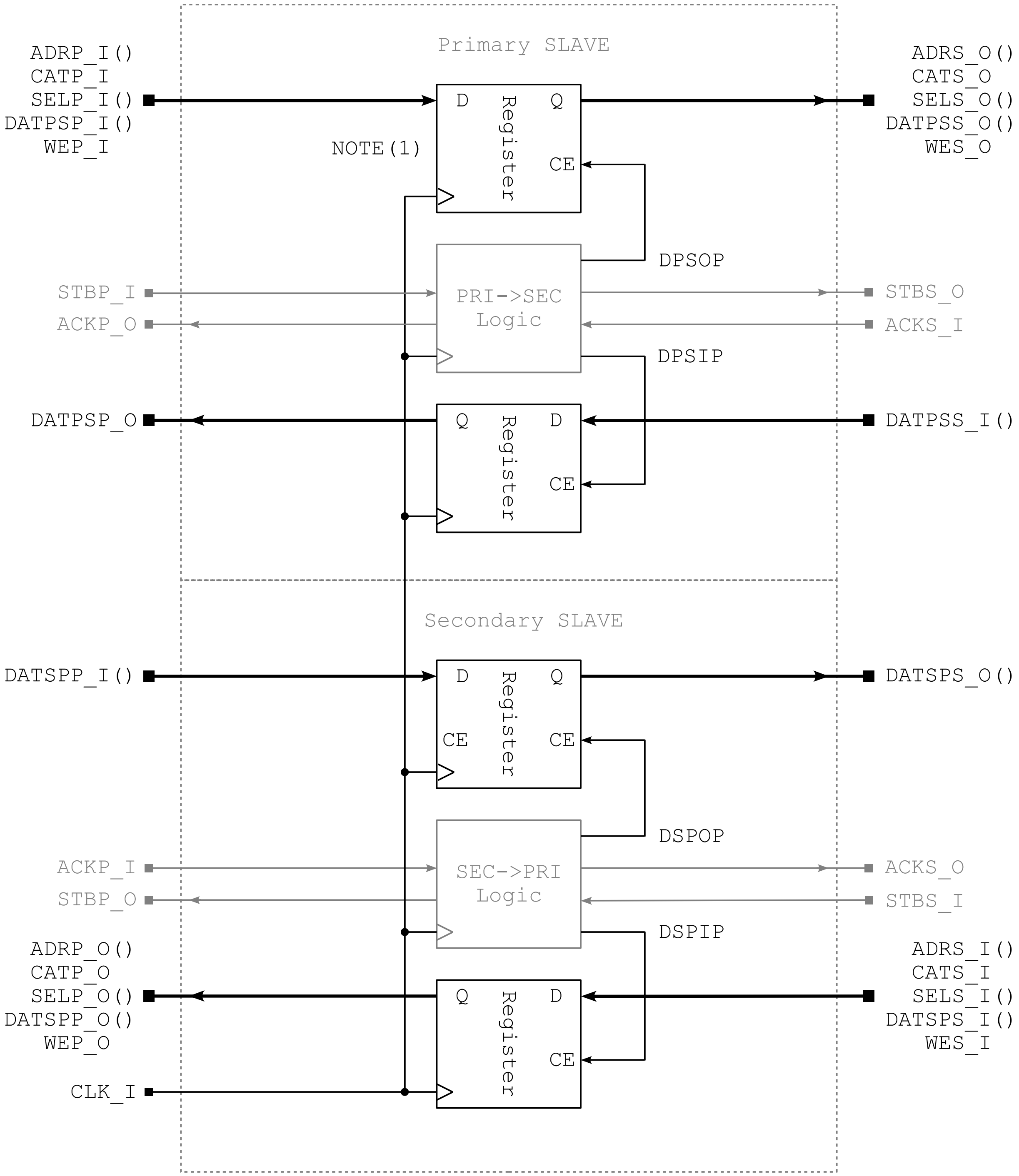


Figure 9. Control transport structure for the common-clock BRIDGE.



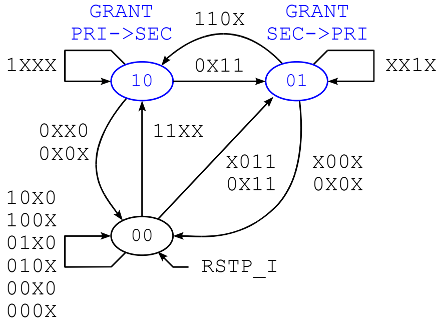
Notes:

(1) Typical register equations for a single bit:

```
ADRS_O(00) := ADRP_I(00) &  DPSOP
            + ADRS_O(00) & /DPSOP;
```

Figure 10. Data transport structure for the common-clock BRIDGE.

State Diagram
Deadlock Arbiter



```
STATES: CYCS_O,CYCP_O
INPUTS: CYCP_I,STBP,CYCS_I,STBS

CYCP_I:  Cycle,  PRI
CYCS_I:  Cycle,  SEC
GTPSX: Grant,  PRI->SEC
GTSSX: Grant,  SEC->PRI
RST:   System reset
STBP:  Strobe, PRI
STBS:  Strobe, SEC
```

General Assumptions

Initial cycle from either side must assert [CYC] and [STB] to start a cycle because signal [CYC] is broadcast to all SLAVEs, without address decoding.

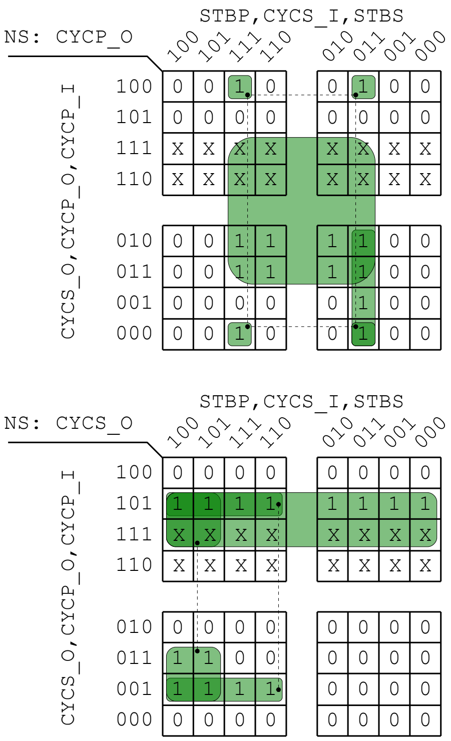
```
 STBS =  STBS_I & /CATS_I;
 STBP =  STBP_I & /CATP_I;
```

By DeMorgan's theorem:

```
/STBP = /STBP_I
      +          CATP_I;
```

(*) Primary has priority over sec.

Karnaugh Map
Deadlock Arbiter



STBP,CYCS_I,STBS

CYCS_O,CYCP_O,CYCP_I

NS: CYCP_O	100	101	111	110	010	011	001	000
100	0	0	1	0	0	1	0	0
101	0	0	0	0	0	0	0	0
111	X	X	X	X	X	X	X	X
110	X	X	X	X	X	X	X	X
010	0	0	1	1	1	1	0	0
011	0	0	1	1	1	1	0	0
001	0	0	0	0	0	1	0	0
000	0	0	1	0	0	1	0	0

STBP,CYCS_I,STBS

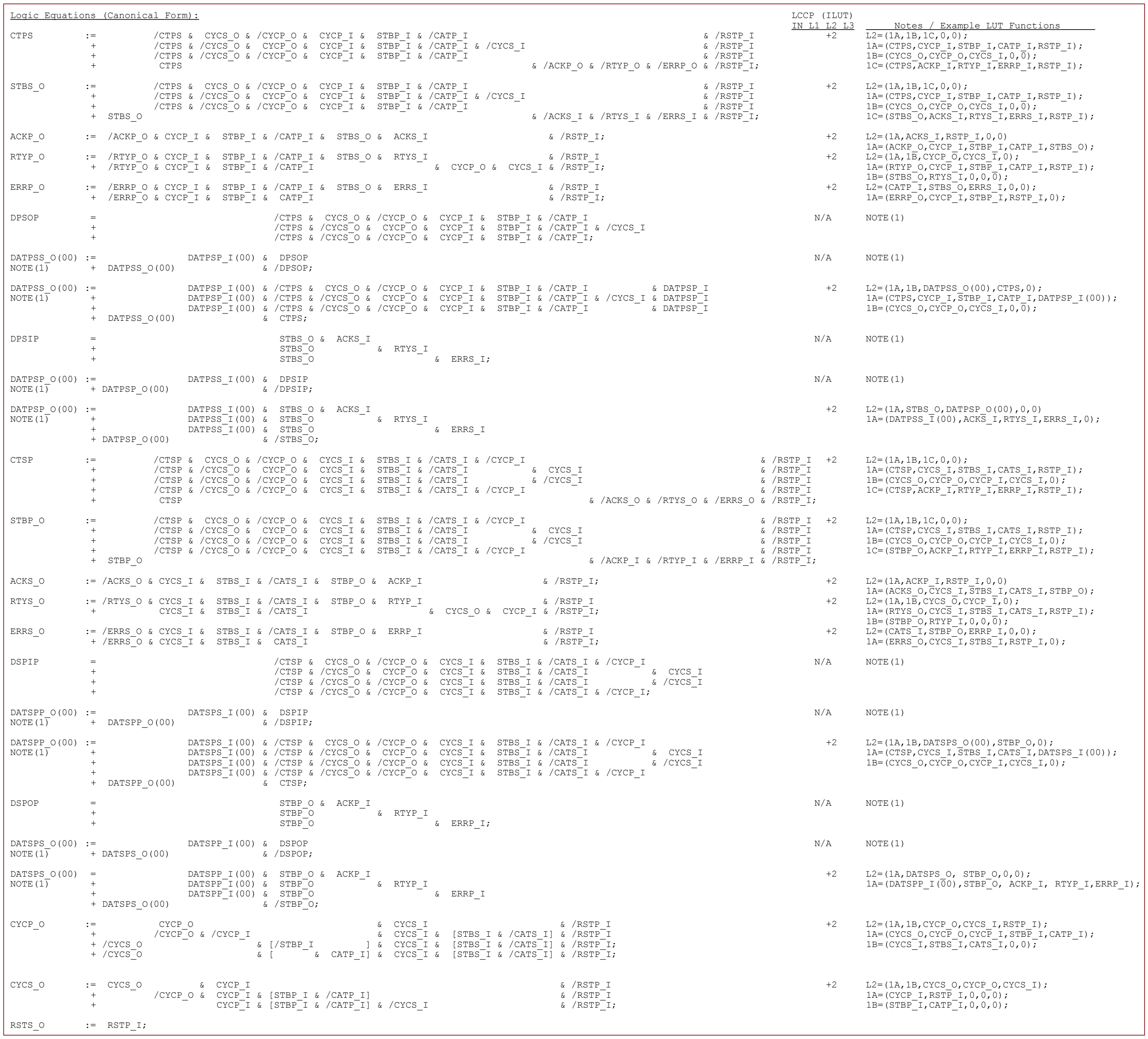
CYCS_O,CYCP_O,CYCP_I

NS: CYCS_O	100	101	111	110	010	011	001	000
100	0	0	0	0	0	0	0	0
101	1	1	1	1	1	1	1	1
111	X	X	X	X	X	X	X	X
110	X	X	X	X	X	X	X	X
010	0	0	0	0	0	0	0	0
011	1	1	0	0	0	0	0	0
001	1	1	1	1	0	0	0	0
000	0	0	0	0	0	0	0	0

Truth Table
Deadlock Arbiter

```
CCCSCS  CC  |  CCCSCS  CC
YYYTYT  YY  |  YYYTYT  YY
CCCBCB  CC  |  CCCBCB  CC
PSPPSS  SP  |  PSPPSS  SP
OOI  I  OO  |  OOI  I  OO
------  --  |  ------  --
100000  00  |  010000  00
100001  00  |  010001  00
100011  01  |  010011  01
100010  00  |  010010  01

100110  00  |  010110  01
100111  01  |  010111  01
100101  00  |  010101  00
100100  00  |  010100  00

101100  10  |  011100  10
101101  10  |  011101  10
101111  10  |  011111  01
101110  10  |  011110  01

101010  10  |  011010  01
101011  10  |  011011  01
101001  10  |  011001  00
101000  10  |  011000  00

111000  XX  |  001000  00
111001  XX  |  001001  00
111011  XX  |  001011  01
111010  XX  |  001010  00

111110  XX  |  001110  10
111111  XX  |  001111  10(*)
111101  XX  |  001101  10
111100  XX  |  001100  10

110100  XX  |  000100  00
110101  XX  |  000101  00
110111  XX  |  000111  01
110110  XX  |  000110  00

110010  XX  |  000010  00
110011  XX  |  000011  01
110001  XX  |  000001  00
110000  XX  |  000000  00
```

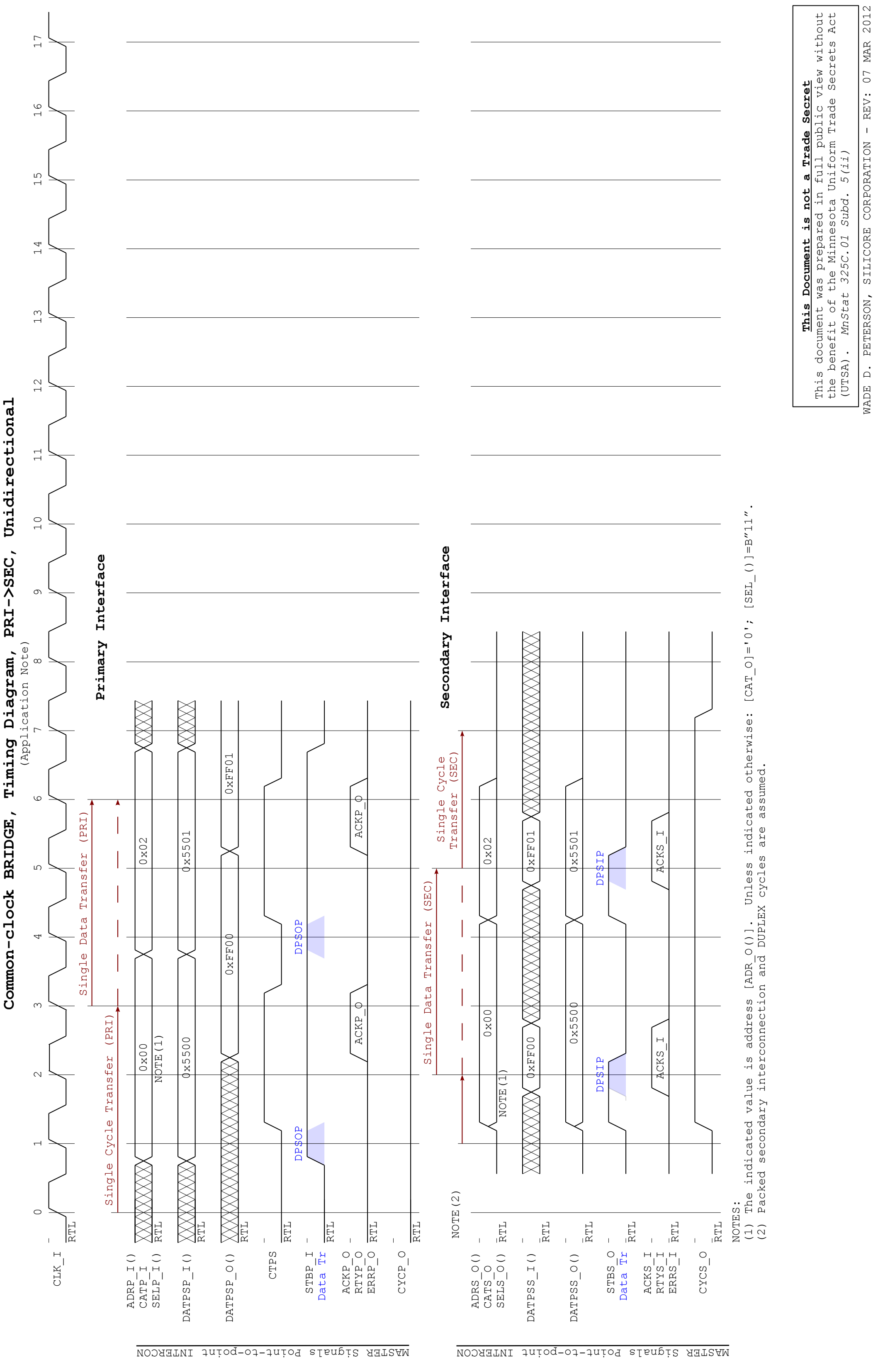
Figure 11. Deadlock arbiter logic for the common-clock BRIDGE.



```
Logic Equations (Canonical Form):                                                                                                        LCCP (ILUT)
                                                                                                                                         IN L1 L2 L3       Notes / Example LUT Functions
CTPS         :=          /CTPS &  CYCS_O & /CYCP_O &  CYCP_I &  STBP_I & /CATP_I                                         & /RSTP_I             +2     L2=(1A,1B,1C,0,0);
              +          /CTPS & /CYCS_O &  CYCP_O &  CYCP_I &  STBP_I & /CATP_I & /CYCS_I                               & /RSTP_I                    1A=(CTPS,CYCP_I,STBP_I,CATP_I,RSTP_I);
              +          /CTPS & /CYCS_O & /CYCP_O &  CYCP_I &  STBP_I & /CATP_I                                         & /RSTP_I                    1B=(CYCS_O,CYCP_O,CYCS_I,0,0);
              +           CTPS                                                             & /ACKP_O & /RTYP_O & /ERRP_O & /RSTP_I;                   1C=(CTPS,ACKP_I,RTYP_I,ERRP_I,RSTP_I);

STBS_O       :=          /CTPS &  CYCS_O & /CYCP_O &  CYCP_I &  STBP_I & /CATP_I                                         & /RSTP_I             +2     L2=(1A,1B,1C,0,0);
              +          /CTPS & /CYCS_O &  CYCP_O &  CYCP_I &  STBP_I & /CATP_I & /CYCS_I                               & /RSTP_I                    1A=(CTPS,CYCP_I,STBP_I,CATP_I,RSTP_I);
              +          /CTPS & /CYCS_O & /CYCP_O &  CYCP_I &  STBP_I & /CATP_I                                         & /RSTP_I                    1B=(CYCS_O,CYCP_O,CYCS_I,0,0);
              +  STBS_O                                                                    & /ACKS_I & /RTYS_I & /ERRS_I & /RSTP_I;                   1C=(STBS_O,ACKS_I,RTYS_I,ERRS_I,RSTP_I);

ACKP_O       :=  /ACKP_O & CYCP_I &  STBP_I & /CATP_I &  STBS_O &  ACKS_I                     & /RSTP_I;                                       +2     L2=(1A,ACKS_I,RSTP_I,0,0)
                                                                                                                                                      1A=(ACKP_O,CYCP_I,STBP_I,CATP_I,STBS_O);
RTYP_O       :=  /RTYP_O & CYCP_I &  STBP_I & /CATP_I &  STBS_O &  RTYS_I                     & /RSTP_I                                        +2     L2=(1A,1B,CYCP_O,CYCS_I,0);
              +  /RTYP_O & CYCP_I &  STBP_I & /CATP_I                     & CYCP_O &  CYCS_I & /RSTP_I;                                               1A=(RTYP_O,CYCP_I,STBP_I,CATP_I,RSTP_I);
                                                                                                                                                      1B=(STBS_O,RTYS_I,0,0,0);
ERRP_O       :=  /ERRP_O & CYCP_I &  STBP_I & /CATP_I &  STBS_O &  ERRS_I                     & /RSTP_I                                        +2     L2=(CATP_I,STBS_O,ERRS_I,0,0);
              +  /ERRP_O & CYCP_I &  STBP_I &  CATP_I                                         & /RSTP_I;                                              1A=(ERRP_O,CYCP_I,STBP_I,RSTP_I,0);

DPSOP        =                                /CTPS &  CYCS_O & /CYCP_O &  CYCP_I &  STBP_I & /CATP_I                                        N/A      NOTE(1)
              +                               /CTPS & /CYCS_O &  CYCP_O &  CYCP_I &  STBP_I & /CATP_I & /CYCS_I
              +                               /CTPS & /CYCS_O & /CYCP_O &  CYCP_I &  STBP_I & /CATP_I;

DATPSS_O(00) :=                DATPSP_I(00) &  DPSOP                                                                                         N/A      NOTE(1)
NOTE(1)       +  DATPSS_O(00)               & /DPSOP;

DATPSS_O(00) :=                DATPSP_I(00) & /CTPS &  CYCS_O & /CYCP_O &  CYCP_I &  STBP_I & /CATP_I           & DATPSP_I                     +2     L2=(1A,1B,DATPSS_O(00),CTPS,0);
NOTE(1)       +                DATPSP_I(00) & /CTPS & /CYCS_O &  CYCP_O &  CYCP_I &  STBP_I & /CATP_I & /CYCS_I & DATPSP_I                           1A=(CTPS,CYCP_I,STBP_I,CATP_I,DATPSP_I(00));
              +                DATPSP_I(00) & /CTPS & /CYCS_O & /CYCP_O &  CYCP_I &  STBP_I & /CATP_I           & DATPSP_I                           1B=(CYCS_O,CYCP_O,CYCS_I,0,0);
              +  DATPSS_O(00)               &  CTPS;

DPSIP        =                                 STBS_O &  ACKS_I                                                                              N/A      NOTE(1)
              +                                STBS_O           &  RTYS_I
              +                                STBS_O                     &  ERRS_I;

DATPSP_O(00) :=                DATPSS_I(00) &  DPSIP                                                                                         N/A      NOTE(1)
NOTE(1)       + DATPSP_O(00)                & /DPSIP;

DATPSP_O(00) :=                DATPSS_I(00) &  STBS_O &  ACKS_I                                                                                +2     L2=(1A,STBS_O,DATPSP_O(00),0,0)
NOTE(1)       +                DATPSS_I(00) &  STBS_O           &  RTYS_I                                                                             1A=(DATPSS_I(00),ACKS_I,RTYS_I,ERRS_I,0);
              +                DATPSS_I(00) &  STBS_O                     &  ERRS_I
              + DATPSP_O(00)                & /STBS_O;

CTSP         :=          /CTSP &  CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I & /CYCP_I                                         & /RSTP_I  +2     L2=(1A,1B,1C,0,0);
              +          /CTSP & /CYCS_O &  CYCP_O &  CYCS_I &  STBS_I & /CATS_I           &  CYCS_I                               & /RSTP_I         1A=(CTSP,CYCS_I,STBS_I,CATS_I,RSTP_I);
              +          /CTSP & /CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I           & /CYCS_I                               & /RSTP_I         1B=(CYCS_O,CYCP_O,CYCP_I,CYCS_I,0);
              +          /CTSP & /CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I & /CYCP_I                                         & /RSTP_I         1C=(CTSP,ACKP_I,RTYP_I,ERRP_I,RSTP_I);
              +           CTSP                                                                       & /ACKS_O & /RTYS_O & /ERRS_O & /RSTP_I;

STBP_O       :=          /CTSP &  CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I & /CYCP_I                                         & /RSTP_I  +2     L2=(1A,1B,1C,0,0);
              +          /CTSP & /CYCS_O &  CYCP_O &  CYCS_I &  STBS_I & /CATS_I           &  CYCS_I                               & /RSTP_I         1A=(CTSP,CYCS_I,STBS_I,CATS_I,RSTP_I);
              +          /CTSP & /CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I           & /CYCS_I                               & /RSTP_I         1B=(CYCS_O,CYCP_O,CYCP_I,CYCS_I,0);
              +          /CTSP & /CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I & /CYCP_I                                         & /RSTP_I         1C=(STBP_O,ACKP_I,RTYP_I,ERRP_I,RSTP_I);
              +  STBP_O                                                                              & /ACKP_I & /RTYP_I & /ERRP_I & /RSTP_I;

ACKS_O       := /ACKS_O & CYCS_I &  STBS_I & /CATS_I &  STBP_O &  ACKP_I                     & /RSTP_I;                                        +2     L2=(1A,ACKP_I,RSTP_I,0,0)
                                                                                                                                                      1A=(ACKS_O,CYCS_I,STBS_I,CATS_I,STBP_O);
RTYS_O       := /RTYS_O & CYCS_I &  STBS_I & /CATS_I &  STBP_O &  RTYP_I                     & /RSTP_I                                         +2     L2=(1A,1B,CYCS_O,CYCP_I,0);
              +          CYCS_I &  STBS_I & /CATS_I                     & CYCS_O &  CYCP_I & /RSTP_I;                                                 1A=(RTYS_O,CYCS_I,STBS_I,CATS_I,RSTP_I);
                                                                                                                                                      1B=(STBP_O,RTYP_I,0,0,0);
ERRS_O       := /ERRS_O & CYCS_I &  STBS_I & /CATS_I &  STBP_O &  ERRP_I                     & /RSTP_I                                         +2     L2=(CATS_I,STBP_O,ERRP_I,0,0);
              + /ERRS_O & CYCS_I &  STBS_I &  CATS_I                                         & /RSTP_I;                                               1A=(ERRS_O,CYCS_I,STBS_I,RSTP_I,0);

DSPIP        =                                /CTSP &  CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I & /CYCP_I                              N/A      NOTE(1)
              +                               /CTSP & /CYCS_O &  CYCP_O &  CYCS_I &  STBS_I & /CATS_I           &  CYCS_I
              +                               /CTSP & /CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I           & /CYCS_I
              +                               /CTSP & /CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I & /CYCP_I;

DATSPP_O(00) :=                DATSPS_I(00) &  DSPIP                                                                                         N/A      NOTE(1)
NOTE(1)       +  DATSPP_O(00)               & /DSPIP;

DATSPP_O(00) :=                DATSPS_I(00) & /CTSP &  CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I & /CYCP_I                                +2     L2=(1A,1B,DATSPS_O(00),STBP_O,0);
NOTE(1)       +                DATSPS_I(00) & /CTSP & /CYCS_O &  CYCP_O &  CYCS_I &  STBS_I & /CATS_I           &  CYCS_I                            1A=(CTSP,CYCS_I,STBS_I,CATS_I,DATSPS_I(00));
              +                DATSPS_I(00) & /CTSP & /CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I           & /CYCS_I                            1B=(CYCS_O,CYCP_O,CYCP_I,CYCS_I,0);
              +                DATSPS_I(00) & /CTSP & /CYCS_O & /CYCP_O &  CYCS_I &  STBS_I & /CATS_I & /CYCP_I
              +  DATSPP_O(00)               &  CTSP;

DSPOP        =                                 STBP_O &  ACKP_I                                                                              N/A      NOTE(1)
              +                                STBP_O           &  RTYP_I
              +                                STBP_O                     &  ERRP_I;

DATSPS_O(00) :=                DATSPP_I(00) &  DSPOP                                                                                         N/A      NOTE(1)
NOTE(1)       + DATSPS_O(00)                & /DSPOP;

DATSPS_O(00)  =                DATSPP_I(00) &  STBP_O &  ACKP_I                                                                                +2     L2=(1A,DATSPS_O, STBP_O,0,0);
NOTE(1)       +                DATSPP_I(00) &  STBP_O           &  RTYP_I                                                                             1A=(DATSPP_I(00),STBP_O, ACKP_I, RTYP_I,ERRP_I);
              +                DATSPP_I(00) &  STBP_O                     &  ERRP_I
              + DATSPS_O(00)                & /STBP_O;

CYCP_O       :=           CYCP_O                                &  CYCS_I                        & /RSTP_I                                      +2     L2=(1A,1B,CYCP_O,CYCS_I,RSTP_I);
              +          /CYCP_O & /CYCP_I                      &  CYCS_I &  [STBS_I & /CATS_I] & /RSTP_I                                             1A=(CYCS_O,CYCP_O,CYCP_I,STBP_I,CATP_I);
              + /CYCS_O                   & [/STBP_I         ] &  CYCS_I &  [STBS_I & /CATS_I] & /RSTP_I;                                             1B=(CYCS_I,STBS_I,CATS_I,0,0);
              + /CYCS_O                   & [       &  CATP_I] &  CYCS_I &  [STBS_I & /CATS_I] & /RSTP_I;


CYCS_O       :=  CYCS_O          &  CYCP_I                                                        & /RSTP_I                                      +2     L2=(1A,1B,CYCS_O,CYCP_O,CYCS_I);
              +          /CYCP_O &  CYCP_I & [STBP_I & /CATP_I]                                   & /RSTP_I                                             1A=(CYCP_I,RSTP_I,0,0,0);
              +                     CYCP_I & [STBP_I & /CATP_I] & /CYCS_I                         & /RSTP_I;                                            1B=(STBP_I,CATP_I,0,0,0);

RSTS_O       :=  RSTP_I;
```

Notes:
(1) Representative of all data transport signals driven by the clock enable signal above. The equation is expanded below, with the complete LCCP estimate.

Figure 12. Logic for the common-clock BRIDGE.



NOTES:
(1) The indicated value is address [ADR_O()]. Unless indicated otherwise: [CAT_O]='0'; [SEL_()]=B"11".
(2) Packed secondary interconnection and DUPLEX cycles are assumed.

This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Subd. 5(ii)*

WADE D. PETERSON, SILICORE CORPORATION - REV: 07 MAR 2012

Figure 13. Typical primary-to-secondary transactions for the common-clock BRIDGE.

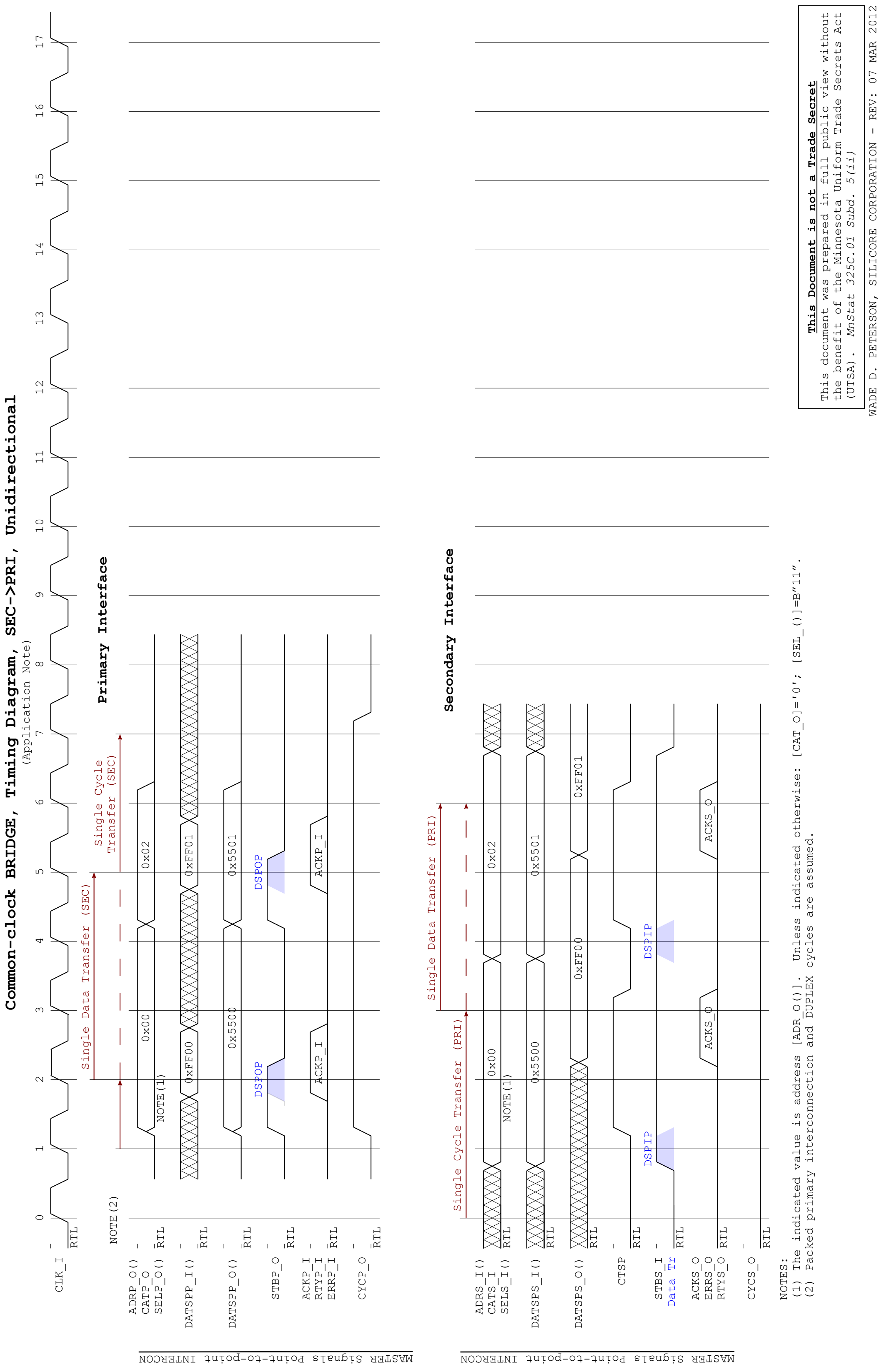
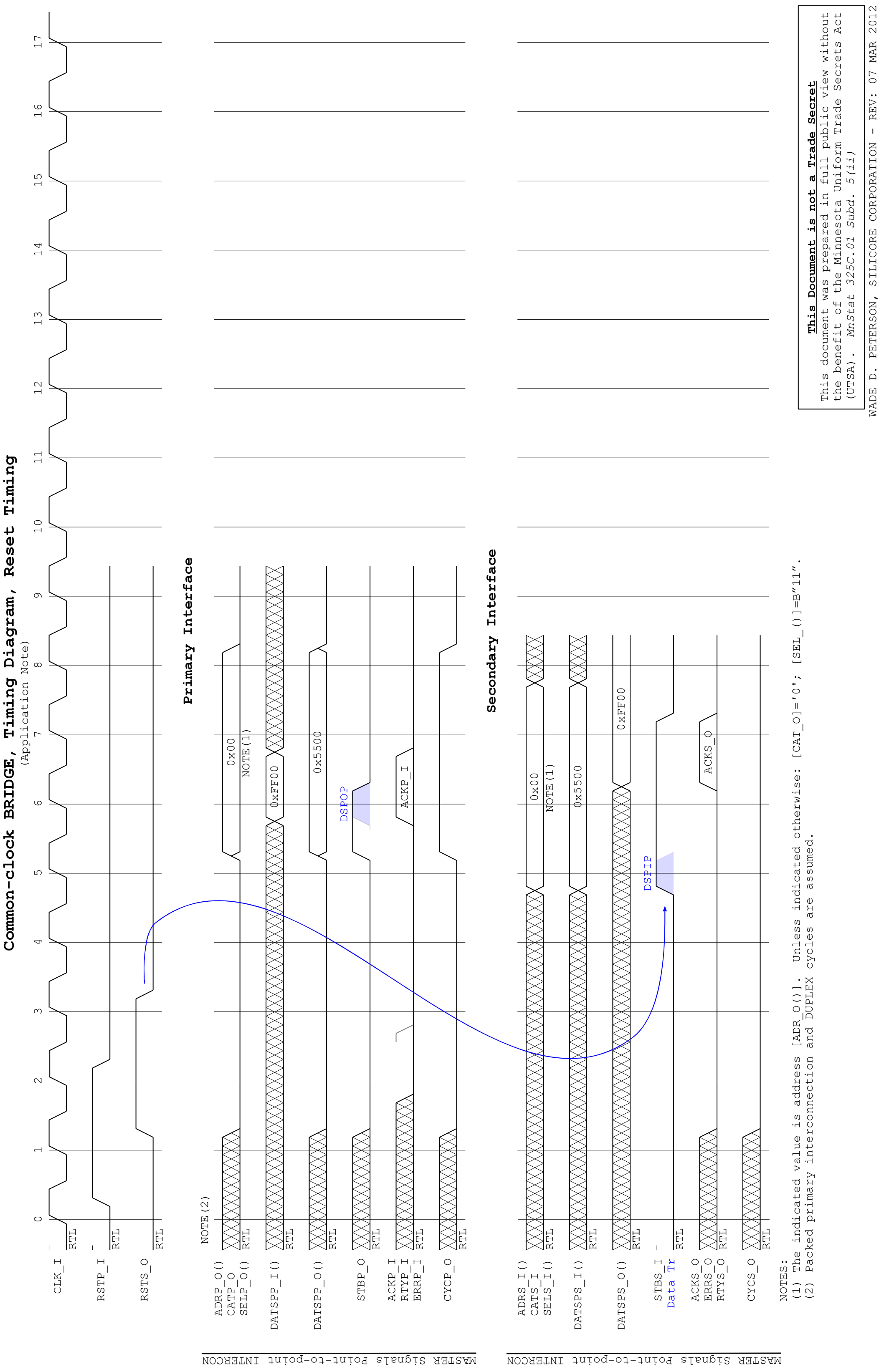

Common-clock BRIDGE, Timing Diagram, SEC->PRI, Unidirectional
(Application Note)
Primary Interface
Secondary Interface
CLK_I
NOTE(2)
ADRP_O()
CATP_O
SELP_O()
DATSPP_I()
DATSPP_O()
STBP_O
ACKP_I
RTYP_I
ERRP_I
CYCP_O
ADRS_I()
CATS_I
SELS_I()
DATSPS_I()
DATSPS_O()
CTSP
STBS_I
Data Tr
ACKS_O
ERRS_O
RTYS_O
CYCS_O
RTL
MASTER Signals Point-to-point INTERCON
Single Cycle Transfer (SEC)
Single Data Transfer (SEC)
Single Cycle Transfer (PRI)
Single Data Transfer (PRI)
0x00
0x02
0xFF00
0xFF01
0x5500
0x5501
NOTE(1)
DSPOP
DSPIP
ACKP_I
ACKS_O
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
NOTES:
(1) The indicated value is address [ADR_O()]. Unless indicated otherwise: [CAT_O]='0'; [SEL_()]=B"11".
(2) Packed primary interconnection and DUPLEX cycles are assumed.
This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). MnStat 325C.01 Subd. 5(ii)
WADE D. PETERSON, SILICORE CORPORATION - REV: 07 MAR 2012

Figure 14. Typical secondary-to-primary transactions for the common-clock BRIDGE.



NOTES:
(1) The indicated value is address [ADR_O()]. Unless indicated otherwise: [CAT_O]='0'; [SEL_()]=B"11".
(2) Packed primary interconnection and DUPLEX cycles are assumed.

This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Subd. 5(ii)*

WADE D. PETERSON, SILICORE CORPORATION - REV: 07 MAR 2012

Figure 15. Typical reset timing for the common-clock BRIDGE.

Performance Benchmarks for the Common-clock BRIDGE Example

Figure 16 is a synopsis of performance benchmarks for the common-clock BRIDGE example. The data was collected using the graphical technique on a scaled timing diagram (see red lines on Figures 13 and 14).

Figure 16(a) shows the closed-loop control transfer clock efficiency (Eff, clk) for packed and unpacked point-to-point interconnections located on the MASTER side of the BRIDGE. This metric shows the number of clock cycles required to transfer data across the BRIDGE, and applies equally to READ, WRITE, DUPLEX and EVENT cycles. Noteworthy observations include:

- *Each data transfer phase is identical*. So long as the remote MASTER and SLAVE interfaces are repeatable, each phase of a multi-phase data transfer cycle requires the same number of clock cycles to complete.

- *Unidirectional is the same as bidirectional*. The clock efficiency is the same regardless of whether it is a unidirectional or a bidirectional BRIDGE.

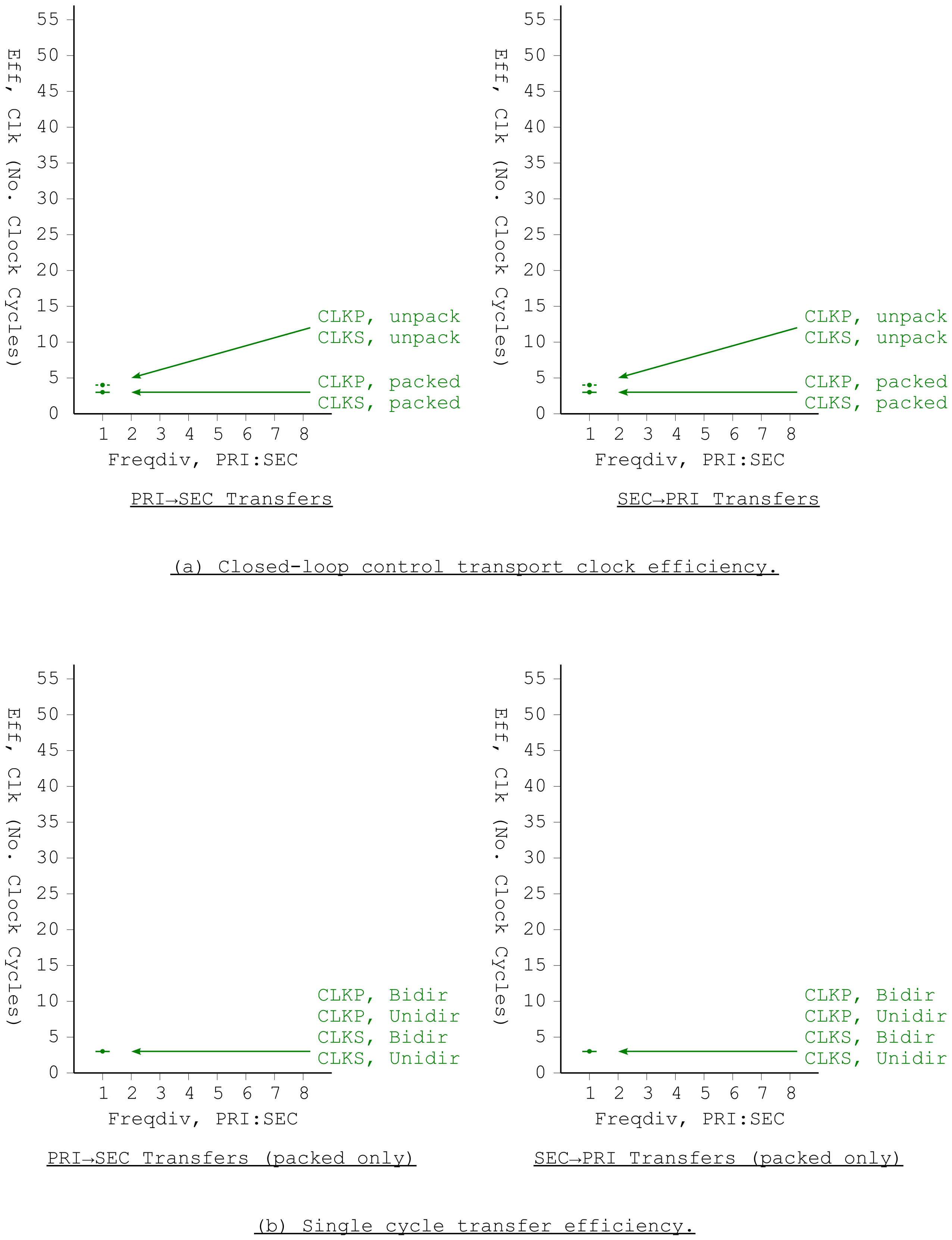
- *Overall clock efficiencies are low, but highest for a BRIDGE*. The overall clock efficiency compares data transfers across a BRIDGE versus across a point-to-point interconnection. It is useful for understanding the effects of blocking. This metric assumes a baseline reference for each point-to-point transfer of one (1) packed and two (2) unpacked clock cycles.

  A packed PRI->SEC or SEC-> PRI common-clock BRIDGE transfer requires three (3) primary clock cycles to complete, which results in an overall clock efficiency of 3:1, or about 33%. An unpacked transfer under the same conditions requires four (4) primary clock cycles, resulting in an overall clock efficiency of 4:1, or 25%.

Figure 16(b) shows the same data transfer scenarios, except from the perspective of the cycle signal [CYC]. Only packed cycles are noted. Noteworthy results include:

- *Deadlock arbiter has no effect*. Because of the design of the common-clock BRIDGE example, there is no overhead caused by the deadlock arbiter. Arbitration functions have been fully integrated into the logic design so that no additional clock cycles are necessary.

- *No disconnect penalty*. There is no disconnect penalty associated with the cycle signal [CYC] in either the unidirectional or bidirectional BRIDGE topologies.



(a) Closed-loop control transport clock efficiency.

(b) Single cycle transfer efficiency.

Figure 16. Performance benchmarks for the common-clock BRIDGE example.

Example: Video Swap Buffer

This example demonstrates the idea of blocking, and a form of parallelism which allows data to be transported across the BRIDGE in both directions at the same time: the DUPLEX[19] data transport cycle.

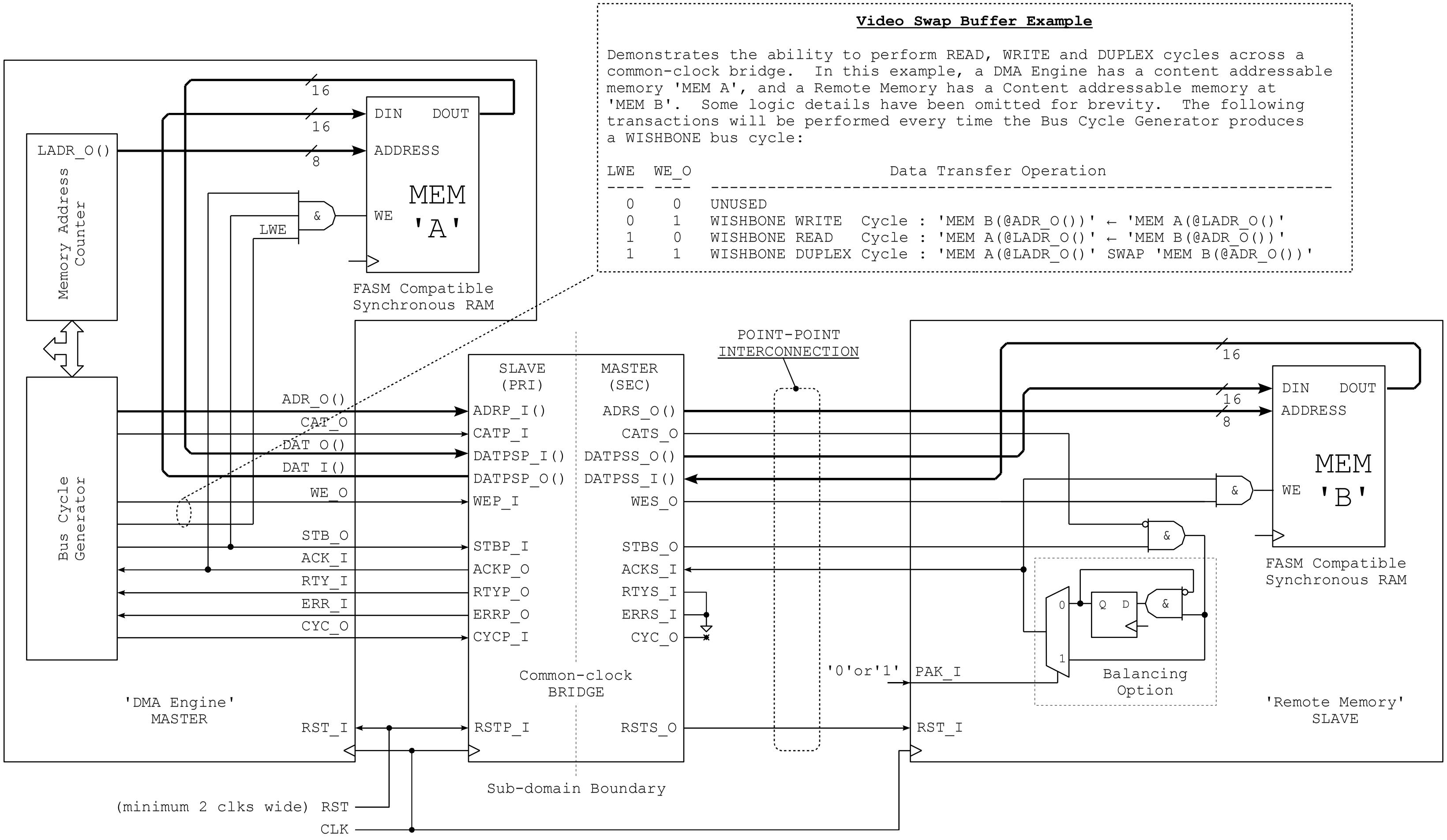
One of the reasons for using a common-clock BRIDGE is to divide an interconnection – and its clock sub-domain region – into two smaller parts. This increases the overall speed of the SoC. Figure 17 shows a video swap buffer example that divides a point-to-point interconnection into two blocks; one that has a DMA engine and another that has a memory block.

A video swap buffer is used by a video controller to substitute one graphic display image for another. For example, a video display controller on a personal computer could use it to replace a small section of the video display memory with the mouse cursor image (e.g. an arrow or a dogbone). Or, a weather radar display might show (substitute) blue rain instead of a green image of the ground.

The example shows a DMA engine has an internal content addressable video display memory 'MEM A', and a remote content addressable memory at 'MEM B', which contains the replacement image. These are located on opposites sides of the common-clock BRIDGE. Because it supports the DUPLEX transaction, the memory location at 'MEM A (@ LADR_O())' can be swapped with the one at 'MEM B (@ADR_O())' in a single bus cycle. When the original image needs to be restored, the whole process is reversed.

The DUPLEX cycle is useful whenever data is transported simultaneously in two directions. Other examples where it might be useful include full duplex radio, telephone and video traffic and submission and retrieval of information to and from a floating point unit, DSP or other numeric processor.

[19] ***DUPLEX***
A data transport attribute for a WISHBONE bus cycle that simultaneously transfers data in two directions between a MASTER interface and a selected SLAVE interface. This activity is prearranged by the system integrator, as there is no special control transport signaling which indicates a bidirectional transfer. The interfaces perform bidirectional transfers only when the write-enable signal [WE] is asserted. Otherwise, when write-enable [WE] is negated, a standard READ cycle is performed. A MASTER interface may ignore any read data during a bidirectional transfer. It cannot be used during a BROADCAST cycle.



Video Swap Buffer Example

Demonstrates the ability to perform READ, WRITE and DUPLEX cycles across a common-clock bridge. In this example, a DMA Engine has a content addressable memory 'MEM A', and a Remote Memory has a Content addressable memory at 'MEM B'. Some logic details have been omitted for brevity. The following transactions will be performed every time the Bus Cycle Generator produces a WISHBONE bus cycle:

LWE	WE_O	Data Transfer Operation
0	0	UNUSED
0	1	WISHBONE WRITE Cycle : 'MEM B(@ADR_O())' ← 'MEM A(@LADR_O()'
1	0	WISHBONE READ Cycle : 'MEM A(@LADR_O()' ← 'MEM B(@ADR_O())'
1	1	WISHBONE DUPLEX Cycle : 'MEM A(@LADR_O()' SWAP 'MEM B(@ADR_O())'

Figure 17. Video swap buffer example.

Chapter 3 - The Synchronized Trans-domain BRIDGE

The synchronized trans-domain BRIDGE allows the transport of WISHBONE DATUM INTERLOCKED bus cycles between two synchronized clock domains. This allows different parts of a value-added data process[20] to operate at different clock frequencies.

Trans-domain BRIDGE structures are notoriously inefficient, meaning it takes lots of clock cycles to interlock signals across the trans-domain boundary. Of the three types of boundary discussed above, the asynchronous boundary is the least efficient and least deterministic. One way to improve both of these factors is by synchronizing the two clocks at the domain crossing boundary. This improves the efficiency of the BRIDGE because it eliminates – or rather controls – signal re-synchronization, and it reduces the overall timing uncertainty. This is the subject of the instant chapter.

There are two general types of synchronized trans-domain BRIDGE. They are categorized by the clock divisor relationship that exists between the primary and secondary domains, and which is controlled by the SYSCON functional module. They are:

- *Synchronized unity BRIDGE*. A primary clock domain drives the secondary clock domain with a driver (buffer), and a primary-to-secondary frequency ratio of 1:1 exists. Used in conjunction with the synchronized unity trans-domain SYSCON.

  The synchronized unity BRIDGE and SYSCON are considered to be special cases (as compared to the non-unity variants) because they can only be used under the following conditions:

  - The primary and secondary clocks operate at the same frequency.
  - The primary and secondary $LCCP_{REL} \geq +2$ ILUT.
  - A unit or n-step clock controller cannot be used.

- *Synchronized non-unity BRIDGE*. A primary clock domain operates at a higher frequency than the secondary clock domain; where the primary and secondary clocks are separated by a counter; and where a primary-to-secondary frequency ratio equal to or greater than 2:1 exists (i.e. 2:1, 3:1, 4:1, etc.). Used in conjunction with the synchronized non-unity trans-domain SYSCON.

  The synchronized non-unity BRIDGE and SYSCON are considered to be general cases because they can be used with a large number of frequency divisor ratios, they support $LCCP_{REL} = +1$ ILUT, and they can be used with a unit or n-step clock controller.

[20] ***value-added data process***
● A method for adding value to information, such as might be performed by computing machinery or the human brain; a calculation. ● *System-on-Chip Operations Management (SoCOM)*. The modification of data as it flows between the input and the output of a chip-level system. Data is regarded as (and is analogous to) inventory as it moves through a supply chain. ■ *Ex*. The synchronized trans-domain BRIDGE allows different parts of a *value-added data process* to operate at different clock frequencies.

Before launching into the synchronized BRIDGE cases, it is useful to review faux clock domains for the purpose of administering domain crossing signals.

Faux Clock Domain

The faux clock domain – or false clock domain – is a method that uses an excitation waveform to control a clock enable input on a state machine. This allows the circuit to operate at a frequency which is less than the clock domain frequency; it allows the adjustment of power consumption by changing the dynamic power of the circuit; and it allows the adjustment of EMI/RFI emissions from the circuit.

The faux clock domain eliminates the need for additional clock distribution networks[21], and it maintains a portable CLASS Machine design methodology which is consistent with the other objectives of the WISHBONE SoC. Figure 18 shows representative faux clock domain excitation signals.

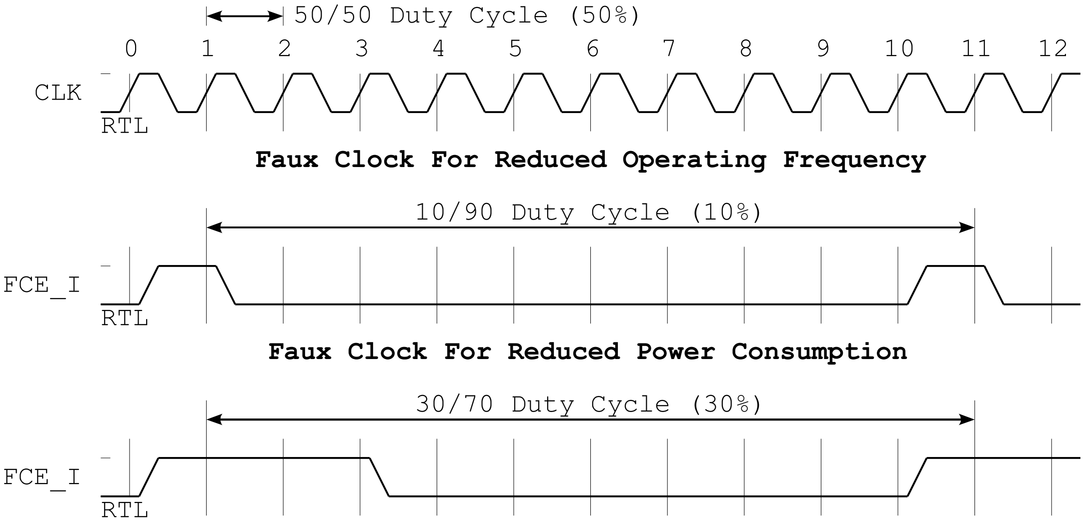


Figure 18. Faux clock domain excitation signals.

For example, a 4-bit synchronous counter in a software defined radio (SDR) receiver system is tuned to a frequency of 20.0000 MHz. If the faux clock domain of the counter is 10 MHz, then the first harmonic of the 10 MHz signal (which appears at 20 MHz) might interfere with radio reception. However, if the faux clock enable rate is adjusted to 11.111 MHz, then it is less likely that the EMI/RFI from the circuit will interfere with the radio receiver.

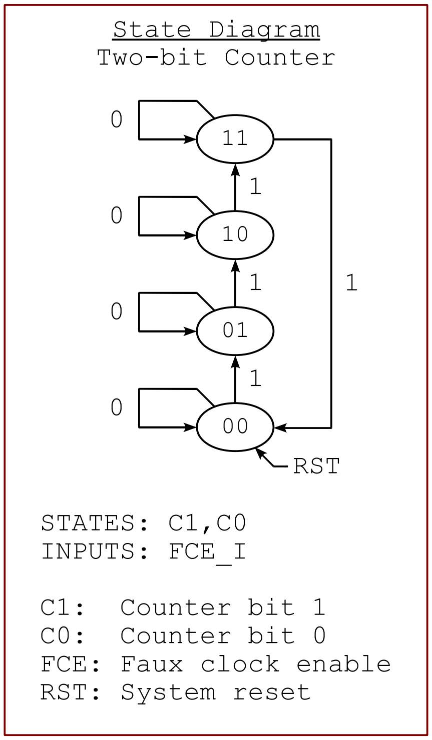
Administration of the faux clock domain is outside of the WISHBONE specification. However, its use is assumed in a number of situations, and should be regarded as a recommended practice. All logic in the faux clock controlled state machine must conform to the other timing specifications in every respect. Thus, even though the faux clock is used to operate a logic element at a lower frequency, it must conform to the standard WISHBONE timing parameters.

[21] Target devices have a limited number of low-skew clock distribution networks. This means that any requirement for additional clocks will reduce the portability of the logic design.

Effect of System Reset Signal [RST] on Faux Clock Logic

Logic which uses the faux clock domain is expected to respond to system reset [RST] as defined by the WISHBONE specification. Thus, once the system reset signal is asserted, all state machines are expected to initialize themselves regardless of the state of the faux clock enable input [FCE_I]. This is somewhat different than what might be expected from a typical clock enable input, where logic typically responds to a reset input only when the clock enable is asserted.

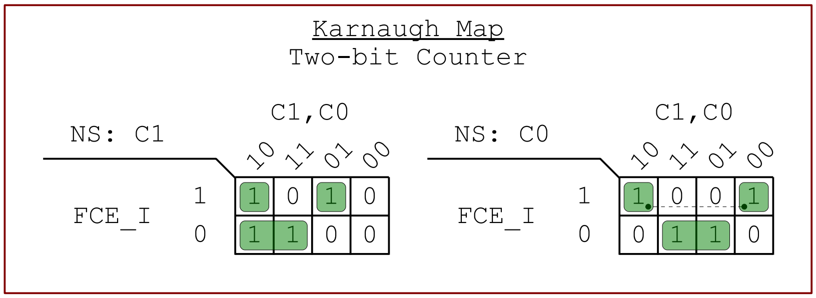
Figure 19 shows the correct use of system reset [RST] with a faux clock enable input[22]. It shows a two-bit binary counter whose state advances at the rate of a faux clock enable input signal [FCE_I].



```
STATES: C1,C0
INPUTS: FCE_I

C1:  Counter bit 1
C0:  Counter bit 0
FCE: Faux clock enable
RST: System reset
```

Truth Table
Two-bit Counter

```
F
CCC CC
E10 10
--- --
100 01
101 10
111 00
110 11

010 10
011 11
001 01
000 00
```



Karnaugh Map
Two-bit Counter

```
                  C1,C0                          C1,C0
NS: C1         10 11 01 00      NS: C0        10 11 01 00
            1 | 1| 0| 1| 0|                1 | 1| 0| 0| 1|
FCE_I       0 | 1| 1| 0| 0|     FCE_I      0 | 0| 1| 1| 0|
```

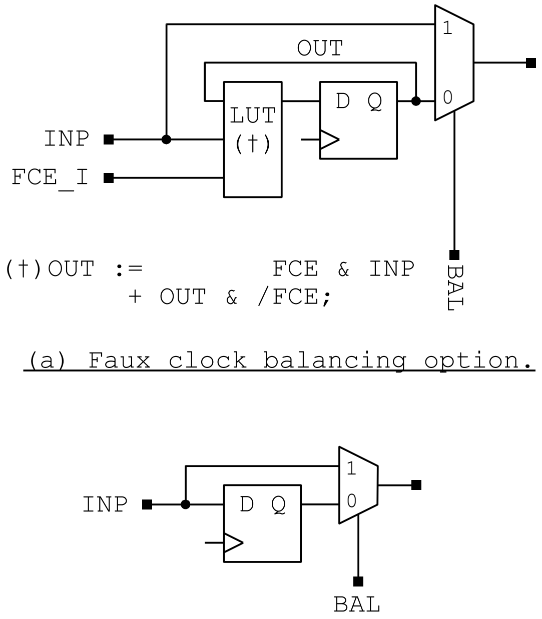
```
Logic Equations (Canonical Form):  LCCP (ILUT)
                                   IN L1 L2 L3
C1 :=  C1 & /C0 &  FCE_I & /RST_I     +1
    + /C1 &  C0 &  FCE_I & /RST_I
    +  C1       & /FCE_I & /RST_I;

C0 :=       /C0 &  FCE_I & /RST_I     +1
    +        C0 & /FCE_I & /RST_I;
```

Figure 19. Example showing correct use of system reset [RST] with a faux clock enable input.

[22] Note that the faux clock and reset inputs are not explicitly shown in the state diagrams (to improve their clarity and simplicity), but are present in the logic equations.

Faux Clock Segment Balancing Option

Figure 20 shows the general form for the faux clock segment balancing option[23]. Note that if the faux clock enable input [FCE_I] is disabled by tying it high ('1' or 'H') at synthesis time, then the balancing option minimizes to the standard balancing option.



(a) Faux clock balancing option.

(b) Standard balancing option.

Figure 20. General form of the faux clock segment balancing option.

Not For Control of the WISHBONE Interconnection

The faux clock is not recommended for WISHBONE MASTER or SLAVE interfaces. While in theory this is possible to do, in practice it can produce awkward problems and poor performance. In this situation the faux clock would effectively produce a multiphase clock, a concept which was rejected early in the preparation of WISHBONE Rev. A.0 (1999)[24]. Thus, there is no clock enable input defined for these interfaces in the WISHBONE specifications, and the only recommended way to synchronize the SoC is through the use of a single low-skew clock distribution.

[23] ***segment balancing option***
● An option used for the purpose of balancing the segments in an IP core, an interconnection or other SoC component. It is typically used for dividing a register-transfer logic (RTL) region into two or more convenient segments, each of which has a longest common combinatorial path (LCCP) that is shorter than the undivided case. In general, the option should not alter the overall form, fit or function of the logic; only the clock efficiency and LCCP metrics are expected to change. ■ *Ex.* A *segment balancing option* often uses a segment unpacking register whose presence is determined by a minimization exclusion.

[24] During the preparation of WISHBONE Rev. A.0 (1999), a multiphase clock was viewed as a very flexible, traditional clocking scheme. However, the disadvantages were: a non-portable design element (at the time many FPGA had only one common, low-skew clock distribution network available); greater system complexity in terms of available resources and timing specification (especially when inexperienced system integrators used economical tool chains for floor planning and routing). There was a perceived (but unproven) disadvantage with respect to power density.

Synchronized Non-unity Trans-domain SYSCON

The synchronized non-unity trans-domain SYSCON[25] synchronizes two clock domains where a primary-to-secondary clock relationship equal to or greater than 1:2 exists. When this type of system controller is used, then a synchronized non-unity trans-domain BRIDGE is used.

Clock Frequency and Duty Cycle Considerations

The synchronized non-unity trans-domain SYSCON uses a counter to convert a periodic clock signal [CLKP] in the primary domain to a periodic clock signal [CLKS] in the secondary domain. These two signals will always conform to the following relationship:

$$FREQ_{CLKS} = \frac{FREQ_{CLKP}}{Freqdiv}$$

$$\text{where: } 2 \leq Freqdiv < \infty$$

where '$\mathrm{FREQ}_{\mathrm{CLKP}}$' is the clock frequency of the primary domain; '$\mathrm{FREQ}_{\mathrm{CLKS}}$' is the clock frequency of the secondary domain; and 'Freqdiv' is a frequency divisor ratio.

One ramification of this relationship is that that the secondary clock can have a duty cycle outside of the 60/40 limits imposed by earlier versions of the WISHBONE specification[26]. If '*Freqdiv*' were limited only to even numbered integers, then the secondary clock would always have a nominal 50/50 (50%) duty cycle.

During the preparation of WISHBONE Rev. A.0 (1999) a single-phase clock with tight (50/50 or 50%) duty cycle control was considered, but then abandoned in favor of wider tolerances. The main disadvantage to tight 50/50 (50%) duty cycle control is that it is notoriously hard to maintain across a large clock distribution network, within a realistic operating envelope of voltage and temperature. Because many logic families do not use balanced output drivers, clock networks can sometimes produce (at their electrical extremities) something more like a 60/40 or 70/30 duty cycle, even when excited by a perfect (divide-by-two) 50/50 clock. Furthermore some target devices do not allow both positive and negative clock edges to be used for RTL transfer purposes, and some do not specify clock distortion or edge-to-edge clock jitter. Both of these can be non-portable design elements in FPGA.

Instead, a single-phase clock with wide tolerance duty cycle was selected for WISHBONE Rev. A.0 (1999). There, the term *wide tolerance* was not meant to tie down the duty cycle so much as it was to

[25] ***SYSCON***
● A WISHBONE [functional] module that drives the system clock [CLK_O] and reset [RST_O] signals. ● Acronym for: SYStem CONtroller.

[26] ***WISHBONE Rev. B.2 (2001+), Chapter 4 – Timing Specification***
"RULE 4.10 - The clock input [CLK_I] MUST have a duty cycle that is no less than 40%, and no greater than 60%." (p. 70)

"PERMISSION 4.10 - The SYSCON module MAY use a variable clock generator. In these cases the clock frequency can be changed by the SYSCON module so long as the clock edges remain clean and monotonic, and if the clock does not violate the duty cycle requirements." (p. 70) corner the designer into using only one clock edge[27]. Furthermore, all RTL registers had to be switched on the rising edge of the clock distribution[28]. This was a reasonable decision given that in 1999 all of the evaluated target devices supported rising edge clock polarities, and only a limited number of FPGA supported falling edges.

This original strategy is reasonable for the primary clock domain [CLKP]. After all, the primary clock domain might be driven directly from an off-chip oscillator with a distribution network which runs across a printed wiring board. Another reasonable assumption is that the frequency of the fastest clock domain in a SoC would be at or near the maximum rated speed of the target device. Such speed ratings are assumed to take into account the (unspecified) clock distortion and edge-to-edge clock jitter mentioned above, and are likely to be most reliable at a balanced 50/50 (50%) duty cycle.

It is equally reasonable to assume that the secondary clock signal [CLKS] needn't be as restricted as the primary clock signal [CLKP]. From the clock frequency relationship defined above, we know that the maximum secondary clock speed – at a frequency divisor (Freqdiv) of two (2) – will always be one-half the frequency of the primary clock domain. This lower frequency is presumed to relax the effects of the (unspecified) maximum clock distortion and jitter mentioned above because the target device manufacturer must assume that these are applied to a clock operating at the maximum rated speed. Furthermore, as demonstrated in Table 1, the maximum duty cycle distortion occurs at a frequency divisor ratio of 3:1, where it produces a 67/33 (67%) clock. Since all registers in the secondary clock domain only respond to the rising clock edge, there are no perceived side effects of using an unbalanced clock in this situation.

Table 1. Nominal duty cycle vs. *Freqdiv*.

Freqdiv	Nominal Secondary Duty Cycle	Clocks High	Clocks Low
2:1	50/50 (50%)	1	1
3 :1(†)	67/33 (67%)	2	1
4:1	50/50 (50%)	2	2
5:1	60/40 (60%)	3	2
6:1	50/50 (50%)	3	3
7:1	57/43 (57%)	4	3
8:1	50/50 (50%)	4	4

(†) Maximum duty cycle distortion.

[27] The duty cycle limitations were specified according to what was (and still is) available on most clock oscillators, and what was (and still is) common practice on microcomputer bus boards: 60/40 – 40/60. Most commercial, digital (square wave) clock oscillators are capable of producing and distributing a 60/40 clock to all of the chips on a printed wiring board, so this seemed to be a prudent choice.

[28] ***WISHBONE Rev. B.2 (2001+), 2.2 WISHBONE Signal Description***
"The clock input [CLK_I] coordinates all activities for the internal logic within the WISHBONE interconnect. All WISHBONE output signals are registered at the rising edge of [CLK_I]. All WISHBONE input signals are stable before the rising edge of [CLK_I]." (p. 34)

Another consideration is the use of a gated clock. A gated clock is one that can be stopped and restarted. In WISHBONE, an optional gated clock generator[29] allows the SYSCON to stop its clock output signal [CLK_O] in its low state. This technique has several benefits, such as the reduction of dynamic power or for observing I/O pins during test and evaluation. Starting and stopping the clock in this way obviously results in large duty cycle distortion.

When used for test and evaluation purposes, the gated clock generator can be further adapted to stop the clock in the second domain after a unit step pulse or an N-step pulse train. This feature is called a *clock controller.* Because proper clock pulse formation can only be achieved with a clock frequency divisor of at least 2:1, this function can only be achieved at a synchronized non-unity trans-domain boundary.

<u>Clock Frequency and the Longest Common Combinatorial Path (LCCP)</u>

The maximum operating frequency of any clock domain is directly proportional to the longest common combinatorial path (LCCP) within it. The LCCP is determined for portable designs – those where the target device has not been selected – using *The FASM Relative Timing (ILUT) Model.* This conceptual model is used for both relative and absolute timing estimates, and are considered only to be first-order approximations.

Relative timing estimates can be made by estimating the FASM toggle frequency of a hypothetical target device. In this case $\mathrm{LCCP_{REL}}$ is expressed in dimensionless unit of measure called the 'ILUT' (or 'interconnection plus LUT')[30], which represents the relative propagation delay along the longest common combinatorial path. The maximum operating frequency is then found from the relationship:

$$Fclk(max) = \frac{FASM\ Toggle\ Frequency}{LCCP_{REL}}$$

where and 'Fclk(max)' is the estimated maximum operating frequency of a clock domain expressed in units of Hz, KHz or MHz; 'FASM toggle frequency' is the estimated operating frequency of a clock domain where $\mathrm{LCCP_{REL}}$ = + 1 ILUT expressed in units of Hz, KHz or MHz; and '$\mathrm{LCCP_{REL}}$' is the relative longest common combinatorial path of the clock domain expressed in dimensionless units of ILUT. For example, if the FASM toggle frequency is estimated to be 400 MHz, then a clock domain with an $\mathrm{LCCP_{REL}}$ = +8 ILUT would result in a maximum operating frequency of about 50 MHz.

Absolute timing estimates can also be made by equating the same inputs to timing specifications for a particular target device or router run. The maximum operating frequency is then found according to the relationship:

$$Fclk(max) = \frac{1}{LCCP_{\mathrm{ABS}}} = \frac{1}{Tpd(DOM), clk-su}$$

[29] ***<u>WISHBONE Rev. B.2 (2001+), Chapter 4 – Timing Specification</u>***
"PERMISSION 4.15 - The SYSCON module MAY use a gated clock generator. In these cases the clock shall be stopped in the low logic state. When the gated clock is stopped and started the clock edges are required to remain clean and monotonic." (p. 71)

[30] An ILUT represents a typical RTL segment which has a 5-input look-up table (LUT) driving a D-type flip-flop.

where and 'Fclk(max)' is the estimated maximum operating frequency of a clock domain expressed in units of Hz, KHz or MHz; '$LCCP_{ABS}$' is the absolute longest common combinatorial path of the clock domain expressed in units of seconds or nanoseconds (it can be expressed alternatively as 'Tpd(DOM), clk-su). For example, if the longest $LCCP_{ABS}$ of any RTL segment on a routed target device is Tpd(DOM), clk-su = 100 nS, then Fclk(max) = 10 MHz.

Note that the relative and absolute propagation times represented by the 'ILUT' and 'Tpd(), clk-su' are proportional to each other: +1 ILUT ~ (1 x Tpd(DOM), clk-su); +2 ILUT ~ (2 x Tpd(DOM) clk-su); and +N ILUT ~ (N x Tpd(DOM) clk-su), where 'N' is the number of LUT levels present within the RTL segment.

These models are typically used to understand the relative timing relationship between the primary and secondary clock domains in portable systems. As an example, Figure 21 shows the timing relationship between clock period and $LCCP_{REL}$ for a 3:1 frequency divisor ratio. Since portable designs are not associated with any particular target device, unless otherwise indicated it is useful to assume that the primary clock domain has a $LCCP_{REL}$ = +1 ILUT. Then, the secondary clock period can be quickly found from the frequency divisor ratio. The example uses a divide-by-three counter to generate the secondary clock, so its primary secondary estimate is $LCCP_{REL}$ = +3 ILUT.

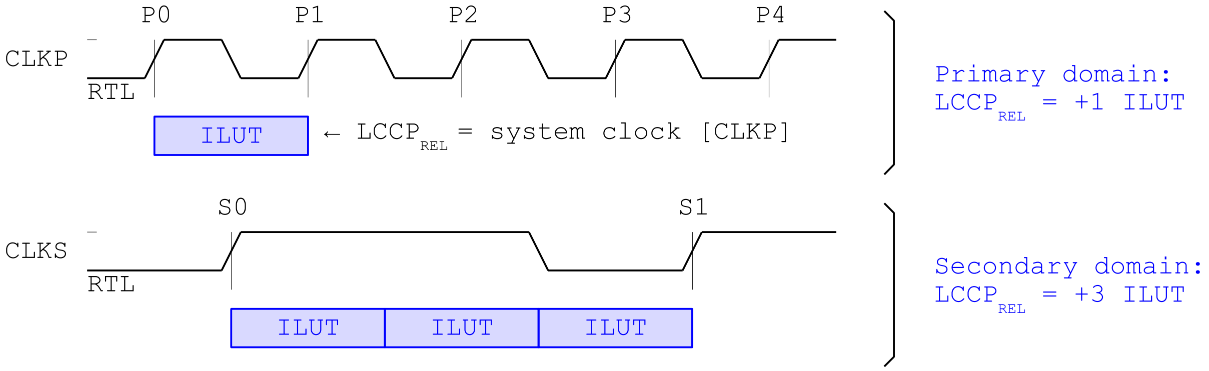


Figure 21. Relationship between clock frequency and LCCP for a frequency divisor ratio of 3:1.

This idea can be expanded for multiple clock domains in the system. Figure 22 shows the relationship between $LCCP_{REL}$ and clock domain frequencies for a hypothetical SoC. There, a vertical center bar shows the following three clock domain regions:

- *Region 0 - Clock Controller*. The highest frequency domain; operates from an external clock source. This domain is only used for a clock controller for test and evaluation that provides unit step pulse and N-step pulse trains for the two synchronized domains which operate below it. $LCCP_{REL}$ = +1 ILUT.

- *Region 1 - Symmetric Processing*[31]. A clock domain driven by a synchronized non-unity trans-domain SYSCON with a 2:1 frequency divisor ratio. This domain provides an environment

[31] **symmetric parallelism; symmetric processor; symmetric processing**
● Two or more concurrent, value-added data processes with a data dependency. ● A value-added data process which is unified by a chain of data dependency. ● A pipeline. ■ *Ex.* An instruction pipeline operating on a particular instruction thread is a form of *symmetric parallelism*.

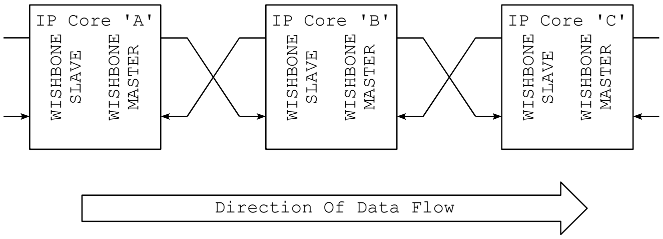
where IP cores tend to perform symmetric processing, and share data over a WISHBONE data flow[32] interconnection. It is optimized for IP cores having RTL segments with LCCP_{REL} = +2 ILUT.

- Region 2 - Asymmetric Processing[33]. A clock domain driven by a synchronized non-unity trans-domain SYSCON with a 4:1 frequency divisor ratio. Since the clock is derived from the domain directly above it, it operates at an 8:1 frequency divisor ratio with respect to the highest frequency domain. This domain provides an environment where IP cores tend to perform asymmetric processing, and share data over a WISHBONE shared bus[34] interconnection. It is optimized for IP cores having RTL segments with LCCP_{REL} = +8 ILUT, such as a typical WISHBONE standard shared bus interconnection having up to four (4) MASTER and four (4) SLAVE interfaces.

A target device for the SoC shown in the diagram must support the three clock domain regions by having at least three low-skew clock distribution networks. The column on the left side of the diagram shows how the clock signal for each domain region is produced. It assumes that Region 0 runs at the highest clock frequency, and is obtained from an external source such as a crystal clock oscillator or a phase-lock loop. The Region 0 clock is used as the excitation signal for Region 1, and the Region 1 clock is then used as the excitation signal for Region 2.

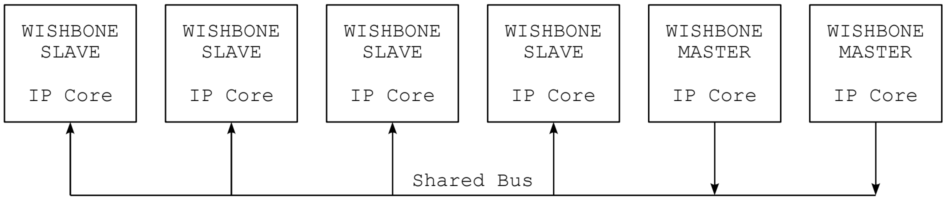
Even though a target device has not been selected for this system, we do known that these three domains are synchronized with each other. This means we can make some general observations about the relative clock frequencies. This is shown in the right hand column under the heading 'Potential

[32] ***data flow interconnection***
An interconnection where data flows through a prearranged set of IP cores in a sequential order. Data flow architectures often have the advantage of parallelism, whereby two or more functions are executed at the same time.

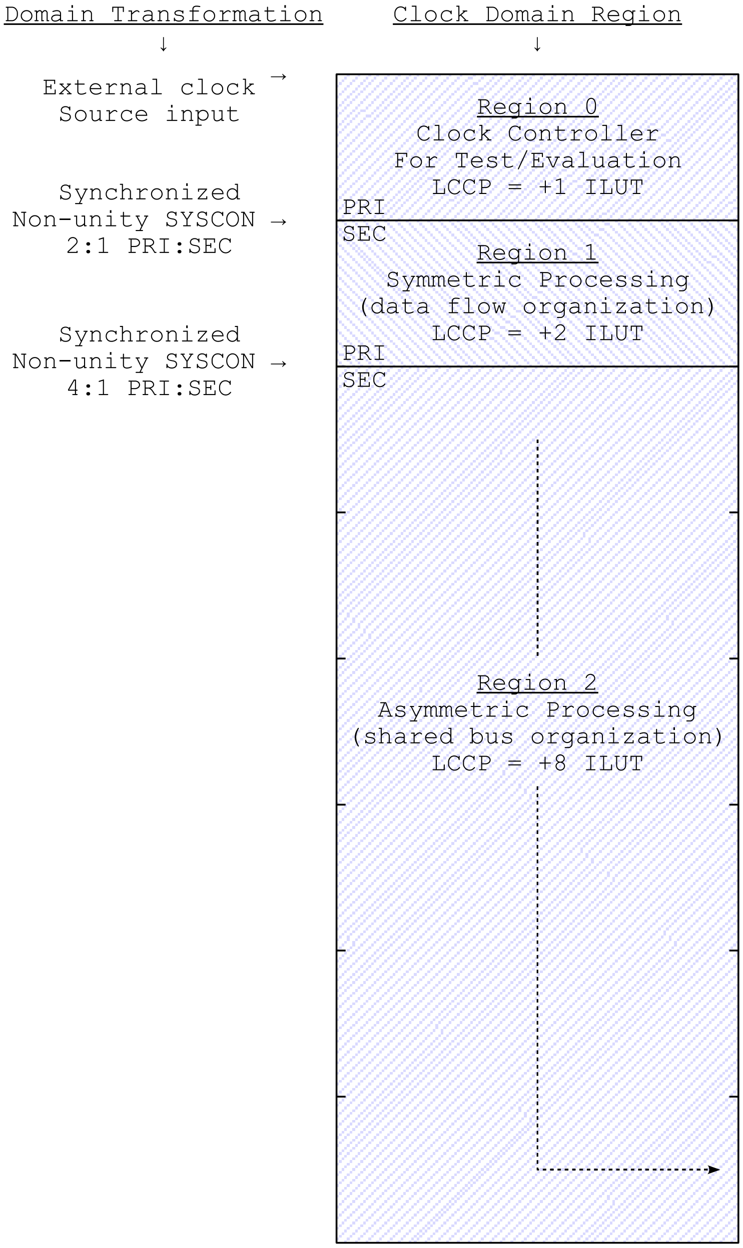


[33] ***asymmetric parallelism; asymmetric processor; asymmetric processing***
● Two or more concurrent, value-added data processes that do not have a data dependency. ■ *Ex.* A dual-core microprocessor running two different jobs at the same time, where the two jobs do not share any data between them, is a form of *asymmetric parallelism*.

[34] ***shared bus interconnection***
The shared bus interconnection is a system where a MASTER initiates addressable bus cycles to a target SLAVE. Traditional buses such as VMEbus and PCI bus use this type of interconnection. As a consequence of this architecture, only one MASTER at a time can use the interconnection resource (i.e. bus).



Clock Domain Frequency (MHz)'. For example, if Region 0 were operated at a clock speed of 400 MHz, then Region 1 would be at 200 MHz, and Region 2 would be at 50 MHz. Three starting cases for Region 0 are presented at 10, 100 and 400 MHz.



Region $LCCP_{REL}$	Potential Clock Domain Frequency (MHz)		
+1 ILUT	(†) 10.00	(†) 100.00	(†) 400.00
+2 ILUT	5.00	50.00	200.00
+3 ILUT	3.33	33.33	133.33
+4 ILUT	2.50	25.00	100.00
+5 ILUT	2.00	20.00	80.00
+6 ILUT	1.67	16.67	66.67
+7 ILUT	1.43	14.29	57.14
+8 ILUT	1.25	12.50	50.00

(†) FASM Toggle Frequency.

Figure 22. Relationship between LCCP and potential clock domain frequencies.

Uncertainty of the Relative LCCP and the ILUT

Because these timing estimates are made on unspecified target devices, there is always some uncertainty associated with $LCCP_{REL}$. Because of routing methodologies, a design that conforms to $LCCP_{REL}$ = +N ILUT is assumed to simultaneously take on the entire range of possible propagation delays over the operating envelope, where:

$$\text{Infinitesimally small value} < LCCP_{REL} \leq +N \text{ ILUT}$$

To understand this concept, consider the routing of multiple interconnections within an RTL segment. In some cases the interconnections will be very short because there is little or no combinatorial logic or very little routing path that the signal must travel through. Thus, some interconnections might have a very small but non-zero delay, such as when the source register in an RTL segment is located next to the destination register. In other cases the source and destination registers will be located much farther apart with more logic and routing between them. In these cases the interconnection might take on the largest possible delay. There is no practical upper limit to this delay, except as enforced by the tool chain (router)[35]. However, within each RTL segment some paths will be short (at the non-zero lower limit), and some will be long (at the upper limit). This means that an interconnection within a RTL segment might take on any propagation delay within the range of possible delays.

Synchronized Trans-domain Clock Phase Lag 'Tpd(TDP), in-out'

The synchronized trans-domain clock phase lag shown in Figure 23 is a time delay introduced between synchronized primary and secondary clock domains. This delay occurs whenever a physical device or a physical phenomenon along the clocking path at a synchronized trans-domain boundary introduces a delay which exceeds the minimum allowed for a low-skew clock distribution network. This value is always added to or subtracted from the LCCP of signals as they move across the trans-domain boundary. When expressed as relative time: $LCCP_{REL} = +1$ ILUT, where 'ILUT' is a unit of measure established by the FASM relative timing (ILUT) model. Units of absolute time: $LCCP_{ABS} \leq$ Tpd(TDP) in-out (max) having units of measure expressed as time, such as in seconds or nanoseconds. The synchronized trans-domain clock phase lag 'Tpd(TDP), in-out' always occurs at a synchronized trans-domain clock boundary.

Clock Phases Identify Signal Entry and Exit Points

The synchronized non-unity trans-domain SYSCON produces the secondary clock as well as two faux clock phases. All are synchronized to the primary clock[36]. These two faux clock phases are used to identify edges on the primary clock [CLKP] where signals may enter and exit the domain. They are defined as follows:

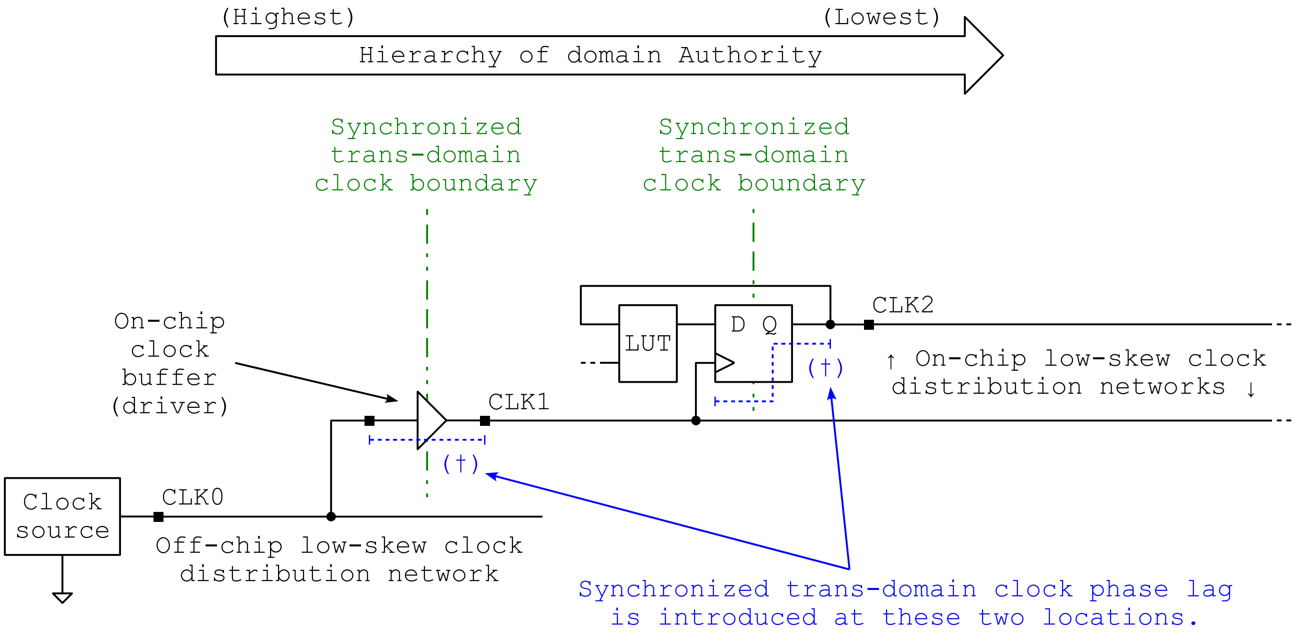
- *Synchronized trans-domain entry phase [ENTP]*. The point in a multiphase timing protocol where a signal synchronized to the secondary clock domain may enter the primary clock domain. This point may may exist at more than one location within the timing protocol, but the specification limits the choices in order to harmonize clock generator designs.

- *Synchronized trans-domain exit phase [EXTP]*. The point in a multiphase timing protocol

[35] In general, a tool chain will only consider the upper limit of a timing parameter.

[36] This is the main difference between the synchronized unity and non-unity SYSCONs. The unity system controller is intended only for 1:1 frequency divisor ratios, and synchronizes a first faux clock phase to the primary clock [CLKP], and a second faux clock phase to the secondary clock [CLKS].

where a signal in the primary clock domain may be synchronized to the secondary clock domain. Because of the synchronized trans-domain clock phase lag, and because the exit transfer delay should be as close as possible to the destination transfer delay, this point exits at only one location within the timing protocol.


(Highest)
(Lowest)
Hierarchy of domain Authority
Synchronized trans-domain clock boundary
Synchronized trans-domain clock boundary
On-chip clock buffer (driver)
LUT
D Q
CLK2
(†)
↑ On-chip low-skew clock distribution networks ↓
CLK1
(†)
Clock source
CLK0
Off-chip low-skew clock distribution network
Synchronized trans-domain clock phase lag is introduced at these two locations.

(†) In a hierarchy of three synchronized clock domains, each clock ([CLK0], [CLK1] and [CLK2]) are each carried by a low-skew clock distribution network. A synchronized trans-domain clock phase lag is introduced by any combination of clock buffer, flip-flop, asynchronous look-up table (LUT), physical Phenomenon or some other entity resides between the low-skew clock distribution networks. Under the FASM Relative Timing (ILUT) Model, this lag (delay) is less-than or equal to one (+1) ILUT.

Figure 23. Synchronized trans-domain clock phase lag.

The purpose of the entry and exit phases is to identify specific clock edges on the primary clock [CLKP] when signals may enter and exit the primary domain. This allows the creation of standard timing parameters for the RTL segments bounded by registers on both sides of the trans-domain boundary.

Figure 24 shows these entry and exit segments. There, each segment has a unique transfer delay according to the direction of signal travel. When a signal (shown in RED) enters the primary domain, the propagation delay across the boundary is constrained by the entry delay, which is 'Tpd(STE), clk-su'. The clock edges when these signals are recognized are indicated by the synchronized trans-domain entry phase signal [ENTP].

When a signal (shown in BLUE) exits the primary domain, the propagation delay across the boundary is constrained by the exit delay 'Tpd(STX), clk-su'. The clock edges when these signals are generated are indicated by the synchronized trans-domain exit phase signal [EXTP].

The entry and exit transfer delays which are associated with the entry and exit points are defined as follows:

- *Synchronized trans-domain entry transfer delay 'Tpd(STE), clk-su'*. The timing parameter that defines the longest common combinatorial path (LCCP) which is allotted for a control transfer signal entering the primary clock domain from the secondary clock domain. It is measured between identified rising edges of the primary and secondary clocks. The RTL transfer region along this path is limited to one (1) look-up table.

  When generated by a synchronized non-unity trans-domain SYSCON, the timing parameter is measured between any secondary clock edge [CLKS] and the next primary clock edge [CLKP] which is identified by the primary entry phase signal [ENTP]. When expressed as relative time: $LCCP_{REL}$ = N x ILUT, where 'N' is any integer between one and infinity (∞) and is equal to the value of the primary clock domain, and 'ILUT' is a unit of measure established by the FASM relative timing (ILUT) model. Units of absolute time: $LCCP_{ABS} \leq$ Tpd(STE), clk-su (max0 having units of measure expressed as time, such as in seconds or nanoseconds.

- *Synchronized trans-domain exit transfer delay 'Tpd(STX), clk-su'*. The timing parameter that defines the longest common combinatorial path (LCCP) which is allotted for a control transfer signal exiting the primary clock domain to the secondary clock domain. It is measured between identified rising edges of the primary and secondary clocks. The RTL transfer region along this path is limited to one (1) look-up table.

  When using a synchronized non-unity trans-domain SYSCON, the timing parameter is measured between a primary clock edge [CLKP] which is identified by the primary entry phase signal [ENTP], and the next secondary clock edge [CLKS]. When expressed as relative time: $LCCP_{REL}$ = {(N x ILUT) - 1}, where 'N' is any integer between one and infinity (∞) and is equal to the value of the secondary clock domain, and 'ILUT' is a unit of measure established by the FASM relative timing (ILUT) model. Units of absolute time: $LCCP_{ABS} \leq$ Tpd(STX), clk-su (max) having units of measure expressed as time, such as in seconds or nanoseconds.

Note that the synchronized trans-domain entry transfer delay 'Tpd(STE), clk-su' for the non-unity SYSCON is always equal to the primary clock domain transfer delay 'Tpd(PRI), clk-su'. However, the exit delay does not equal that of the secondary domain 'Tpd(SEC), clk-su'. This lack of symmetry has two causes: the synchronized trans-domain clock phase lag (noted above), and clock edge predictability (which comes about because there are always two or more primary clock edges for every secondary edge).

The Multiphase Clock Generator

The multiphase clock generator excitation is provided by the primary clock signal [CLKP]. It produces the secondary clock signal [CLKS] as well as the two faux clock phases [ENTP] and [EXTP].

Figure 25 shows all of the possible entry and exit phases of the synchronized non-unity trans-domain SYSCON at clock divider ratios between 2:1 and 5:1. All possible clock phases are shown for informational purposes; these are identified as faux clock enable [FCE(4..0)]. However, only two need to be produced by the SYSCON (i.e. the others are redundant, and may be omitted). One additional clock phase is available for every increment in divisor ratio. Thus, a 2:1 ratio produces two clock phases; a 3:1 ratio produces three clock phases and so forth. However, by convention only two of the clock phases are used to identify when signals may enter or exit the primary domain, which are shown in RED and BLUE respectively.

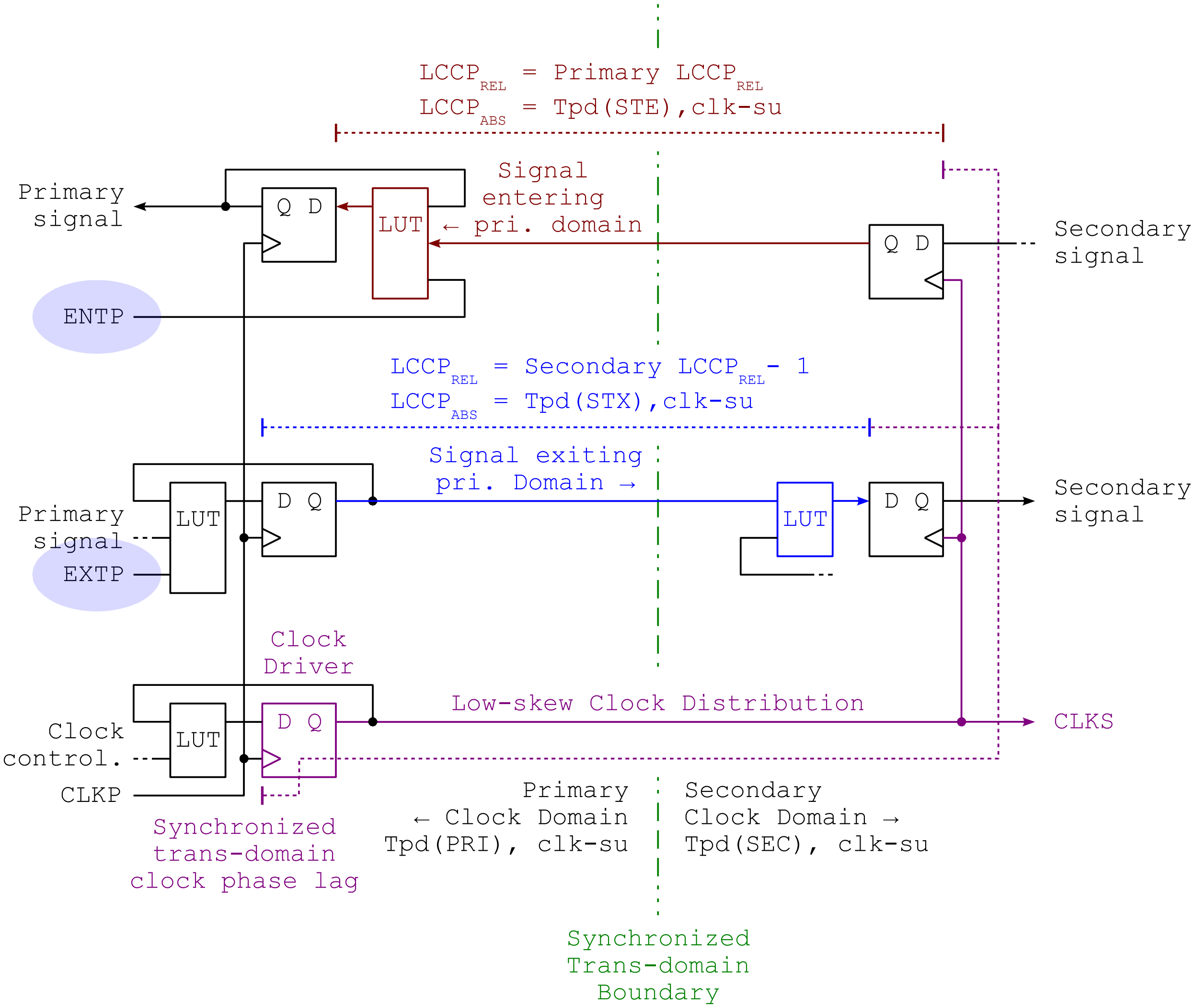
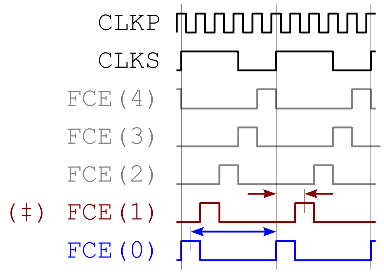

LCCP_REL = Primary LCCP_REL
LCCP_ABS = Tpd(STE),clk-su
Primary signal
Q D
LUT
Signal entering ← pri. domain
Q D
Secondary signal
ENTP
LCCP_REL = Secondary LCCP_REL - 1
LCCP_ABS = Tpd(STX),clk-su
Signal exiting pri. Domain →
Primary signal
LUT
D Q
EXTP
LUT
D Q
Secondary signal
Clock Driver
Low-skew Clock Distribution
Clock control.
LUT
D Q
CLKP
CLKS
Synchronized trans-domain clock phase lag
Primary ← Clock Domain Tpd(PRI), clk-su
Secondary Clock Domain → Tpd(SEC), clk-su
Synchronized Trans-domain Boundary

Figure 24. Synchronized non-unity trans-domain SYSCON entry and exit segments.

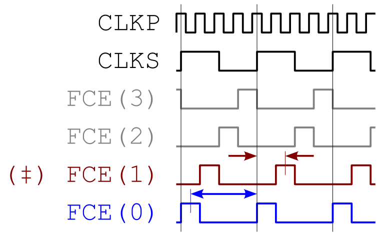
The entry and exit transfer delays 'Tpd(STE), clk-su' and 'Tpd(STX), clk-su' take into account the uncertainty of the trans-domain LCCP. As an example, Figure 26 shows the uncertainty associated with the entry phase signal [ENTP]. The exit phase signal [EXTP] is not shown, but is handled similarly. This is a graphical analysis of how LCCP uncertainty is factored into the entry phase timing parameter 'Tpd(STE), clk-su'. In this case a secondary signal which enters the primary clock domain is first registered at the rising edge of secondary clock [CLKS]. It then crosses the trans-domain boundary and is then registered at the primary clock edge which is indicated by the entry phase signal [ENTP]. The uncertainty also factors in the synchronized trans-domain clock phase lag.

The diagram also shows how phase [FCE(0)] (shown in gray) fails to meet the requirements for the entry phase. Note that phase [FCE(0)] is does meet the requirements for the exit phase (not shown).

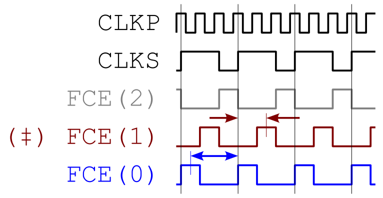
5:1	Freq. Divisor	Duty Cycle	Symetr Output	Secondary LCCP$_{REL}$(†) Tot. LCCP$_{REL}$(Phases)
	5:1	3/2 (60%)	NO	+5 ILUT (1-1-1-1-1)



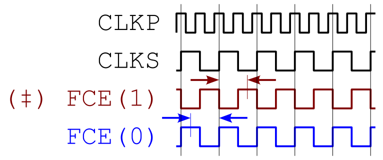
4:1	Freq. Divisor	Duty Cycle	Symetr Output	Secondary LCCP$_{REL}$(†) Tot. LCCP$_{REL}$(Phases)
	4:1	2/2 (50%)	YES	+4 ILUT (1-1-1-1)



3:1	Freq. Divisor	Duty Cycle	Symetr Output	Secondary LCCP$_{REL}$(†) Tot. LCCP$_{REL}$(Phases)
	3:1	2/1 (67%)	NO	+3 ILUT (1-1-1)



2:1	Freq. Divisor	Duty Cycle	Symetr Output	Secondary LCCP$_{REL}$(†) Tot. LCCP$_{REL}$(Phases)
	2:1	1/1 (50%)	YES	+2 ILUT (1-1)



(†) Assumes that primary clock domain LCCP$_{REL}$ = +1 ILUT.

(‡) FCE(1): Synchronous trans-domain entry phase LCCP$_{REL}$ = PRI LCCP$_{REL}$ (ILUT)
LCCP$_{ABS}$ = Tpd(STE), clk-su (nS)

FCE(0): Synchronous trans-domain exit phase LCCP$_{REL}$ = SEC LCCP$_{REL}$ - 1 (ILUT)
LCCP$_{ABS}$ = Tpd(STX), clk-su (nS)

Figure 25. Entry and exit phases at various clock divider ratios.

Secondary and Global Reset Monitor Signals

Because a hierarchy of domain authority exists at all trans-domain boundaries, a system reset is asserted and negated in the primary domain before the secondary domain. This is problematic if a remote MASTER in the primary domain attempts to access a remote SLAVE in the secondary domain before initialization has been completed. To overcome this problem, the state of the secondary system reset logic is mirrored back to the primary side of the domain boundary. This is called a secondary reset monitor signal, and it can be used to prevent the primary domain from accessing the secondary domain before both are ready.

The secondary reset monitor output signal [SRMP_O] can be daisy-chained with multiple clock domains residing below it. This chain is tied in with a global reset monitor signal, and it can be used to prevent any remote master in any clock domain from accessing any other domain until all have been initialized. This chain notifies all upstream domains when all of the downstream domains are ready for operation. Thus the monitor in the first and second domains of a three (3) domain system are notified only after all three domains are ready.

The necessity to use the secondary or global reset monitor signal depends on the application. It is not used in all situations.

The secondary and global reset monitor signals follow the rules for reset blocking. Thus, the primary reset signal [RSTP] must be at least two clock pulses wide to insures that both the primary reset signal [RSTP] and the secondary reset monitor signal [RSTS] are both asserted at the same time sometime during the reset interval.

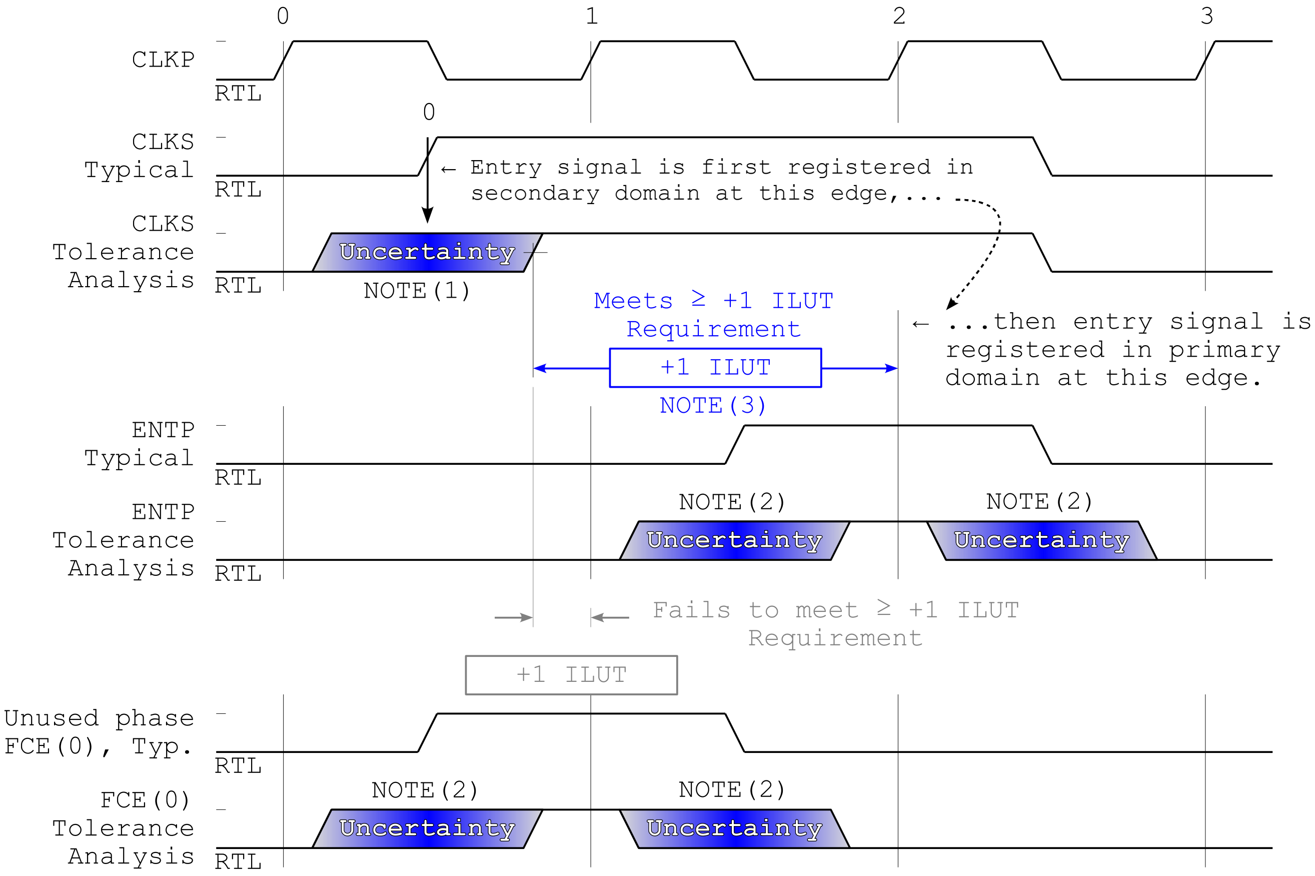
Example: Synchronized Non-unity Trans-domain SYSCON

Figure 27 shows an example of a synchronized non-unity trans-domain SYSCON[37]. This functional module demonstrates many of the principles described in the previous sections.

This section provides a CLASS Machine design representation for this type of SYSCON. There are no special portability requirements other than requirements for two low-skew clock distribution networks, and adherence to the timing parameter 'Tpd(TPD), in-out'. In the SYSCON example, closed-loop $LCCP_{REL}$ = +1 ILUT, leading balance $LCCP_{REL}$ = +1 ILUT, with no lagging balance. This makes it suitable for the highest clock domain in any system[38].

[37] The signal names listed in the external view of the diagram indicate the clock domain to which it is synchronized. Those synchronized to the primary clock domain are indicated by '...P_I' or '...P_O'; those synchronized to the secondary clock domain are indicated by '...S_I' or '...S_O'. This signal nomenclature is used throughout this report.

[38] The FASM toggle frequency, and $LCCP_{REL}$ at the highest level in the hierarchy of domain authority, are both +1 ILUT.



Notes:

(1) Synchronized trans-domain clock phase lag uncertainty: $0 < LCCP_{REL} \le +1$ ILUT.

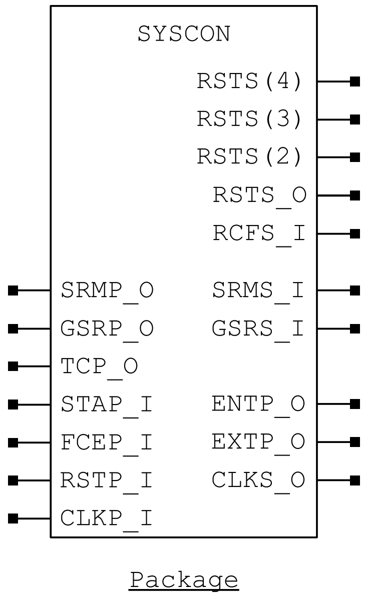
(2) Phase position uncertainty: $0 < LCCP_{REL} \le +1$ ILUT.

(3) Relative ILUT size: +1 ILUT = Uncertainty

Figure 26. Uncertainty associated with the entry phase signal [ENTP].

Signal Description

Signal	Dir	Description
CLKP_I	INP	System clock, primary clock domain
CLKS_O	OUT	System clock, secondary clock domain
ENTP_O	OUT	Trans-domain entry phase
EXTP_O	OUT	Trans-domain exit phase
FCEP_I	INP	Faux clock enable, primary [CLKP_I]
GSRS_I	INP	Global system reset
GSRP_O	OUT	Global system reset
RCFS_I	INP	Reset clock feedback
RSTS(4..2)	OUT	Reset clock feedback Connecting [RCFS_I] to [RSTS(2)] Results in a two (2) clock reset pulse, connecting to [RSTS(3)] results in a three (3) clock Pulse and so on. Signal output [RSTS_O] is equivalent to [RSTS(1)].
RSTP_I	INP	System reset, primary clock domain
RSTS_O	OUT	System reset, secondary clock domain
SRMS_I	INP	Secondary clock domain reset monitor
SRMP_O	OUT	Secondary clock domain reset monitor
STAP_I	INP	Start *'1': Counter start*. If tied high the counter is restarted automatically. If it is negated, then the counter stops at the next terminal count pulse. Thus, pulsing the start ignal [STA] for one primary clock cycle will initiate a single group. *'0': Counter off/stop*.
TCP_O	OUT	Terminal count



Key to Figures and Acronyms

BIT:	Shift counter bit	RUN:	Counter running
CLK:	System clock	SFT:	Shift/load
CNTL:	Shift counter control	SIN:	Serial in
DIN:	Data in	SOU:	Serial out
ENT:	Clock domain entry phase	SRM:	Secondary reset monitor
EXT:	Clock domain exit phase	STA:	Counter start
FCE:	Faux clock enable	TC:	Terminal count
GSR:	Global system reset		
PIN:	Parallel input	P ▷	Primary clock domain [CLKP_I]
PRM:	Primary reset monitor		
RCF:	Reset count feedback		
RST:	System reset		
		S ▷	Secondary clock domain [CLKS_I]

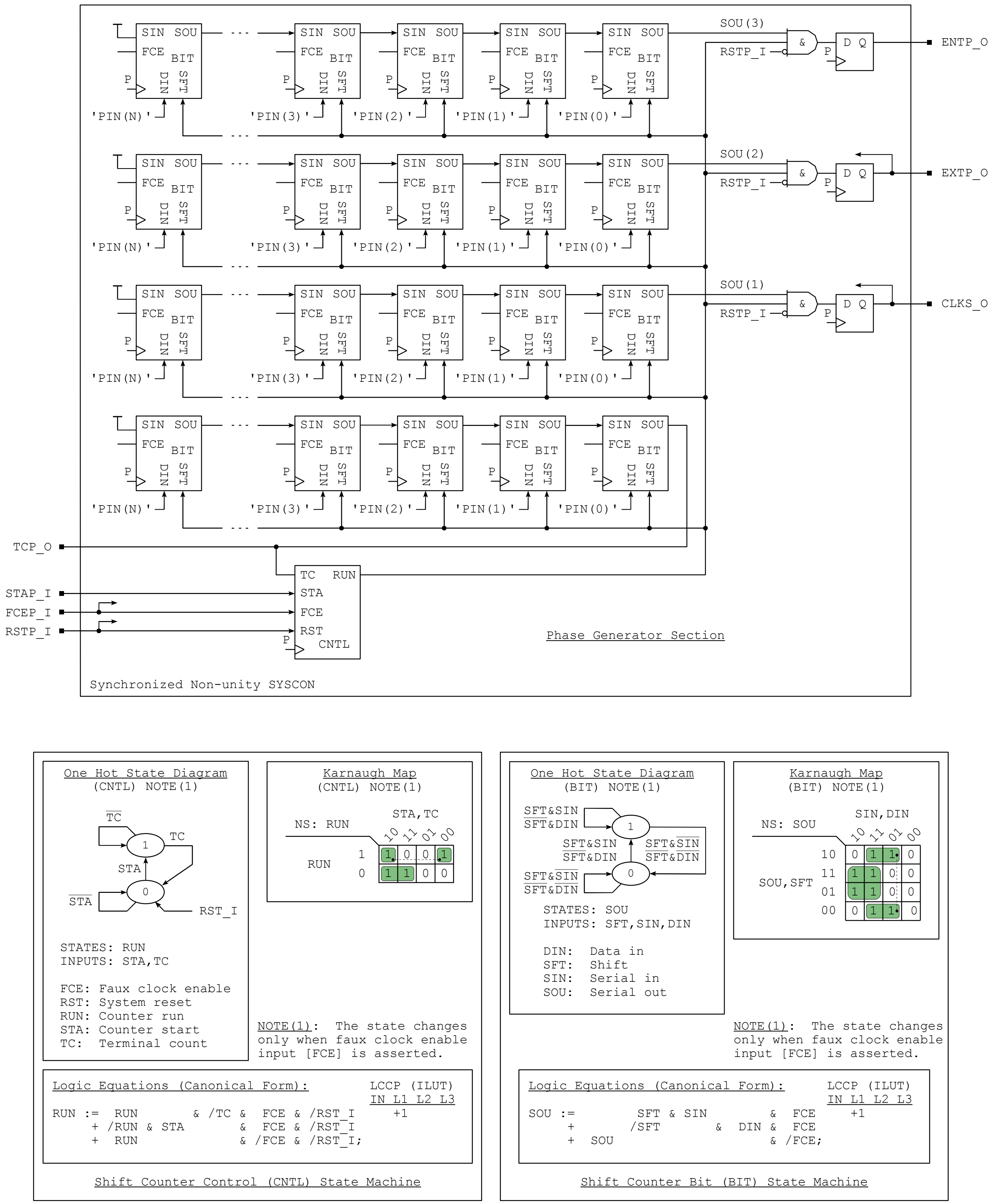
Figure 27. External view of the synchronized non-unity trans-domain SYSCON.

This SYSCON has three different operating modes. They are configured on the primary clock side by selecting a combination of start input signal [STAP_I] and faux clock enable input [FCEP_I], thusly:

- *Continuous mode*. The continuous mode is intended for normal run-time operation, and produces a steady stream of secondary clock pulses. It is selected by asserting both the clock start input signal [STAP_I] and faux enable [FCEP_I] for one clock cycle input [CLKP_I].
- *Single (unit) step mode*. This mode requires an external clock controller module. The single (unit) step mode is intended for evaluation and test purposes, and produces one positive and one negative edge on the secondary clock (i.e. a clock-by-clock basis). It is selected by asserting the start input [STAP_I], and advancing one state by asserting the faux clock enable input [FCEP_I].
- *Group step mode*. This mode requires an external clock controller module. The group step mode is intended for evaluation and test purposes, and advances the secondary clock by one group of clock [CLKS_O] plus entry [ENTP] and exit [EXTP] phases (i.e. on a group-by-group basis). It is selected by pulsing the start input signal [STAP_I] and asserting the clock enable input [FCEP_I] for one clock cycle input [CLKP_I].

Figure 28 is a detail showing the internal view of the phase generator section. There, a series of four shift registers produce four serial outputs which generate timing waveforms for terminal count pulse output signal [TCP_O], secondary clock [SOU(1)], exit phase [SOU(2)] and entry phase [SOU(3)]. The diagram illustrates the case where each shift register has a length of 'N' bits, where 'N+1' equals the divisor ratio of the generator. Thus, a generator providing a divisor ratio of 4:1 would only require four-bit shift registers.

At the beginning of each cycle group, each shift counter chain is initialized with a parallel input value [PIN()]. These are summarized in Figure 29 for frequency divisor ratios between 2:1 and 8:1. The parallel input value in combination with the shift counter produces a four channel timing generator.



STATES: RUN
INPUTS: STA,TC

FCE: Faux clock enable
RST: System reset
RUN: Counter run
STA: Counter start
TC: Terminal count

Karnaugh Map
(CNTL) NOTE(1)

NS: RUN

RUN \ STA,TC	10	11	01	00
1	1	0	0	1
0	1	1	0	0

NOTE(1): The state changes only when faux clock enable input [FCE] is asserted.

Logic Equations (Canonical Form):

```
                                             LCCP (ILUT)
                                             IN L1 L2 L3
RUN :=  RUN        & /TC &  FCE & /RST_I      +1
     + /RUN & STA        &  FCE & /RST_I
     +  RUN              & /FCE & /RST_I;
```

Shift Counter Control (CNTL) State Machine

STATES: SOU
INPUTS: SFT,SIN,DIN

DIN: Data in
SFT: Shift
SIN: Serial in
SOU: Serial out

Karnaugh Map
(BIT) NOTE(1)

NS: SOU

SOU,SFT \ SIN,DIN	10	11	01	00
10	0	1	1	0
11	1	1	0	0
01	1	1	0	0
00	0	1	1	0

NOTE(1): The state changes only when faux clock enable input [FCE] is asserted.

Logic Equations (Canonical Form):

```
                                            LCCP (ILUT)
                                            IN L1 L2 L3
SOU  :=        SFT & SIN         &  FCE      +1
     +        /SFT        &  DIN &  FCE
     +  SOU                      & /FCE;
```

Shift Counter Bit (BIT) State Machine

Figure 28. Phase generator for the synchronized non-unity trans-domain SYSCON.

```
              PIN()  Sec:   [CLK_O]
Cntr. Chain 76543210 Pri.  Duty Cycl.          LCCP_ABS
----------- -------- ----- ---------- -----------------------
SOU(3)/ENT: 00000010  8:1     4/4  (†)  Tpd(SEC) = 8 x Tpd(PRI)
SOU(2)/EXT: 00000001                    Tpd(STX) = 7 x Tpd(PRI)
SOU(1)/CLK: 00001111                    Tpd(STE) = 1 x Tpd(PRI)
SOU(0)/TC:  01000000

SOU(3)/ENT:  0000010  7:1     4/3       Tpd(SEC) = 7 x Tpd(PRI)
SOU(2)/EXT:  0000001                    Tpd(STX) = 6 x Tpd(PRI)
SOU(1)/CLK:  0001111                    Tpd(STE) = 1 x Tpd(PRI)
SOU(0)/TC:   0100000

SOU(3)/ENT:   000010  6:1     3/3  (†)  Tpd(SEC) = 6 x Tpd(PRI)
SOU(2)/EXT:   000001                    Tpd(STX) = 5 x Tpd(PRI)
SOU(1)/CLK:   000111                    Tpd(STE) = 1 x Tpd(PRI)
SOU(0)/TC:    010000

SOU(3)/ENT:    00010  5:1     3/2       Tpd(SEC) = 5 x Tpd(PRI)
SOU(2)/EXT:    00001                    Tpd(STX) = 4 x Tpd(PRI)
SOU(1)/CLK:    00111                    Tpd(STE) = 1 x Tpd(PRI)
SOU(0)/TC:     01000

SOU(3)/ENT:     0010  4:1     2/2  (†)  Tpd(SEC) = 4 x Tpd(PRI)
SOU(2)/EXT:     0001                    Tpd(STX) = 3 x Tpd(PRI)
SOU(1)/CLK:     0011                    Tpd(STE) = 1 x Tpd(PRI)
SOU(0)/TC:      0100

SOU(3)/ENT:      010  3:1     2/1       Tpd(SEC) = 3 x Tpd(PRI)
SOU(2)/EXT:      001                    Tpd(STX) = 2 x Tpd(PRI)
SOU(1)/CLK:      011                    Tpd(STE) = 1 x Tpd(PRI)
SOU(0)/TC:       010

SOU(3)/ENT:       10  2:1     1/1  (†)  Tpd(SEC) = 2 x Tpd(PRI)
SOU(2)/EXT:       01                    Tpd(STX) = 1 x Tpd(PRI)
SOU(1)/CLK:       01                    Tpd(STE) = 1 x Tpd(PRI)
SOU(0)/TC:        01

(†) [CLKS_O] is symmetrical (50% duty cycle).
```

Figure 29. Shift counter pre-load values for the synchronized non-unity trans-domain SYSCON.

A shift register is used for the phase generator because it provides a simple means for understanding and controlling the various output phases, and because it has an $LCCP_{REL}$ = +1 ILUT.

The shift counter control (CNTL) state machine responds to the start [STAP_I], faux clock enable [FCEP_I] and primary reset [RSTP_I] inputs. It is responsible for implementing the continuous mode, single (unit) step mode, and group step modes described above. When advancing the clock generator, the shift counter control asserts the run [RUN] output, which enables the shift register counters and output driver flip-flops. In response to a primary system reset, the shift counter control always disables the secondary clock.

Figure 30 is a detail showing the internal view of the reset generator section. This section is responsible for secondary system reset timing. There, the reset monitor state machine insures that the secondary system reset output signal [RSTS_O] is asserted immediately after the primary system reset input signal [RSTP_I]. It keeps the secondary system reset signal asserted until at least one rising clock edge on the secondary clock [CLKS] has occurred, thus insuring that all synchronous logic in the secondary domain is initialized.

The reset generator supports reset blocking. Every block in the secondary domain is simultaneously reset for at least one clock cycle. The duration of the reset output signal [RSTS_O] can be extended by one clock period for every block in series. The number of clock periods is selected by connecting the reset count feedback input [RCFS_I] to one of the output clock pulses. For example, if it is connected to [RSTS_O], then reset is generated for one clock period; if it is connected to [RSTS(2)] it is generated for two clock periods, and so on. Note that the diagram shows sufficient logic for the generation of up to four (4) clock periods, but this may be extended to any number by duplicating the state machine at [RSTS(4)] and adding it to the end of the chain.

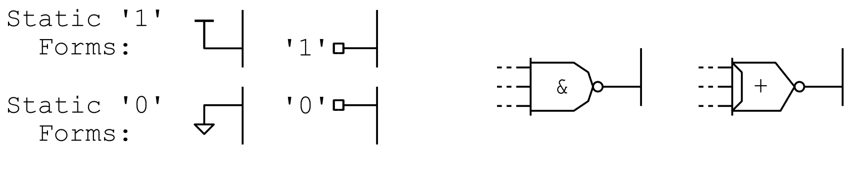
The adjustment method used here was chosen because the number of clock periods may be selected at SoC integration time, or at run time. If selected at SoC integration time the adjustment is a minimization exclusion[39], and any redundant logic will be removed by synthesis tools. However, if the clock generator is to be used with the WISHBONE Off-chip Component Interconnection (WOCI), then the adjustment connections can be routed to I/O pins, and the number of clock periods selected by off-chip switches, jumpers or other arrangements.

Figure 31 is a detail showing the internal view of the secondary reset monitor section. This section is responsible for alerting the primary clock domain when the initialization of the secondary domain is complete, and ready for service.

On the primary side, the secondary reset monitor output signal [SRMP_O] is asserted immediately after the primary system reset input signal [RSTP_I], and is negated after the secondary reset monitor input [SRMS_I] has been negated (but is not sampled until the secondary reset [RSTS_O] has been asserted

[39] ***minimization exclusion***

● A logic element that alters the form, fit or function of a CLASS Machine design representation. ■ *Ex.* The purpose of the *minimization exclusion* is to provide design flexibility while maintaining portability and tool-chain independence. An alternative approach is to rely on the pre-processor directives found in assemblers, compilers, synthesis or other tool-chain elements. However, those techniques are inherently non-portable because they are not tool-chain independent. This problem is overcome by implementing all possible design choices with logic elements in the front end design; by selecting each of these options with static inputs; and by relying on logic minimization tools to strip away any unused sections of the design. Static inputs tie an input high ('1', 'H' or Vcc) or low ('0', 'L' or GND) and operates at synthesis time (as opposed to a dynamic input, which operates at at run-time). In all cases, the same options defined by a *minimization exclusion* could be configured at run-time by enabling them with dynamic inputs, but the resulting target device might require more logic gates and may run slower.



(a) Static inputs

(b) Dynamic inputs

for at least one secondary clock cycle [CLKS_O]).

The input may be daisy-chained with additional clock domains residing below it in a hierarchy domain authority, if present. Thus, the secondary reset monitor output signal [SRMP_O] is negated only when all clock domains are available.

SYSCON.

A timing diagram for a synchronized non-unity trans-domain SYSCON with 4:1 clock divisor ratio is shown in Figure 32. This diagram assumes a single clock system reset on the secondary domain, and that the secondary reset monitor input [SRMS_I] is driven by [SRMS_O] from an identical downstream system controller.

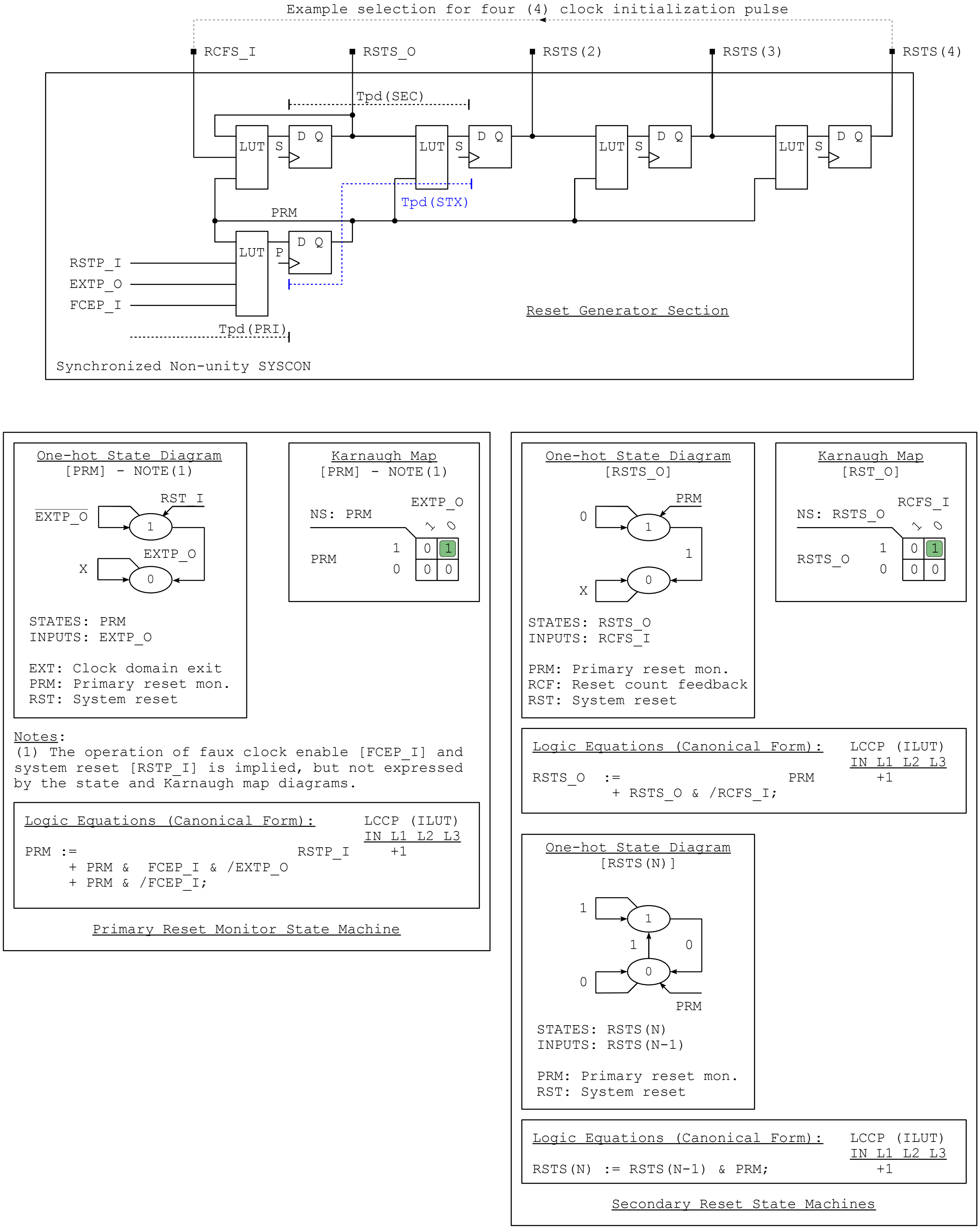
Example: Three-domain SoC.

Figure 33 shows a block diagram of a three domain system using the synchronized non-unity trans-domain SYSCON described above. Figure 34 shows a timing diagram of its reset operation. The three domains are organized as follows:

- *Clock domain #0*. The system clock for this domain is driven by off-chip system clock and system reset inputs. It is responsive to the global reset signal [GSR] which is driven by logic in any of the three domains. $LCCP_{REL}$ = +1 ILUT.

- *Clock domain #1*. The system clock for this domain is driven by a synchronized non-unity trans-domain SYSCON, operating at a frequency divisor ratio of 2:1, and producing a one-clock system reset pulse. $LCCP_{REL}$ = +2 ILUT.

- *Clock domain #2*. The system clock for this domain is driven by a synchronized non-unity trans-domain SYSCON, operating at a frequency divisor ratio of 3:1, and producing a one-clock system reset pulse. $LCCP_{REL}$ = +6 ILUT.



Karnaugh Map [PRM] - NOTE(1)

NS: PRM		EXTP_O 1	EXTP_O 0
PRM	1	0	🟩 1
	0	0	0

Karnaugh Map [RST_O]

NS: RSTS_O		RCFS_I 1	RCFS_I 0
RSTS_O	1	0	🟩 1
	0	0	0

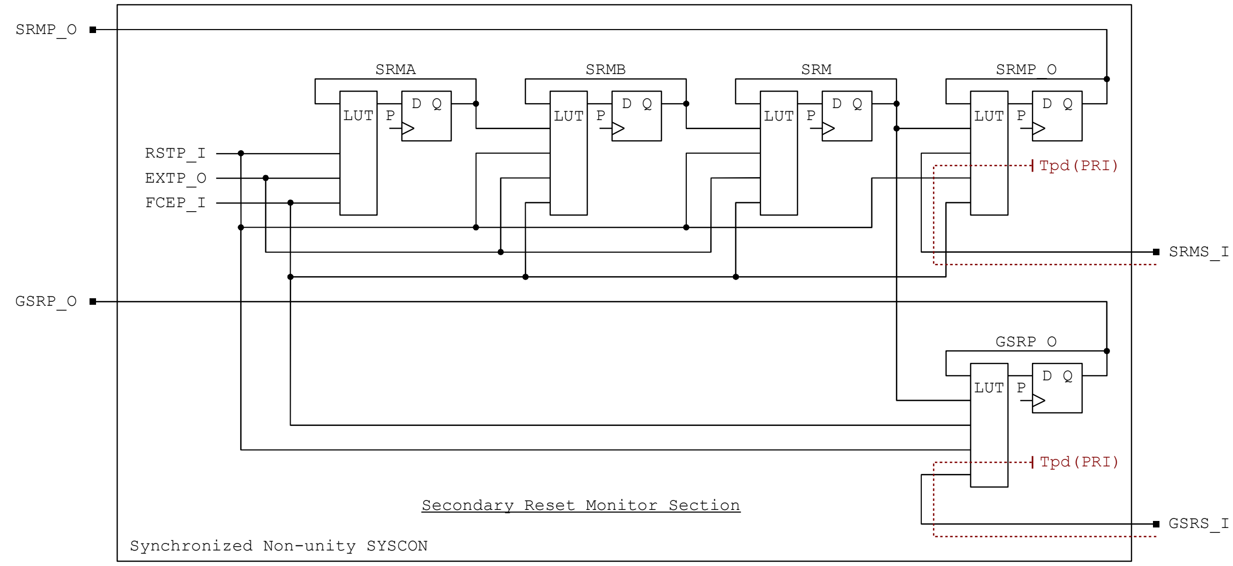
Notes:
(1) The operation of faux clock enable [FCEP_I] and system reset [RSTP_I] is implied, but not expressed by the state and Karnaugh map diagrams.

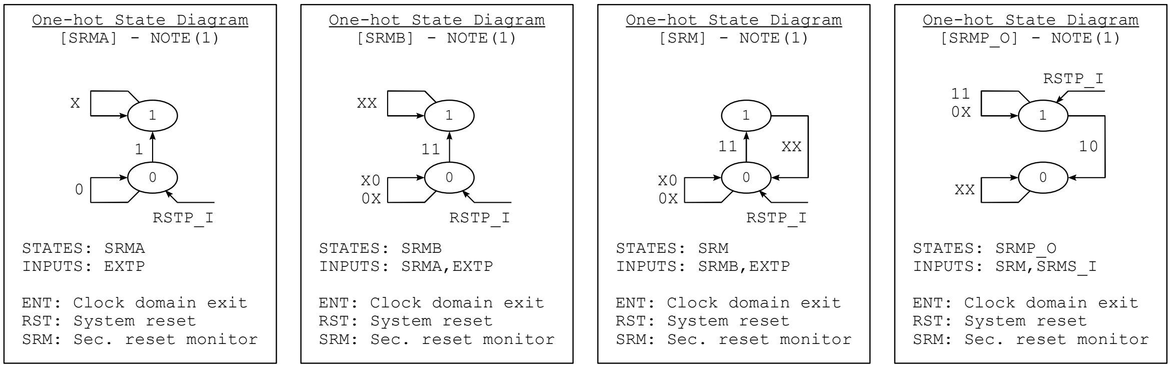
```
Logic Equations (Canonical Form):                  LCCP (ILUT)
                                                   IN L1 L2 L3
PRM :=                                  RSTP_I        +1
     + PRM &  FCEP_I & /EXTP_O
     + PRM & /FCEP_I;
```

```
Logic Equations (Canonical Form):      LCCP (ILUT)
                                       IN L1 L2 L3
RSTS_O  :=                      PRM       +1
        + RSTS_O & /RCFS_I;
```

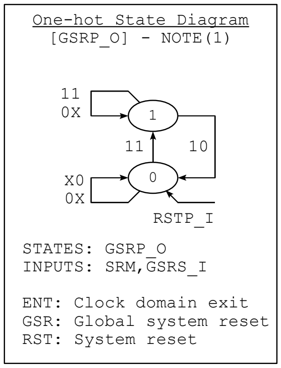
```
Logic Equations (Canonical Form):      LCCP (ILUT)
                                       IN L1 L2 L3
RSTS(N) := RSTS(N-1) & PRM;               +1
```

Figure 30. The reset generator section for the synchronized non-unity trans-domain SYSCON.





```
Logic Equations (Canonical Form):                      LCCP (ILUT)
                                                       IN L1 L2 L3
SRMA   := /SRMA   &  FCEP_I & EXTP_I & /RSTP_I            +1
        +  SRMA                      & /RSTP_I;

SRMB   := /SRMB   &  FCEP_I & EXTP_I & SRMA & /RSTP_I     +1
        +  SRMB                             & /RSTP_I;

SRM    := /SRMB   &  FCEP_I & EXTP_I & SRMB & /RSTP_I     +1
        +  SRMB   & /FCEP_I                 & /RSTP_I;

SRMP_O :=                                      RSTP_I     +1 NOTE(2)
        +  SRMP_O &  FCEP_I &  SRM & SRMS_I
        +  SRMP_O &  FCEP_I & /SRM
        +  SRMP_O & /FCEP_I;

GSRP_O := /GSRP_O &  FCEP_I &  SRM & GSRS_I & /RSTP_I     +1 NOTE(2)
        +  GSRP_O &  FCEP_I &  SRM & GSRS_I & /RSTP_I
        +  GSRP_O &  FCEP_I & /SRM          & /RSTP_I
        +  GSRP_O & /FCEP_I                 & /RSTP_I;
```



Notes:

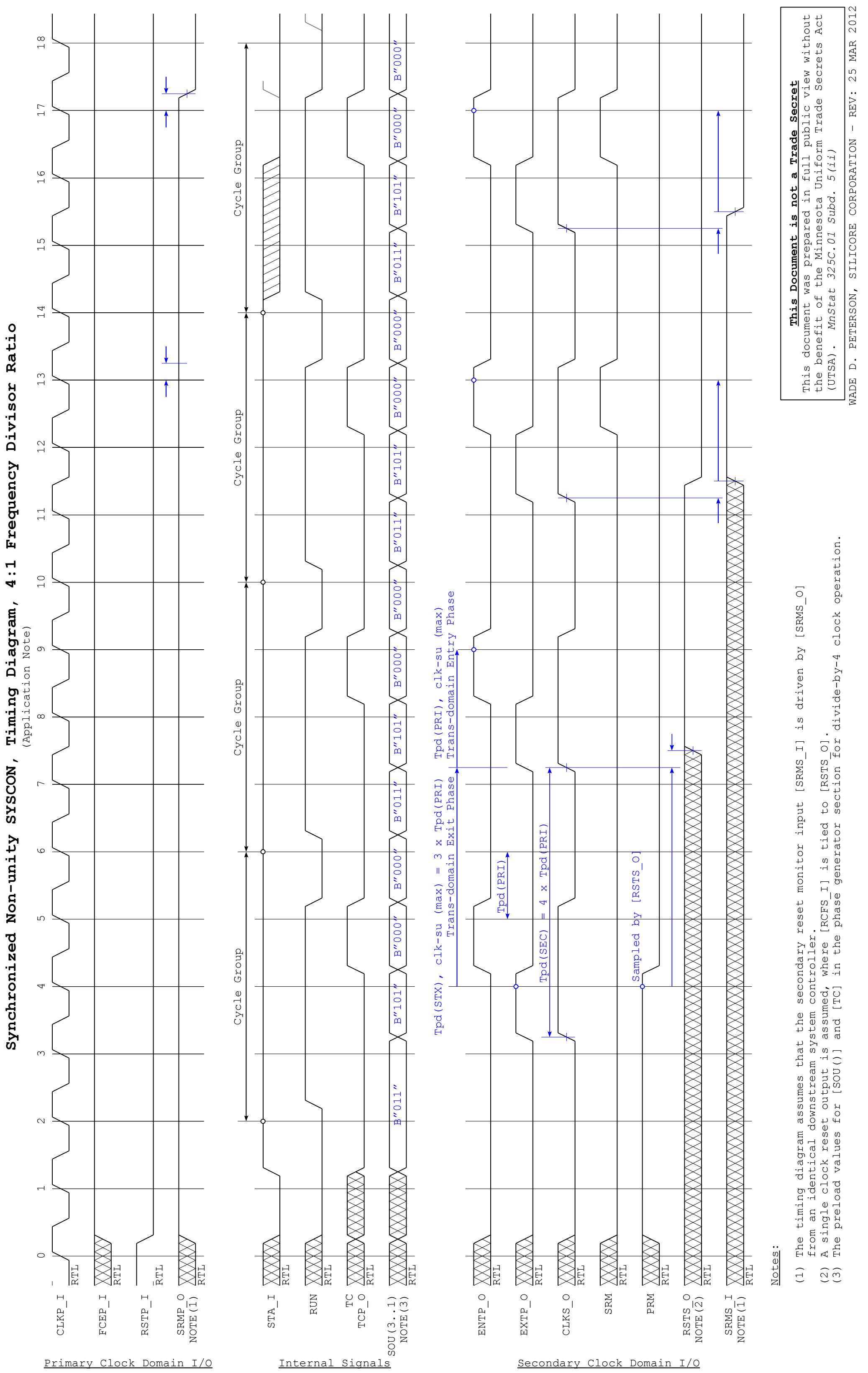
(1) The operation of faux clock enable [FCEP_I] and system reset [RSTP_I] is implied, but not expressed by the state diagrams. Logic equations were determined by direct inspection; the Karnaugh maps are not shown.

(2) The equation contains minterms which are dependent upon a domain crossing entry signal, but which are not qualified by entry phase signal [ENTP_I]. Normally this would be considered poor practice, except that each is qualified by the secondary reset monitor signal [SRM], which has the same timing relationship as the entry phase signal [ENTP_I]. This was done to maintain an LCCP of +1 ILUT.

Domain crossing exit signal inputs are shown in light blue; LCCP = +1 ILUT; Tpd(STX), clk-su.
Domain crossing entry signal inputs are shown in red ; LCCP = +1 ILUT; Tpd(PRI), clk-su.

Secondary Reset Monitor [SRM] State Machines

Figure 31. The secondary reset monitor section for the synchronized non-unity trans-domain



Notes:

(1) The timing diagram assumes that the secondary reset monitor input [SRMS_I] is driven by [SRMS_O] from an identical downstream system controller.

(2) A single clock reset output is assumed, where [RCFS_I] is tied to [RSTS_O].

(3) The preload values for [SOU()] and [TC] in the phase generator section for divide-by-4 clock operation.

This Document is not a Trade Secret

This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Subd. 5(ii)*

WADE D. PETERSON, SILICORE CORPORATION - REV: 25 MAR 2012

Figure 32. Timing diagram for a 4:1 synchronized non-unity trans-domain SYSCON.

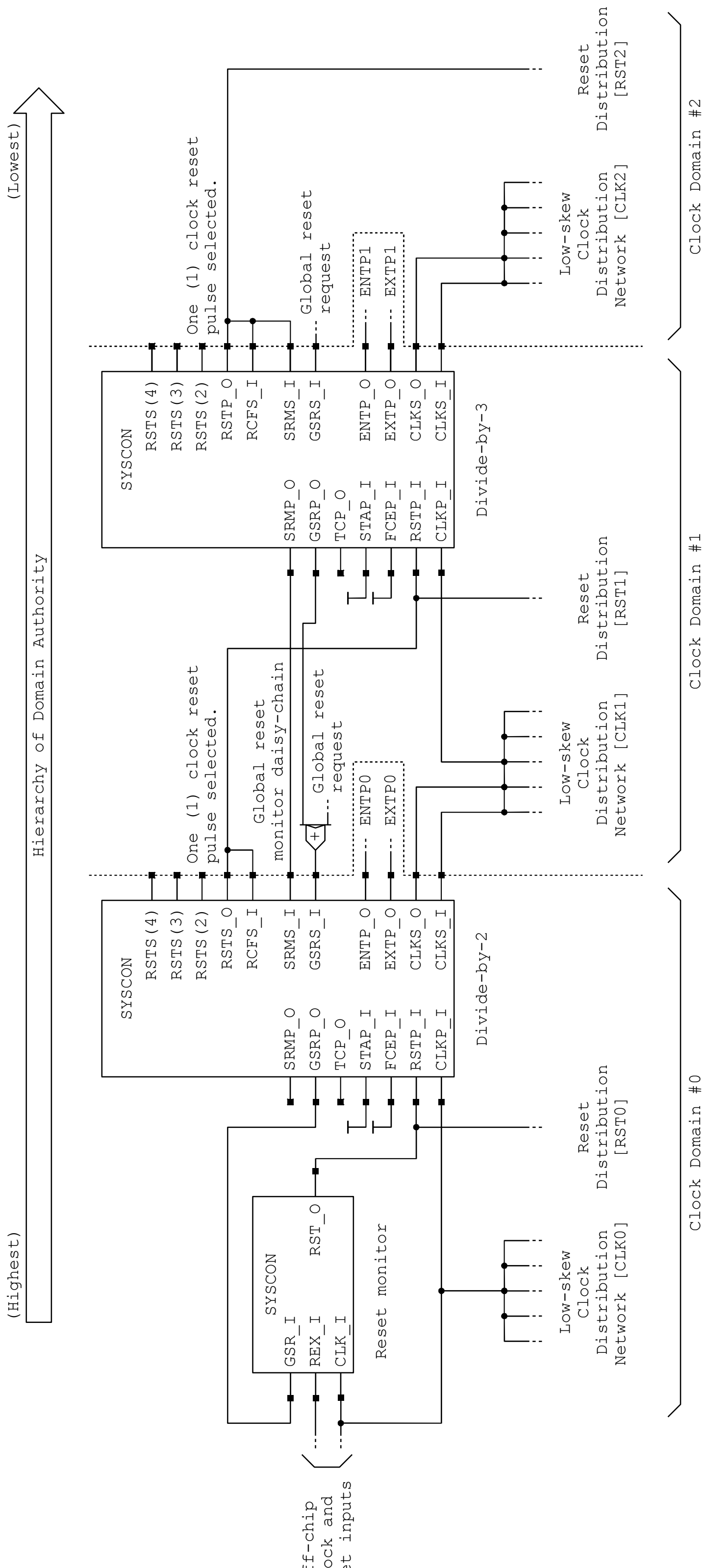
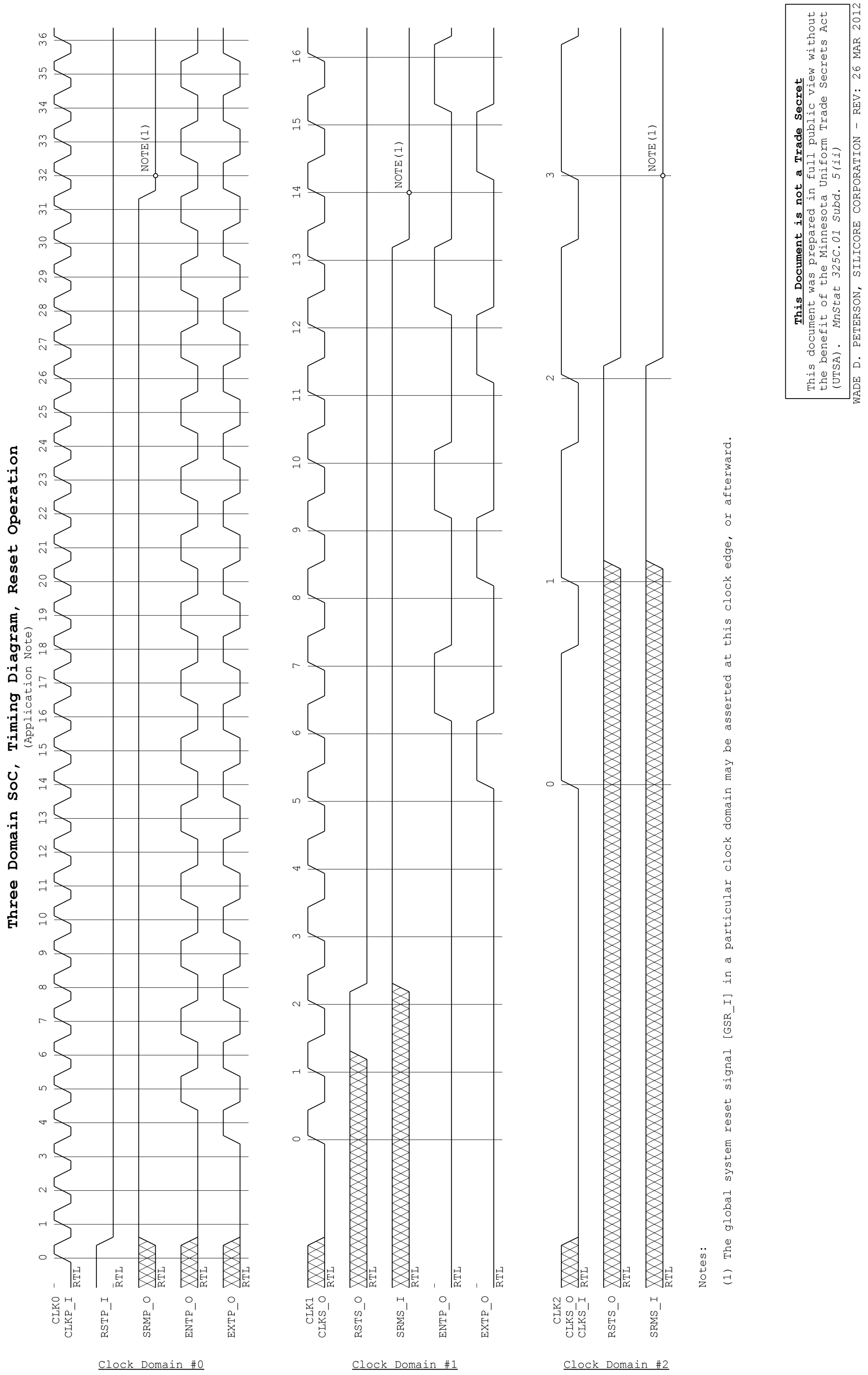


Figure 33. Block diagram for the three domain SoC.



Three Domain SoC, Timing Diagram, Reset Operation

(Application Note)

Notes:

(1) The global system reset signal [GSR_I] in a particular clock domain may be asserted at this clock edge, or afterward.

This Document is not a Trade Secret

This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Subd. 5(ii)*

WADE D. PETERSON, SILICORE CORPORATION - REV: 26 MAR 2012

Figure 34. Timing diagram for the three domain SoC.

Example: Reset Monitor SYSCON

The reset monitor SYSCON is a WISHBONE functional module that drives a system reset output signal [RST_O] in response to an external reset input or a global system reset input.

Figure 35 shows an example of a reset monitor SYSCON which is suitable for use with the synchronized unity or non-unity trans-domain SYSCONs described elsewhere in this report. It asserts the system reset output signal [RST_O] in response to the external reset input [REX_I] or the global system reset [GRS_I].

This section provides a CLASS Machine design representation for this type of SYSCON. There are no special portability requirements other than adherence to timing parameter 'Tpd(AIS), clk-su'. It has a closed-loop $LCCP_{REL}$ = +1 ILUT, leading balance $LCCP_{REL}$ = +1 ILUT, with no lagging balance. This makes it suitable for the highest clock domain in any system.

The external reset input [REX_I] excites an internal double-D type synchronizer to prevent metastable glitches from entering the system[40]. The reset output signal [RST_O] remains asserted so long as the external reset input remains asserted.

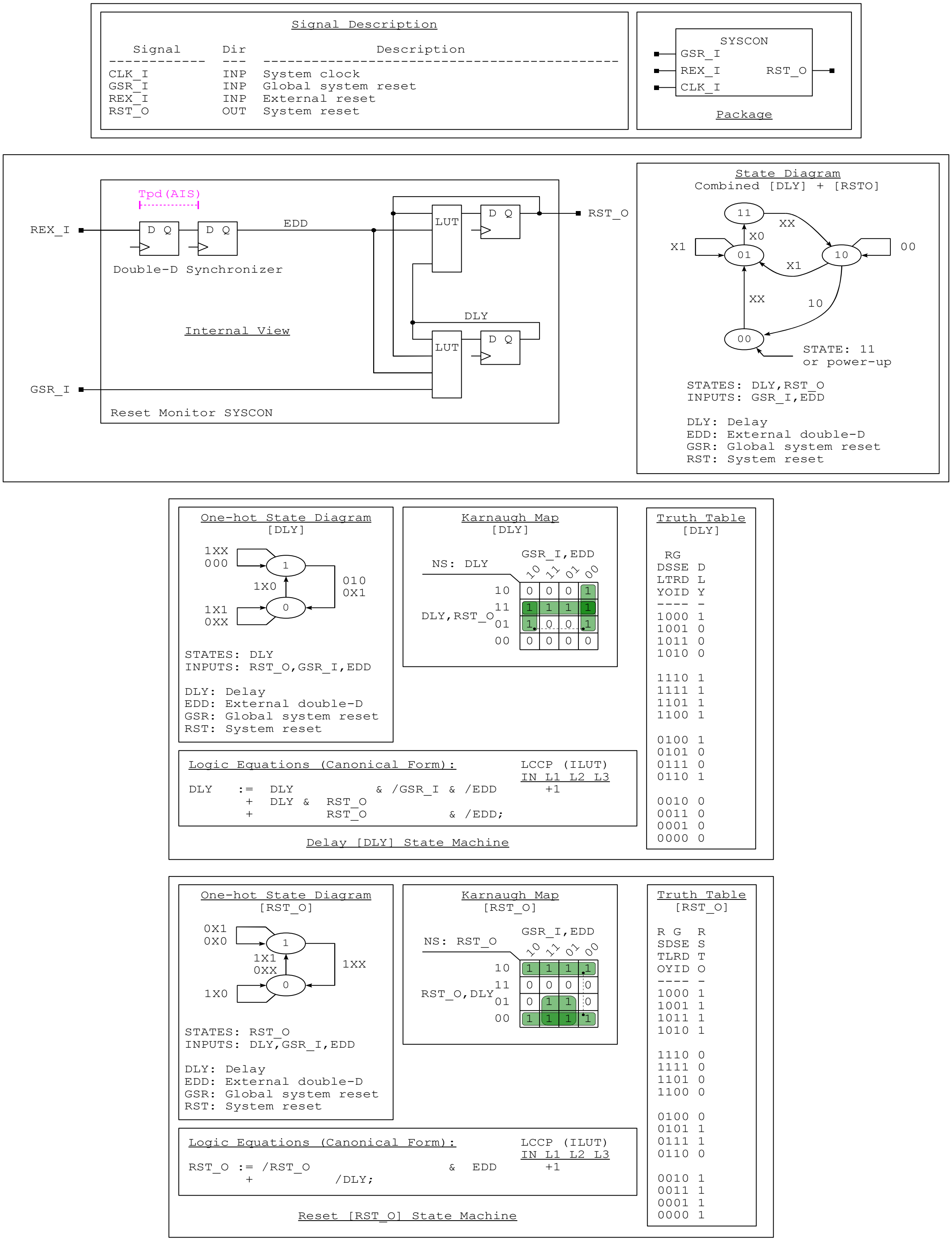
The global system reset input [GSR_I] allows logic functions in a secondary clock domain, or any domain operating below the primary clock domain which is attached to a global reset daisy-chain, to generate a system wide reset.

The reset monitor asserts [RST_O] for two system clock pulses. A minimum of one clock pulse width is required to meet the WISHBONE reset timing, but a double pulse satiates the reset blocking requirements for (a) the secondary reset monitor output in the secondary trans-domain clock controller application, and (b) any unpacked SLAVE interfaces within the domain.

In order to function properly, the internal state machines must be initialized in one of two ways. In a first method, the internal state machines are initialized in response to FPGA device configuration[41], which forces all flip-flops into an initial low state. In a second method, the internal state machines are initialized to a low state in response to the external input [REX_I]. This input signal is also used by simulators to establish initial conditions.

[40] It is anticipated that a noisy external reset input can be further conditioned by a digital low pass filter, such as the 'LPFILTER' described by Peterson (2001a), p. 120.

[41] A similar design can be found in the 'SYC0001a' system controller described in Peterson (2001b), p. 23.



Signal Description

Signal	Dir	Description
CLK_I	INP	System clock
GSR_I	INP	Global system reset
REX_I	INP	External reset
RST_O	OUT	System reset

STATES: DLY,RST_O
INPUTS: GSR_I,EDD

DLY: Delay
EDD: External double-D
GSR: Global system reset
RST: System reset

STATES: DLY
INPUTS: RST_O,GSR_I,EDD

DLY: Delay
EDD: External double-D
GSR: Global system reset
RST: System reset

Karnaugh Map
[DLY]

NS: DLY DLY,RST_O	GSR_I,EDD 10	11	01	00
10	0	0	0	1
11	1	1	1	1
01	1	0	0	1
00	0	0	0	0

Truth Table
[DLY]

RG DSSE LTRD YOID	D L Y
1000	1
1001	0
1011	0
1010	0
1110	1
1111	1
1101	1
1100	1
0100	1
0101	0
0111	0
0110	1
0010	0
0011	0
0001	0
0000	0

```
Logic Equations (Canonical Form):                 LCCP (ILUT)
                                                  IN L1 L2 L3
DLY    :=  DLY            & /GSR_I  & /EDD           +1
        +  DLY  &  RST_O
        +          RST_O            & /EDD;
```

Delay [DLY] State Machine

STATES: RST_O
INPUTS: DLY,GSR_I,EDD

DLY: Delay
EDD: External double-D
GSR: Global system reset
RST: System reset

Karnaugh Map
[RST_O]

NS: RST_O RST_O,DLY	GSR_I,EDD 10	11	01	00
10	1	1	1	1
11	0	0	0	0
01	0	1	1	0
00	1	1	1	1

Truth Table
[RST_O]

R G SDSE TLRD OYID	R S T O
1000	1
1001	1
1011	1
1010	1
1110	0
1111	0
1101	0
1100	0
0100	0
0101	1
0111	1
0110	0
0010	1
0011	1
0001	1
0000	1

```
Logic Equations (Canonical Form):                 LCCP (ILUT)
                                                  IN L1 L2 L3
RST_O := /RST_O                     &  EDD           +1
       +           /DLY;
```

Reset [RST_O] State Machine

Figure 35. Example of a reset monitor SYSCON.

Example: Clock Controller

A clock controller is used for system test and evaluation purposes. It can also stop the clock after a unit step pulse or an N-step pulse train. This example shows how to advance the system clock a predetermined number of clock cycles, stop it, and then observe the state of I/O pins in comparison to known test vectors.

Note that a clock controller can only be used in conjunction with a synchronized non-unity trans-domain SYSCON. This is because the controller operates a gated clock which requires precision control of the secondary clock. This can only be achieved when the secondary clock operates at a lower frequency than the primary.

Figure 36 shows the external view of the clock controller in relation to a synchronized non-unity trans-domain SYSCON. It uses a simple serial shift register as a counter to determine the number of primary clock cycles to advance the clock generator. This topology is generally limited to a few dozen clock pulses, but other techniques (not shown here) can use 32-bit broadside counters (or larger)[42] for very large numbers of clock pulses which can be delivered at the full rated speed of the primary clock.

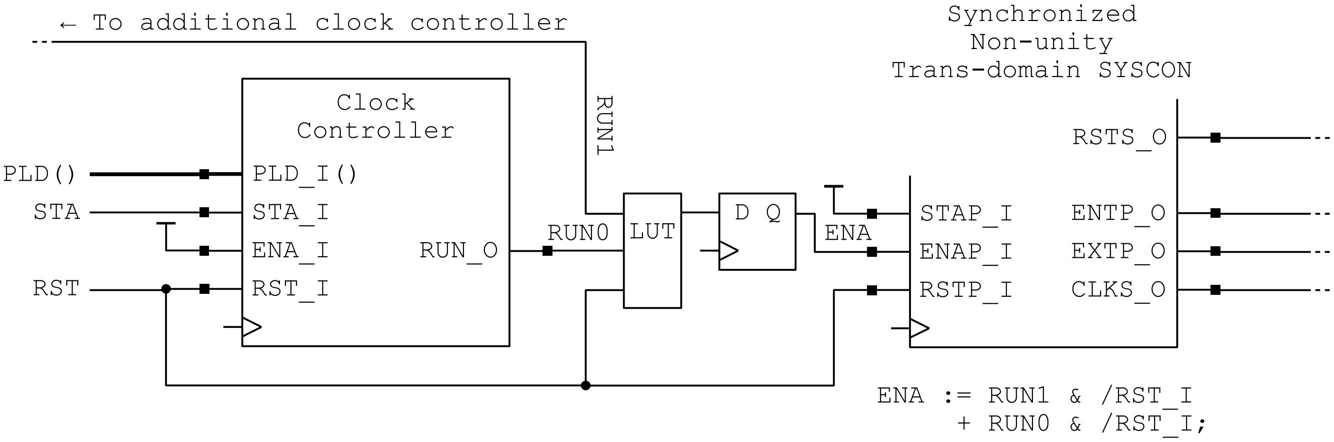


Figure 36. External view of the clock controller.

In the SYSCON example, closed-loop $LCCP_{REL}$ = +1 ILUT, leading balance $LCCP_{REL}$ = +1 ILUT, with no lagging balance. This makes it suitable for the highest clock domain in any system.

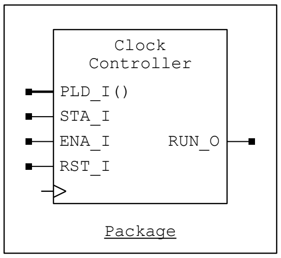
It is presumed that the programmable load [PLD()] might be selected by automated test equipment (which has been omitted for brevity).

For maximum flexibility it is recommended that the clock controller be operated on a clock-by-clock basis rather than on a group-by-group basis (where a 'group' refers to the secondary clock plus the entry and exit phase signals). Groups can still be produced by selecting the correct count in the clock controller (e.g. a 4-count group for a divide-by-four clock generator).

Figure 37 shows the internal view of the clock controller.

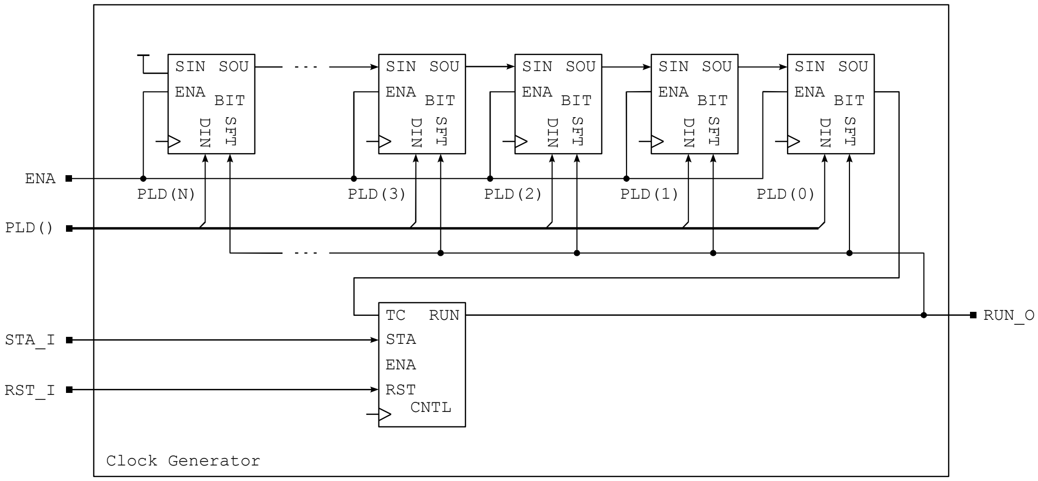
[42] Clock controller counters of any size can be created for domains where $LCCP_{REL}$ = +1 ILUT.

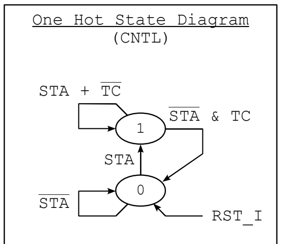
```
                         Signal Description

   Signal     Dir                  Description
------------  ---  -------------------------------------------
CLK_I         INP  System clock, primary clock domain
ENA_I         INP  Standard clock enable for counter
RST_I         INP  System reset, primary clock domain
PLD_I         INP  Counter preload
RUN_O         OUT  Terminal count
STA_I         INP  Start
                      '1': Counter start.  If held high the
                           counter is restarted automatically.
                           If it is negated, then the counter
                           stops at the next terminal count
                           pulse.  Thus, pulsing the [STA]
                           signal for one primary clock
                           cycle will initiate a single count
                           interval.
                      '0': Counter off/stop.
```



Acronyms

BIT:	Shift counter bit
CLK:	System clock
CNTL:	Shift counter control
DIN:	Data in
ENA:	Standard clock enable
ENT:	Clock domain entry
EXT:	Clock domain exit
MOD:	Mode (single or continuous)
PLD:	Counter preload
RST:	System reset
RUN:	Counter running
SFT:	Shift/load
SIN:	Serial in
SOU:	Serial out
STA:	Counter start
TC:	Terminal count

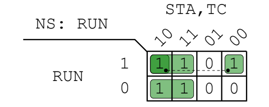




STATES: RUN
INPUTS: STA,TC

RST: System reset
RUN: Counter run
STA: Counter start
TC: Terminal count

Karnaugh Map
(CNTL)


NS: RUN
STA,TC
10
11
01
00
RUN
1
0

NS: RUN		10	11	01	00
RUN	1	1	1	0	1
	0	1	1	0	0

```
Logic Equations (Canonical Form):  LCCP (ILUT)
                                   IN L1 L2 L3
RUN :=  RUN & STA        & /RST_I      +1
     +  RUN       & /TC  & /RST_I
     + /RUN & STA        & /RST_I;
```

Control (CNTL) State Machine

Figure 37. Internal view of the clock controller.

Example: Synchronized Non-unity Trans-domain BRIDGE

A synchronized non-unity trans-domain BRIDGE connects two synchronized clock domain regions. It applies when the primary clock domain operates at a higher frequency than the secondary clock domain; where the primary and secondary clocks are synchronized; and where a primary-to-secondary frequency ratio equal to or greater than 2:1 exists (i.e. 2:1, 3:1, 4:1, etc.).

This section provides a CLASS Machine design representation for this type of BRIDGE. There are no special portability requirements other than two low-skew clock distribution networks, and adherence to the following transfer delays at a synchronized trans-domain boundary: (i) the destination clock domain transfer delay 'Tpd(DOM)', (ii) the synchronized trans-domain entry transfer delay 'Tpd(STE), clk-su' and (iii) the synchronized trans-domain exit transfer delay 'Tpd(STX), clk-su'. It has a closed-loop $LCCP_{REL}$ = +2 ILUT, leading balance $LCCP_{REL}$ = +2 ILUT, with no lagging balance.

It is an automatic BRIDGE, meaning that it responds automatically to bus cycles presented to its SLAVE interface.

It is a bidirectional BRIDGE, meaning it has two MASTER and two SLAVE interfaces in opposition which propagate bus cycles in two directions. SLAVE interfaces on a bidirectional BRIDGE will also resolve a deadly embrace by asserting the retry output signal [RTY_O]. However, it can also be configured as a unidirectional BRIDGE through the use of a minimization exclusion. This also eliminates the timing overhead associated with an internal deadlock arbiter.

The BRIDGE supports all combinations of DATUM INTERLOCKED cycles, which include:

```
CONGA{
            DATUM INTERLOCKED( READ() ),
            DATUM INTERLOCKED( WRITE() ),
            DATUM INTERLOCKED( DUPLEX() ),
            DATUM INTERLOCKED( EVENT() )
};
```

The BRIDGE relies on signals produced by the synchronized non-unity trans-domain SYSCON described elsewhere in this report. It operates at all valid combinations of primary-to-secondary clock ratios of 2:1, 3:1, 4:1, … up to ∞:1.

Closed-loop Control Transport Structure

Figure 38 shows the general topology of the closed-loop control transport structure for the synchronized non-unity trans-domain BRIDGE. Logic equations are shown in Figure 39.

The closed-loop control transport structure for a BRIDGE always begins at the strobe output signal [STB_O] of a remote MASTER interface, and ends with the terminal signal (e.g. [ACK_I]) at the same MASTER. The detailed operating description which follows is for a primary-to-secondary (PRI->SEC) transaction during unidirectional operation, where the remote interfaces are connected by point-to-point interconnections. The following steps complete the closed-loop control transport process:

1. *Remote MASTER.* Remote MASTER addresses BRIDGE SLAVE with a DATUM INTERLOCKED cycle, causing primary strobe input signal [STBP_I] to be asserted.

2. *Primary SLAVE Loop Logic*. Primary SLAVE Loop Logic generates primary strobe signal [STBP], which is synchronized to the secondary clock domain by exit phase input signal [EXTP_I]. The primary strobe signal [STBP] is asserted for the duration of one secondary clock cycle to insure detection by secondary logic. The signal propagates across the clock domain boundary, and must be registered at the secondary side within the time specified by the propagation delay exit specification Tpd(STX), clk-su.

   Note: if a BLOCK INTERLOCKED cycle is requested (i.e. CONGA address tag [CATP_I] is asserted), then the error output signal [ERRP_O] is returned to the remote MASTER, and the primary strobe signal [STBP] is never asserted.

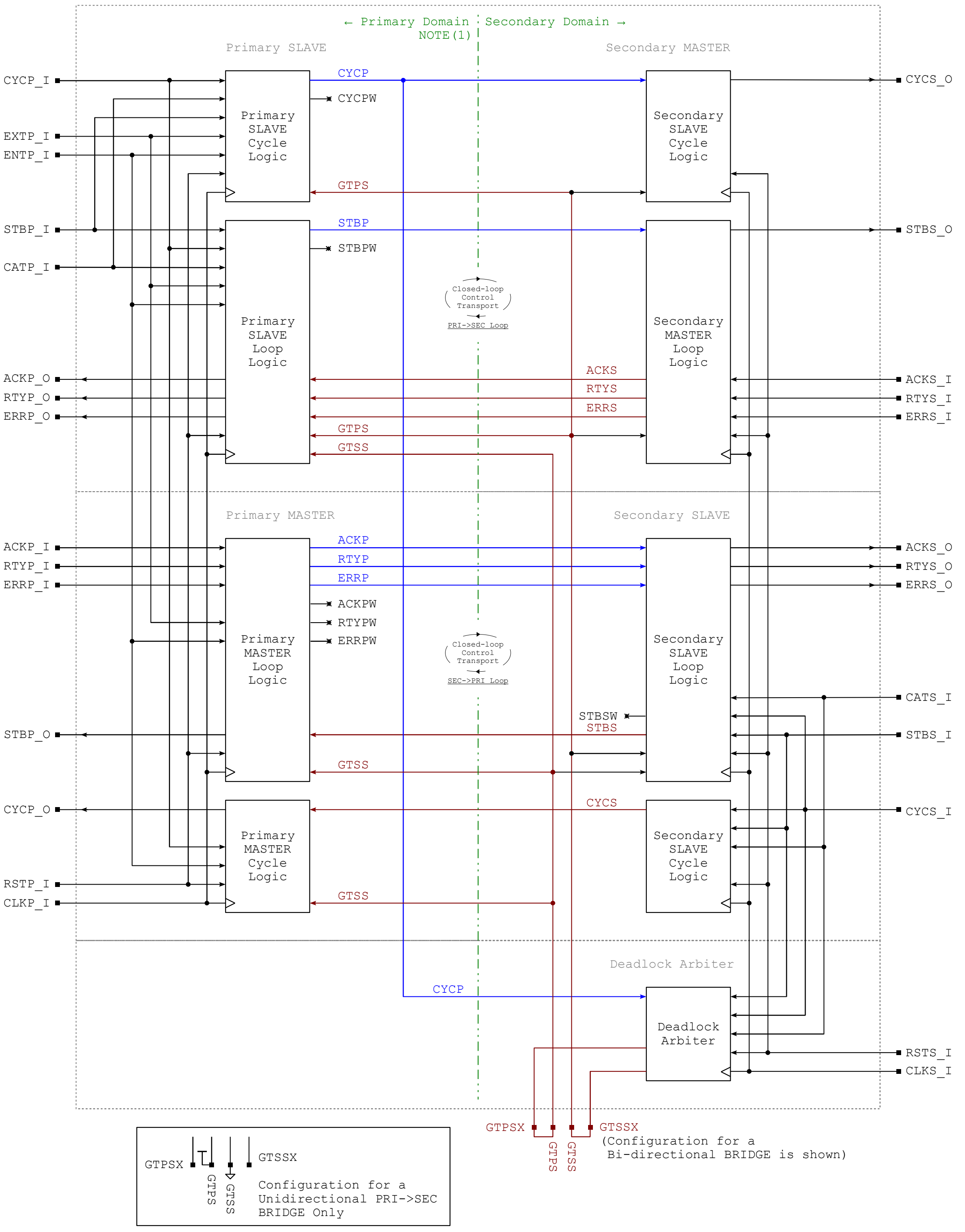
3. *Secondary MASTER Loop Logic*. Secondary MASTER Loop Logic responds by asserting secondary strobe output signal [STBS_O]. The remote SLAVE interface responds by asserting a terminal signal (e.g. [ACKS_I]), which causes the Secondary MASTER Loop Logic to assert one of three related domain-crossing signals. Thus, if acknowledge input signal [ACKS_I] is received from the remote SLAVE, then the secondary acknowledge signal [ACKS] is asserted, if [RTYS_I] then [RTYS] or if [ERRS_I] then [ERRS]. The asserted signal propagates back across the clock domain boundary, and is registered at the primary side within the time specified by the propagation delay entry specification Tpd(STE), clk-su.

4. *Primary SLAVE Loop Logic*. Primary SLAVE Loop Logic responds by asserting a related terminal signal back to the remote MASTER. Thus, if secondary acknowledge signal [ACKS] is asserted, then the acknowledge output signal [ACKP_O] is sent to the remote MASTER; if [RTYS] then [RTYS_O] or if [ERRS] then [ERRS_O].

5. *Remote MASTER*. Remote MASTER terminates the cycle causing primary strobe input signal [STBP_I] to be negated. Alternatively, the MASTER can begin another phase of a multi-phase DATUM INTERLOCKED cycle.

When configured as a bidirectional BRIDGE, the control transport structure also implements a deadlock arbiter, which resides in the secondary clock domain. Figure 40 shows the truth table and Karnaugh maps for the arbiter. Deadly embrace (deadlock) is a situation that might occur when both sides of the BRIDGE attempt to access the other at the same time. This is prevented when the deadlock arbiter recognizes the situation and responds by allowing one side to complete its bus cycle, and terminates the other by asserting the retry output signal [RTYP_O] or [RTYS_O].

Ownership of the bidirectional BRIDGE is requested only after a SLAVE monitors that both the cycle input signal [CYC_I] and strobe input signal [STB_I] are both asserted. Once the deadlock arbiter grants ownership of the BRIDGE, a request from the opposite side of the BRIDGE is rejected by asserting the retry output signal [RTYP_O] or [RTYS_O]. In the event that requests are obtained simultaneously from both sides, the deadlock arbiter grants the higher priority to the primary interface. Ownership is maintained until the related cycle input signal [CYCP_I] or [CYCS_I] is negated.



Notes:

(1) Domain crossing exit signal inputs are shown in light blue; LCCP = +1 ILUT; Tpd(STX), clk-su.
Domain crossing entry signal inputs are shown in red ; LCCP = +1 ILUT; Tpd(PRI), clk-su.

Figure 38. Control transport structure for the synchronized non-unity trans-domain BRIDGE.

```
Logic Equations (Canonical Form):
Automatic Bi-directional Control Transport, NOTE(1)                                     LCCP (ILUT)
                                                                                        IN L1 L2 L3       Notes / Example LUT Functions
CYCP   :=   /CYCP &  EXTP_I &  /CYCPW &  CYCP_I &  STBP_I & /CATP_I & /RSTP_I                 +2     L2=(CYCP,EXTP_I,CYCPW,RSTP_I,1A);
        +    CYCP           &   CYCPW                               & /RSTP_I                        1A=(CYCP_I,STBP_I,CATP_I,0,0);
        +    CYCP & /EXTP_I &  /CYCPW                               & /RSTP_I;

CYCPW  :=  /CYCPW &  ENTP_I &    CYCP &    GTPS           & /RSTP_I                           +2     L2=(GTPS,1A,1B,RSTP_I,0);
        +   CYCPW                               &  CYCP_I & /RSTP_I;                                 1A=(CYCPW,ENTP_I,CYC,0,0);
                                                                                                     1B=(CYCPW,CYCP_I,0,0,0);
STBP   :=            EXTP_I &  /STBPW &  CYCP_I &  STBP_I & /CATP_I & /RSTP_I                 +2     L2=(STBP,EXTP_I,RSTP_I,1A,0)
        +    STBP & /EXTP_I                                         & /RSTP_I;                       1A=(STBPW,CYCP_I,STBP_I,CATP_I,0);

STBPW  :=  /STBPW &  ENTP_I &    GTPS &    STBP                               & /RSTP_I       +2     L2=(GTPS,1A,1B,0,0);
        +   STBPW                               & /ACKP_O & /RTYP_O & /ERRP_O & /RSTP_I;             1A=(STBPW,ENTP_I,STBP,RSTP_I,0);
                                                                                                     1B=(STBPW,ACKP_O,RTYP_O,ERRP_O,RSTP_I);
ACKP_O := /ACKP_O &  ENTP_I           &    ACKS &  CYCP_I &  STBP_I & /CATP_I & /RSTP_I;      +2     L2=(ACKP_O,ENTP_I,ACKS,1A,0);
                                                                                                     1A=(CYCP_I,STBP_I,CATP_I,RSTP_I,0);
RTYP_O := /RTYP_O &  ENTP_I           &    RTYS &  CYCP_I &  STBP_I & /CATP_I & /RSTP_I       +2     L2=(RTYP_O,ENTP_I,GTSS,RTYS,1A);
        + /RTYP_O &  ENTP_I &    GTSS           &  CYCP_I &  STBP_I & /CATP_I & /RSTP_I;             1A=(CYCP_I,STBP_I,CATP_I,RSTP_I,0);

ERRP_O := /ERRP_O &  ENTP_I           &    ERRS &  CYCP_I &  STBP_I & /CATP_I & /RSTP_I       +2     L2=(ERRP_O,ENTP_I,ERRS,1A,1B);
NOTE(2) + /ERRP_O                               &  CYCP_I &  STBP_I &  CATP_I & /RSTP_I;             1A=(CYCP_I,STBP_I,CATP_I,RSTP_I,0);
                                                                                                     1B=(CYCP_I,STBP_I,CATP_I,RSTP_I,0);
CYCP_O :=            ENTP_I &    GTSS &    CYCS & /RSTP_I                                  +1        Self-evident (1 LUT)
        +           /ENTP_I                     & /RSTP_I;

STBP_O := /STBP_O &  ENTP_I &    GTSS &    STBS                               & /RSTP_I       +2     L2=(GTSS,STBS,1A,1B,0);)
        +  STBP_O                               & /ACKP_I & /RTYP_I & /ERRP_I & /RSTP_I;             1A=(STBP_O,ENTP_I,RSTP_I,0,0);
                                                                                                     1B=(STBP_O,ACKP_I,RTYP_I,ERRP_I,RSTP_I);
ACKPW  :=  /ACKPW           &  STBP_O &  ACKP_I & /RSTP_I                                  +1        Self-evident (1 LUT)
        +   ACKPW & /EXTP_I                     & /RSTP_I;

RTYPW  :=  /RTYPW           &  STBP_O &  RTYP_I & /RSTP_I                                  +1        Self-evident (1 LUT)
        +   RTYPW & /EXTP_I                     & /RSTP_I;

ERRPW  :=  /ERRPW           &  STBP_O &  ERRP_I & /RSTP_I                                  +1        Self-evident (1 LUT)
        +   ERRPW & /EXTP_I                     & /RSTP_I;

ACKP   :=   /ACKP &  EXTP_I &  STBP_O &  ACKP_I           & /RSTP_I                           +2     L2=(ACKP,EXTP_I,RSTP_I,ACKPW,1B);
        +   /ACKP &  EXTP_I                     &   ACKPW & /RSTP_I                                  1A=(STBP_O,ACKP_I,0,0,0);
        +    ACKP & /EXTP_I                               & /RSTP_I;

RTYP   :=   /RTYP &  EXTP_I &  STBP_O &  RTYP_I           & /RSTP_I                           +2     L2=(RTYP,EXTP_I,RSTP_I,RTYPW,1B);
        +   /RTYP &  EXTP_I                     &   RTYPW & /RSTP_I                                  1A=(STBP_O,RTYP_I,0,0,0);
        +    RTYP & /EXTP_I                               & /RSTP_I;

ERRP   :=   /ERRP &  EXTP_I &  STBP_O &  ERRP_I           & /RSTP_I                           +2     L2=(ERRP,EXTP_I,RSTP_I,ERRPW,1B);
        +   /ERRP &  EXTP_I                     &   ERRPW & /RSTP_I                                  1A=(STBP_O,ERRP_I,0,0,0);
        +    ERRP & /EXTP_I                               & /RSTP_I;

CYCS_O := /CYCS_O &    GTPS &    CYCP & /RSTS_I                                            +1        Self-evident (1 LUT)
        +  CYCS_O           &    CYCP & /RSTS_I;

STBS_O := /STBS_O &    GTPS &    STBP                               & /RSTS_I;                +2     L2=(STBS_O,GTPS,STBP,RSTS_I,1A);
        +  STBS_O                     & /ACKS_I & /RTYS_I & /ERRS_I & /RSTS_I;                       1A=(STBS_O,ACKS_I,RTYS_I,ERRS_I,RSTS_I);

ACKS   :=   /ACKS &  STBS_O &  ACKS_I & /RSTS_I;                                           +1        Self-evident (1 LUT)

RTYS   :=   /RTYS &  STBS_O &  RTYS_I & /RSTS_I;                                           +1        Self-evident (1 LUT)

ERRS   :=   /ERRS &  STBS_O &  ERRS_I & /RSTS_I;                                           +1        Self-evident (1 LUT)

CYCS   :=   /CYCS &  CYCS_I &  STBS_I & /CATS_I & /RST_I                                   +1        Self-evident (1 LUT)
NOTE(3) +    CYCS &  CYCS_I                     & /RST_I;

STBS   :=   /STBS &  CYCS_I &  STBS_I & /CATS_I & /STBSW                      & /RSTS_I;   +1        Self-evident (1 LUT)

STBSW  :=  /STBSW &  CYCS_I &  STBS_I & /CATS_I                               & /RSTS_I       +2     L2=(STBSW,1A,1B,0,0);
        +   STBSW                               & /ACKS_O & /RTYS_O & /ERRS_O & /RSTS_I;             1A=(CYCS_I,STBS_I,CATS_I,RSTS_I,0);
                                                                                                     1B=((STBSW,ACKS_O,RTYS_O,ERRS_O,RSTS_I);
ACKS_O := /ACKS_O &    GTSS &    ACKP &  CYCS_I & STBS_I  & /CATS_I & /RSTS_I;                +2     L2=(ACKS_O,GTSS,ACKP,CYC_I,1A);
                                                                                                     1A=(STBS_I,CATS_I,RSTS_I,0,0);
RTYS_O := /RTYS_O &    GTSS           &    RTYP &  CYCS_I & STBS_I  & /CATS_I & /RSTS_I       +2     L2=(GTSS,GTPS,RTYP,1A,0);
        + /RTYS_O           &    GTPS           &  CYCS_I & STBS_I  & /CATS_I & /RSTS_I;             1A=(RTYS_O,CYCS_I,STBS_I,CATS_I,RSTS_I);

ERRS_O := /ERRS_O &    GTSS &    ERRP &  CYCS_I & STBS_I  & /CATS_I & /RSTS_I                 +2     L2=(GTSS,ERRP,CATS_I,1A,0);
        + /ERRS_O                     &  CYCS_I & STBS_I  &  CATS_I & /RSTS_I;                       1A=(ERRS_O,CYCS_I,STBS_I,RSTS_I,0);

GTPSX  :=   /GTSSX &   CYCP           & /RSTS_I                                            +1        Self-evident (1 LUT)
NOTE(4) +              CYCP & /CYCS_I & /RSTS_I;

GTSSX  :=    GTSSX          &  CYCS_I                    & /RSTS_I                            +2     L2=(GTSSX,CYCS_I,RSTS_I,1A,0);
NOTE(4) +             /CYCP &  CYCS_I &  STBS_I & CATS_I & /RSTS_I;                                  1A=(CYCP,STBS_I,CATS_I,0,0);
```

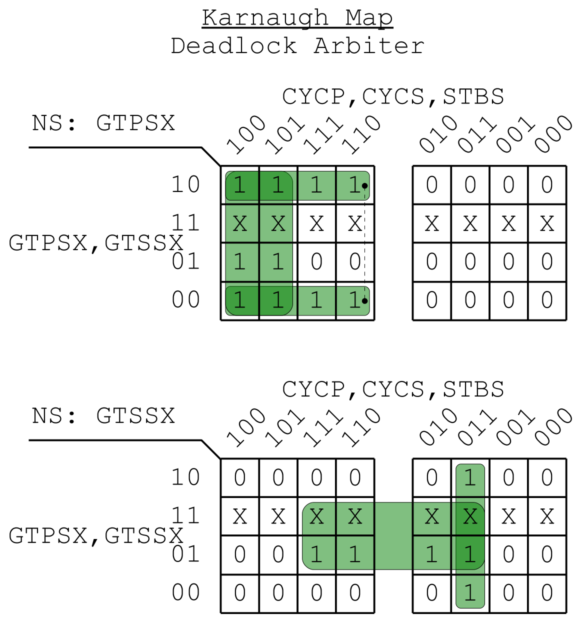
Notes:

(1) Domain crossing exit signal inputs are shown in light blue; LCCP = +1 ILUT; Tpd(STX), clk-su.
    Domain crossing entry signal inputs are shown in   red   ; LCCP = +1 ILUT; Tpd(PRI), clk-su.
(2) The interface responds with an error when BLOCK INTERLOCKED cycles are directed at the SLAVE.
(3) The initial strobe phase within a CONGA cycle which is directed at the SLAVE MUST be a DATUM INTERLOCKED cycle. Once a MASTER in a first block is granted the bus in a second block, the remote interconnection is held by the MASTER until it negates its cycle signal [CYC].
(4) Bidirectional operation: [GTSS] = [GTSSX]; [GTPS] = [GTPSX].
    Unidirectional PRI->SEC operation: [GTPS] = '1'; [CYCS_I],[STBS_I],[CATS_I],[ACKP_I],[RTYP_I],[ERRP_I] = '0'.
    Unidirectional SEC->PRI operation: [GTSS] = '1'; [CYCP_I],[STBP_I],[CATP_I],[ACKS_I],[RTYS_I],[ERRS_I] = '0'.

Figure 39. Logic equations for the synchronized non-unity trans-domain BRIDGE.

Truth Table

GGCCS	GG
TTYYT	TT
PSCCB	PS
SSPSS	SS
XX	XX
-----	--
10000	00
10001	00
10011	01
10010	00
10110	10
10111	10
10101	10
10100	10
11100	XX
11101	XX
11111	XX
11110	XX
11010	XX
11011	XX
11001	XX
11000	XX
01000	00
01001	00
01011	01
01010	01
01110	01
01111	01
01101	10
01100	10
00100	10
00101	10
00111	10(*)
00110	10
00010	00
00011	01
00001	00
00000	00



Karnaugh Map
Deadlock Arbiter

NS: GTPSX

GTPSX,GTSSX \ CYCP,CYCS,STBS	100	101	111	110	010	011	001	000
10	1	1	1	1	0	0	0	0
11	X	X	X	X	X	X	X	X
01	1	1	0	0	0	0	0	0
00	1	1	1	1	0	0	0	0

NS: GTSSX

GTPSX,GTSSX \ CYCP,CYCS,STBS	100	101	111	110	010	011	001	000
10	0	0	0	0	0	1	0	0
11	X	X	X	X	X	X	X	X
01	0	0	1	1	1	1	0	0
00	0	0	0	0	0	1	0	0

General Assumptions

Initial secondary cycle must show [CYCS_I] and [STBS_I] both asserted because signal [$\overline{\text{CYC}}$] is broadcast to all SLAVEs, without address decoding.

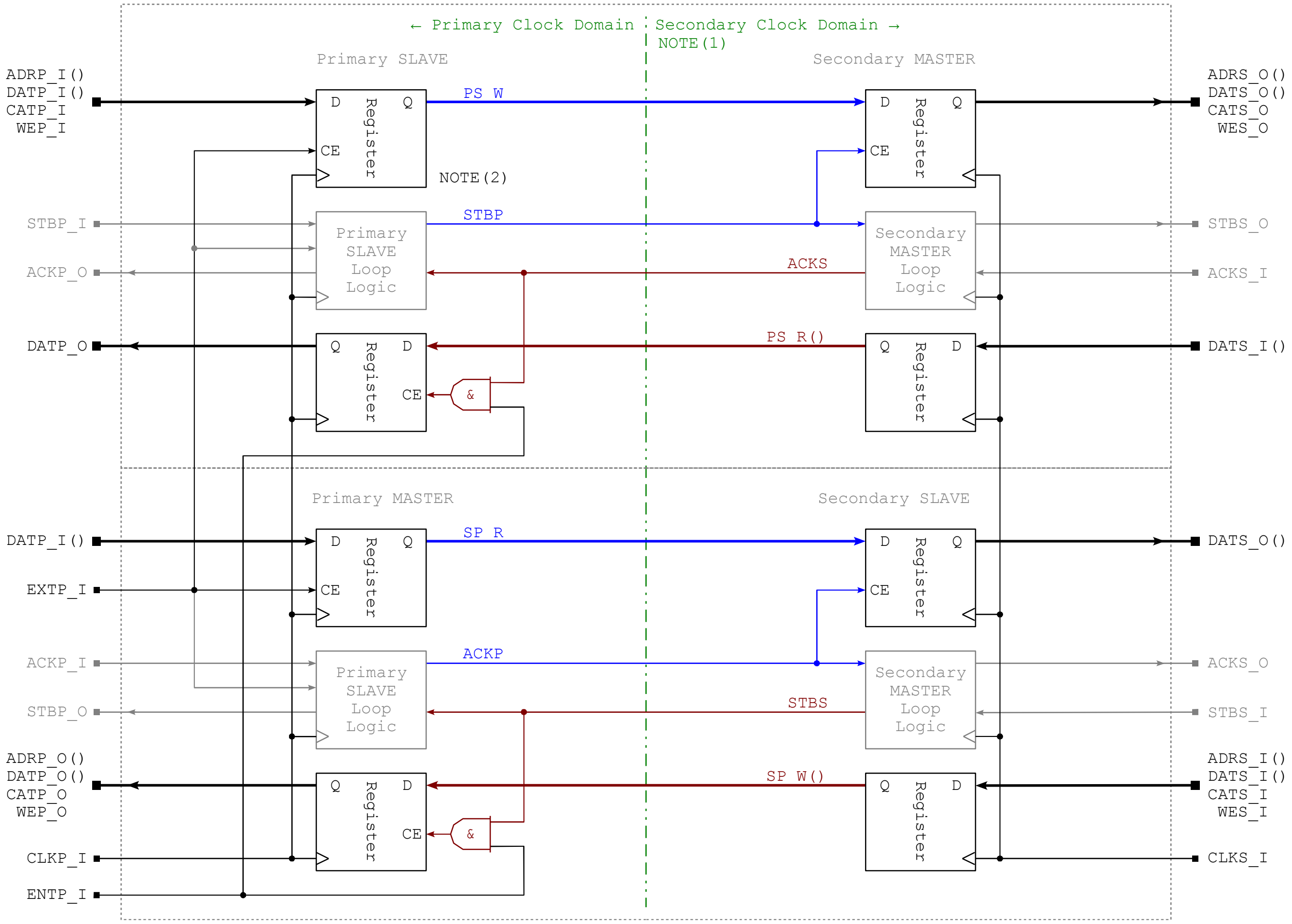
(*) Primary has priority over secondary.

Figure 40. Deadlock arbiter logic for the synchronized non-unity trans-domain BRIDGE.

Data Transport Structure

Figure 41 shows the data transport structure for the synchronized non-unity trans-domain BRIDGE. It demonstrates how identical registers and logic are used as generic transports for address bus, data bus, CONGA address tag and write enable signals. The blocks identified as Primary SLAVE Loop Logic (etc.) are the same as those shown in the earlier control transport diagram.

Note that the write enable input signals [WEP_I] and [WES_I] do not impact the flow of data through the BRIDGE.



```
Notes:

(1) Domain crossing exit  signal inputs are shown in light blue; LCCP = +1 ILUT; Tpd(STX), clk-su.
    Domain crossing entry signal inputs are shown in     red    ; LCCP = +1 ILUT; Tpd(PRI), clk-su.

(2) Typical register equations for DAT() (bitwise): DATPS_W(00)  := DATPS_I(00)  &  CE
                                                                  + DATPS_W(00)  & /CE;

                                                    DATSP_O(00)  := DATSP_W(00)  &  CE  &  ENTP_I
                                                                  + DATSP_O(00)        & /ENTP_I
                                                                  + DATSP_O(00)  & /CE;
```

Figure 41. Data transport structure for the synchronized non-unity trans-domain BRIDGE.

System Reset Operation

The BRIDGE is designed to work in conjunction with the synchronized non-unity trans-domain SYSCON described elsewhere in this report. The two sides of the BRIDGE respond independently to primary and secondary reset input signals [RSTP_I] and [RSTS_I].

The secondary side of the BRIDGE will always have its reset signal asserted and negated sometime after the primary side, and (depending upon the frequency divisor ratio) there can be a considerable delay between them. Until the secondary side is reset the domain crossing signal levels from that side are unknown, which could result in an erratic state on the primary side. In theory, further complications might arise when a remote MASTER on the primary side of the bridge begins a bus cycle before the secondary side is initialized.

These problems are overcome by using the recommended practices shown in Figure 42. That uses the secondary reset monitor input signal [SRMS_I] and output signal [SRMP_O] to excite the BRIDGE reset input signals [RSTS_I] and [RSTP_I] respectively. Thus:

- *Coincident reset*. The reset signals on both sides of the BRIDGE are guaranteed to be reset at the same time for at least one clock interval, regardless of clock divisor ratio. This prevents any transitory states from producing a erratic state at one of the WISHBONE interfaces.

- *Early remote MASTER cycle*. Bus cycles from remote MASTERs can take place immediately after a local (primary or secondary) reset signal is negated. These are ignored by the BRIDGE SLAVE interfaces until the secondary reset monitor signal is negated.

The diagram also shows a theoretical context switch for evaluating the source of the secondary reset monitor input signal [SRMS_I]. When set to the local position 'LCL', it is assumed that the source is the secondary reset distribution [RSTS]. When set to the global position 'GLO', it is assumed that further (lower frequency) clock domains are present, and that the source is the global reset monitor daisy-chain. Either way, the BRIDGE will function correctly.

Figures 43 and 44 shows reset timing diagrams for the PRI->SEC and SEC->PRI cycles for unidirectional BRIDGEs with 3:1 clock divisor ratios.

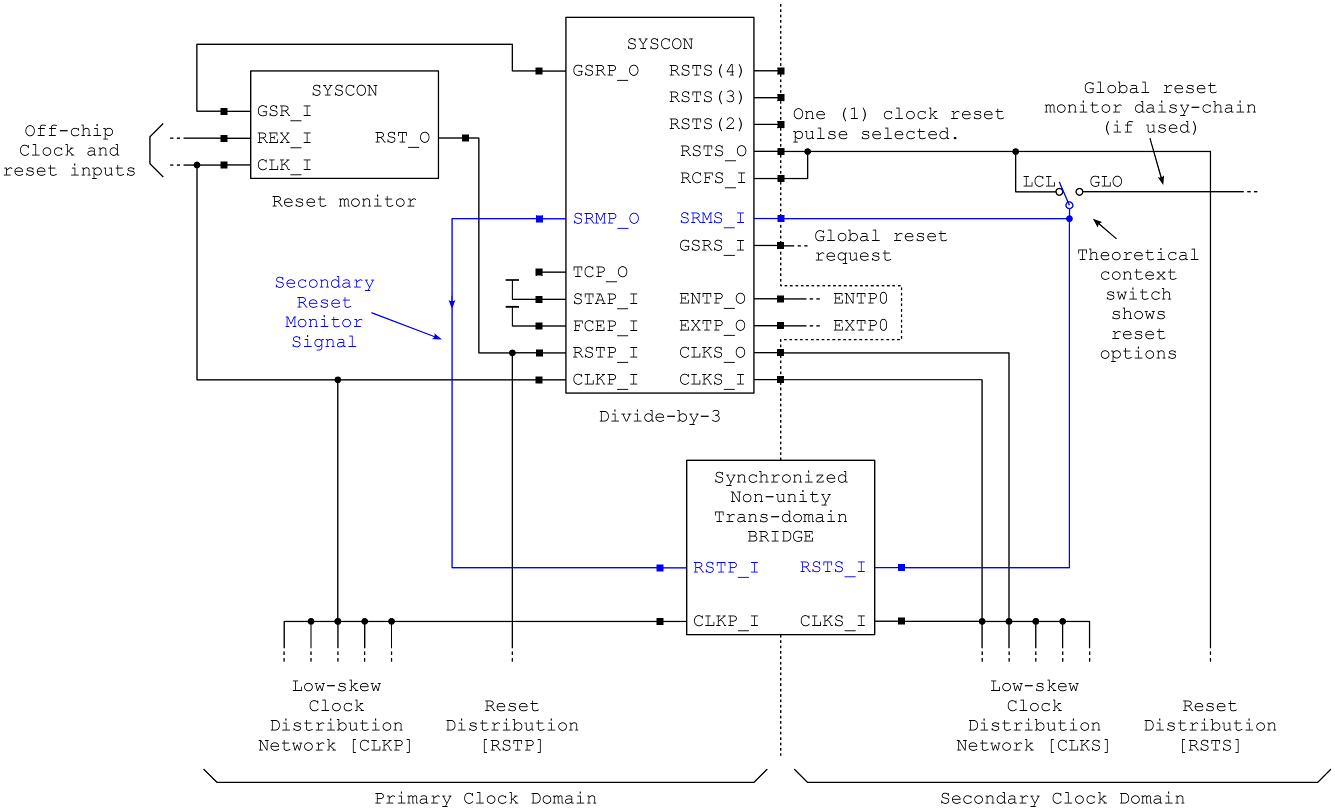
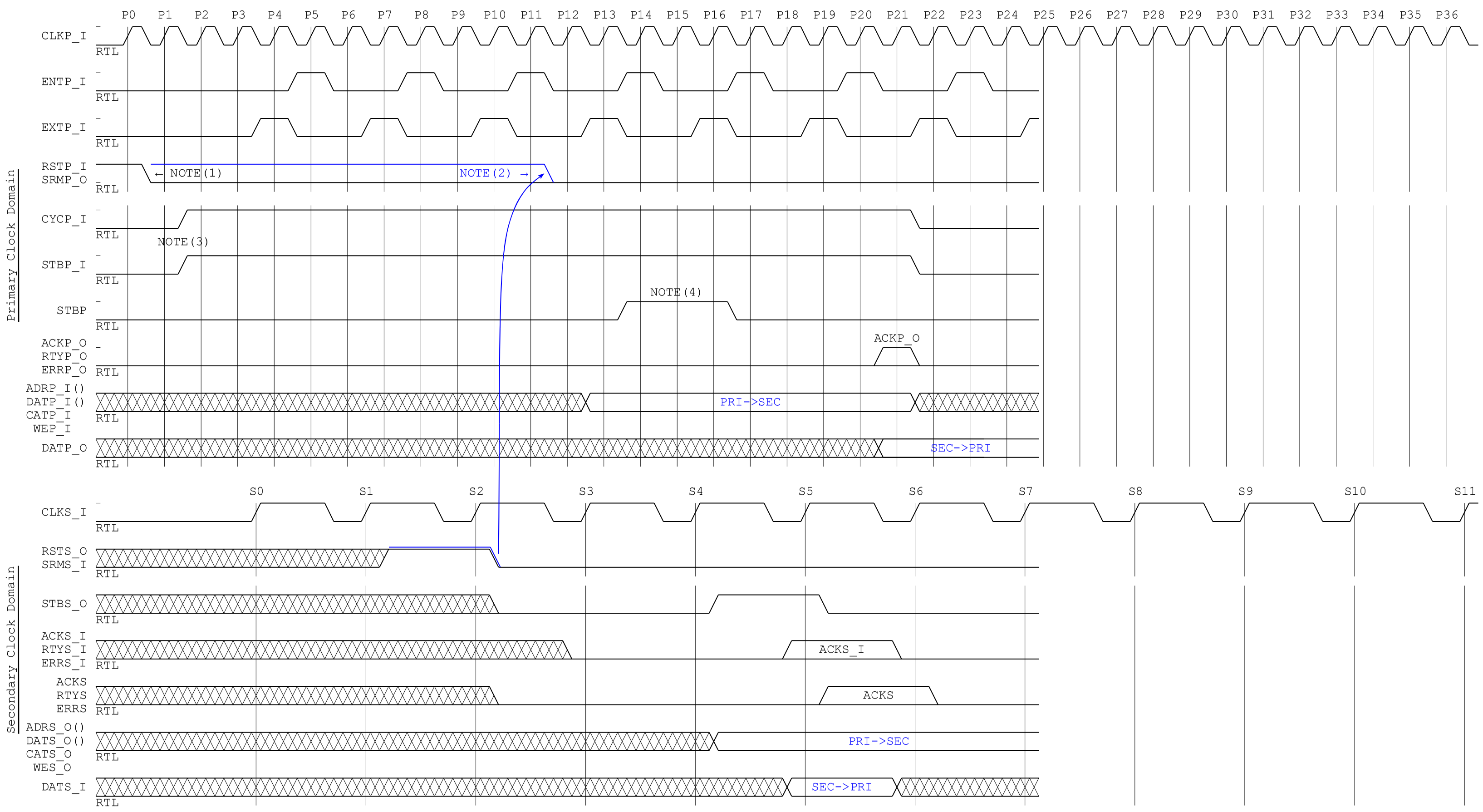

SYSCON
GSRP_O
RSTS(4)
RSTS(3)
RSTS(2)
RSTS_O
RCFS_I
SRMP_O
SRMS_I
GSRS_I
TCP_O
STAP_I
FCEP_I
RSTP_I
CLKP_I
ENTP_O
EXTP_O
CLKS_O
CLKS_I
Divide-by-3
SYSCON
GSR_I
REX_I
CLK_I
RST_O
Reset monitor
Off-chip Clock and reset inputs
One (1) clock reset pulse selected.
Global reset monitor daisy-chain (if used)
LCL
GLO
Global reset request
ENTP0
EXTP0
Theoretical context switch shows reset options
Secondary Reset Monitor Signal
Synchronized Non-unity Trans-domain BRIDGE
RSTP_I
RSTS_I
CLKP_I
CLKS_I
Low-skew Clock Distribution Network [CLKP]
Reset Distribution [RSTP]
Low-skew Clock Distribution Network [CLKS]
Reset Distribution [RSTS]
Primary Clock Domain
Secondary Clock Domain

Figure 42. Recommended reset practices for the synchronized non-unity trans-domain BRIDGE.

Synchronized Non-unity Trans-domain BRIDGE, Timing Diagram, PRI->SEC, Unidir. (3:1)



Notes:
(1) When the secondary reset monitor output signal [SRMP_O] is used, the primary system reset signal [RSTP_I] must be at least two clock cycles wide to satisfy reset blocking requirements.
(2) Secondary reset monitor output signal [SRMP_O] is negated at clock edge P11.
(3) Remote MASTER starts a cycle after primary reset signal [RSTP] is negated.
(4) BRIDGE begins PRI->SEC cycle after secondary reset monitor signal [SRMP_O] is negated.

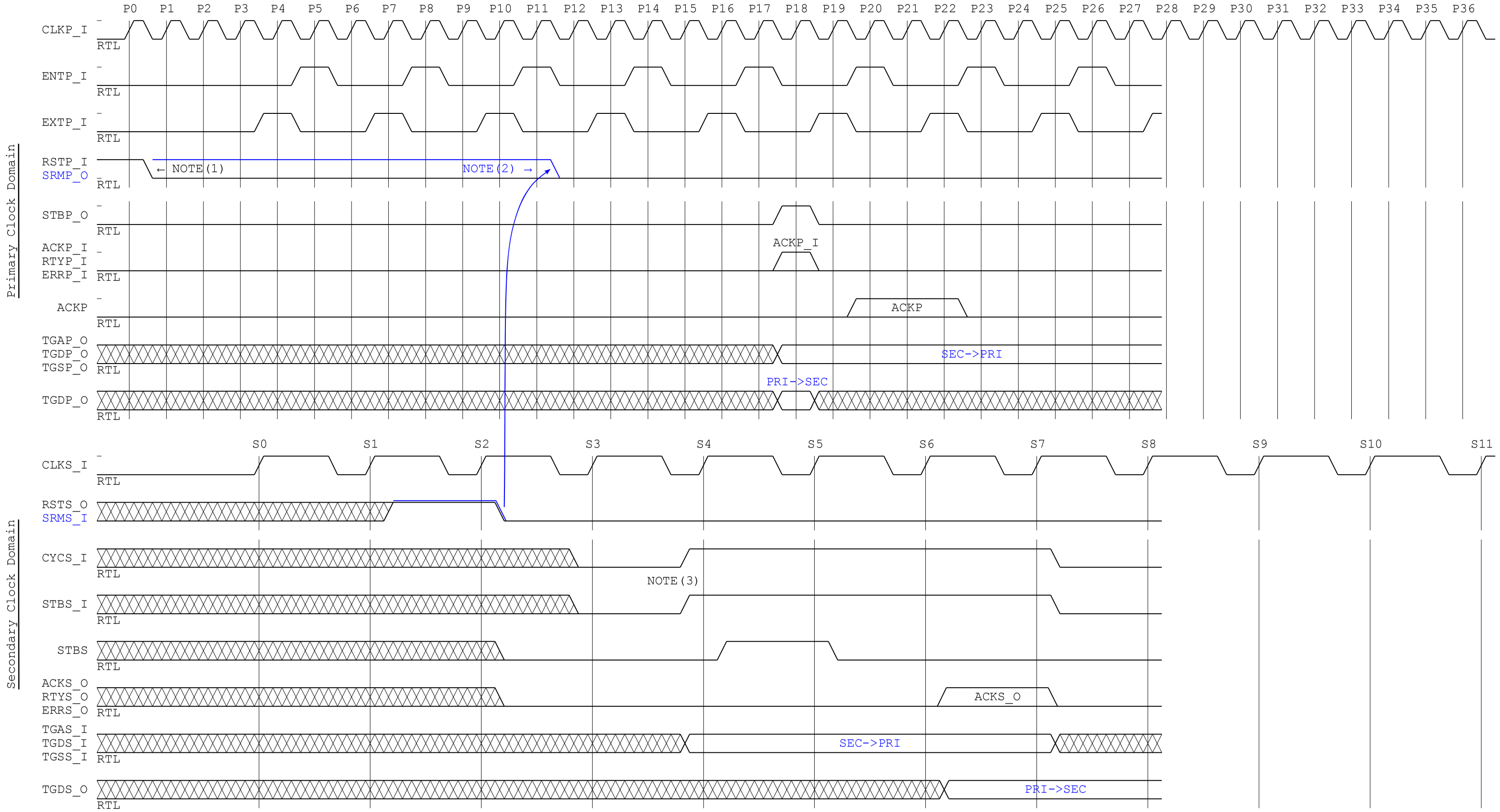
This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). MnStat 325C.01 Subd. 5(ii)

WADE D. PETERSON, SILICORE CORPORATION - REV: 26 MAR 2012

Figure 43. Reset timing with PRI->SEC cycles for a synchronized non-unity trans-domain BRIDGE.

Synchronized Non-unity Trans-domain BRIDGE, Timing Diagram, SEC->PRI, Unidir. (3:1)

(Application Note)



Notes:
(1) When the secondary reset monitor output signal [SRMP_O] is used, the primary system reset signal [RSTP_I] must be at least two clock cycles wide to satisfy reset blocking requirements.
(2) Secondary reset monitor output signal [SRMP_O] is negated at clock edge P11.
(3) Remote MASTER starts a cycle after secondary reset signal [RSTS] is negated.

This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Subd. 5(ii)*

WADE D. PETERSON, SILICORE CORPORATION - REV: 18 MAR 2012

Figure 44. Reset timing with SEC->PRI cycles for a synchronized non-unity trans-domain BRIDGE.

Performance Benchmarks for the Synchronized Non-unity Trans-domain BRIDGE

Figure 45 is a synopsis of performance benchmarks for the synchronized non-unity trans-domain BRIDGE example. Data was acquired using the graphical technique described elsewhere in this report.

Figure 45(a) shows the closed-loop control transport clock efficiency for PRI->SEC and SEC->PRI cycles using packed and unpacked interconnections on the destination side. All graphs apply to the READ, WRITE, DUPLEX and EVENT data transport operations. Noteworthy results include:

- *Each data transfer phase is identical*. So long as the remote MASTER and SLAVE interfaces are repeatable, each phase of a multi-phase data transfer requires the same number of clock cycles to complete.

- *Unidirectional and bidirectional operations are the same*. The clock efficiency is the same regardless of whether it is a unidirectional or a bidirectional BRIDGE.

- *Overall clock efficiencies are low*. The overall clock efficiency compares data transfers across a BRIDGE versus across a point-to-point interconnection. It is useful for understanding the effects of blocking. This metric assumes a baseline reference for each point-to-point transfer of one (1) packed and two (2) unpacked clock cycles.

  A packed PRI->SEC BRIDGE transfer at a divisor ratio of 4:1 requires twelve (12) primary clock cycles to complete. Comparison to the baseline reference results in a clock efficiency of 12:1, or about 8%. An unpacked transfer requires sixteen (16) primary clock cycles, and results in a clock efficiency of 16:2, or about 12.5%.

  Note that PRI->SEC transfers, when clock efficiency is measured on the secondary side, compare favorably to the common-clock BRIDGE performance shown elsewhere in this report.

Figure 45(b) shows the single cycle transfer efficiency. This measures the time duration in which the cycle signal [CYC] is asserted on the primary and secondary sides of the BRIDGE. Only packed cycles are considered. Noteworthy results include:

- *PRI->SEC transfers exhibit syncup uncertainty*. When the remote primary MASTER begins a bus cycle, the BRIDGE must synchronize it with the secondary clock. Blue shaded areas indicates uncertainty caused by synchronization cycles (syncup) between the input strobe [STBP_I] and the synchronized trans-domain exit phase input signal [EXTP_I], where: 0 ≤ syncup ≤ Freqdiv – 1. Note that the syncup uncertainty does not occur during SEC->PRI transfers.

  Under the right circumstances, the clever design of a remote MASTER might eliminate the syncup uncertainty by coordinating its bus cycle with the exit phase signal [EXTP].

- *Effect of deadlock arbiter*. The effect of the deadlock arbiter can be measured by comparing the number of clock pulses the cycle signal [CYC] is asserted in the unidirectional BRIDGE (which does not have a deadlock arbiter) versus the bidirectional BRIDGE (which does have a deadlock arbiter).

A bidirectional BRIDGE having a divisor ratio of 4:1 requires four additional primary clock cycles to complete a single PRI->SEC data transfer. This manifests itself as a 'connect penalty' associated with the cycle signal [CYC], and equals the number of extra primary clock cycles versus the unidirectional condition.

A SEC->PRI BRIDGE transfer at a divisor ratio of 4:1 requires no additional secondary clock cycles to complete a single data transfer.

These results are consistent with the deadlock arbiter location, which is on the secondary side of the BRIDGE.

- *No disconnect penalty*. There is no 'disconnect penalty' associated with the cycle signal [CYC] in either the unidirectional or bidirectional BRIDGE topologies.

Example: Synchronized Unity Trans-domain SYSCON

The synchronized unity trans-domain SYSCON synchronizes two clock domains where a primary-to-secondary clock relationship of 1:1 exists. When this type of system controller is used, then a synchronized unity trans-domain BRIDGE is used.

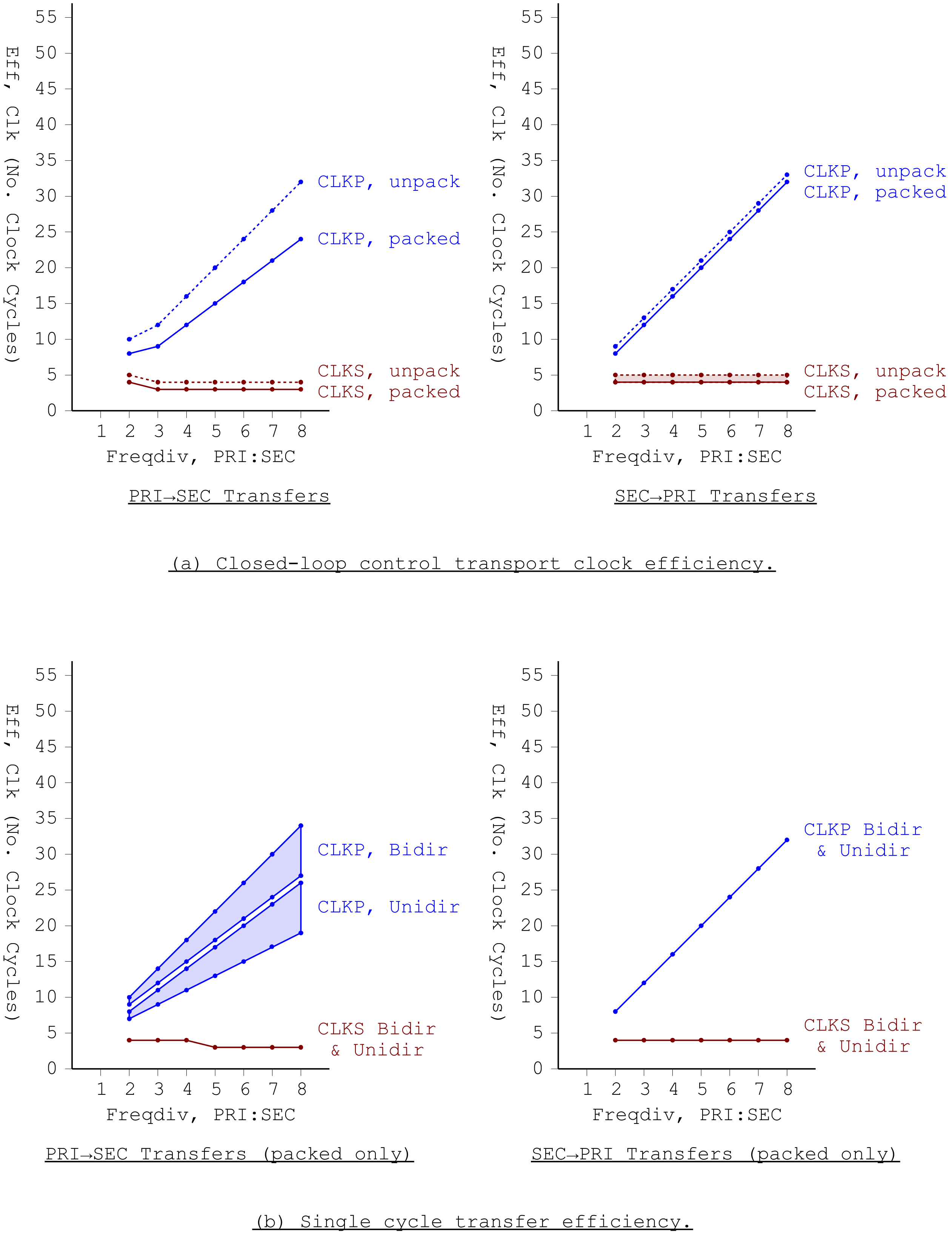
The synchronized unity trans-domain BRIDGE and SYSCON are considered to be a special cases (as compared to the non-unity variants) because they can only be used under the following conditions:

- *Clock divisor ratio*. The primary and secondary clocks always operate at the same frequency.

- *Longest common combinatorial path*. The primary and secondary LCCP ≥ + 2 ILUT. This is because of a timing relationship which is imposed on signals exiting the primary domain.

- *Clock controller*. A unit or n-step clock controller cannot be used with the unity SYSCON. This is because the controller operates a gated clock which requires precision control of the secondary clock. This can only be achieved when the clock is divided to a lower frequency (e.g. divide-by-two, divide-by-three, etc).

This SYSCON can be made as a CLASS Machine design representation. There are no special portability requirements other than requirements for two low-skew clock distribution networks, and adherence to the following four timing parameters: 'Tpd(AIS), clk-su', 'Tpd(TPD), in-out', 'Tpd(STE), clk-su' and 'Tpd(STX), clk-su'. In the SYSCON example, closed-loop $LCCP_{REL}$ = +1 ILUT, leading balance $LCCP_{REL}$ = +1 ILUT, with no lagging balance. This makes it suitable for the highest clock domain in any system.



PRI→SEC Transfers

SEC→PRI Transfers

(a) Closed-loop control transport clock efficiency.

PRI→SEC Transfers (packed only)

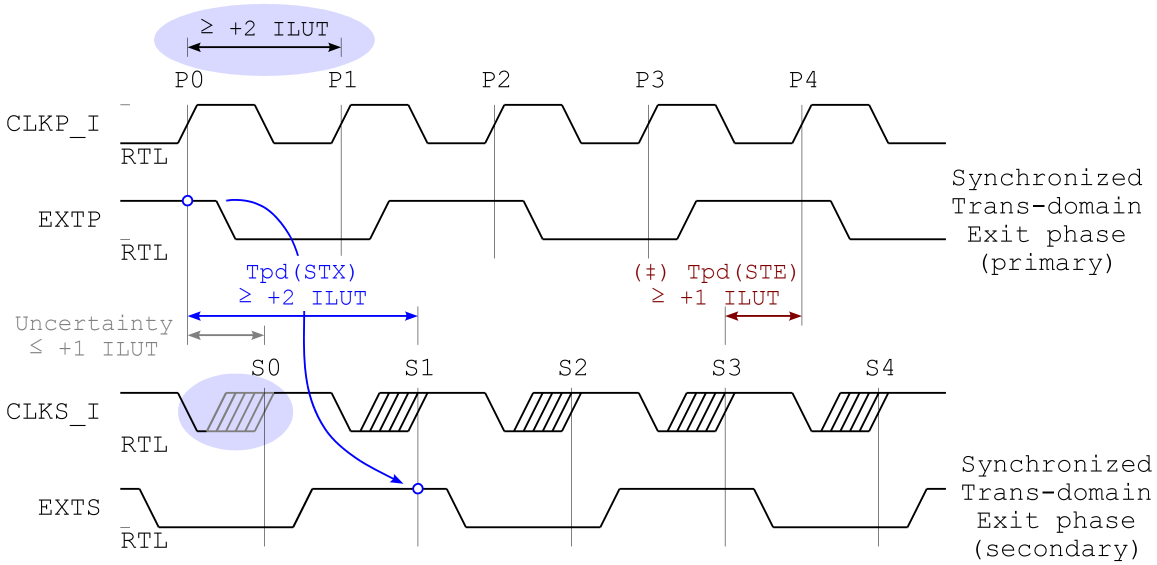
SEC→PRI Transfers (packed only)

(b) Single cycle transfer efficiency.

Figure 45. Performance benchmarks for the synchronized non-unity trans-domain BRIDGE.

Clock Phases Identify Signal Exit Points

Figure 46 shows the timing for the synchronized unity trans-domain SYSCON. There is one exit phase for the primary and the secondary domains, but there are no entry phases.



(†) Uncertainty caused by the synchronized trans-domain clock phase lag.
(‡) Secondary to primary clock edge synchronization.

Figure 46. Timing for the synchronized unity trans-domain SYSCON.

The purpose of the exit phases is to identify specific clock edges on both the primary clock [CLKP] and secondary clock [CLKS] when signals may cross the trans-domain boundary.

The entry and exit transfer delays which are associated with the entry and exit points are defined as follows:

- *Synchronized trans-domain entry transfer delay 'Tpd(STE), clk-su'*. The timing parameter that defines the longest common combinatorial path (LCCP) which is allotted for a control transfer signal entering the primary clock domain from the secondary clock domain. It is measured between identified rising edges of the primary and secondary clocks. The RTL transfer region along this path is limited to one (1) look-up table.

  When generated by a synchronized unity trans-domain SYSCON, neither the primary nor the secondary clock edges are identified by an entry phase signal; the timing parameter is measured between any sequence of a secondary clock edge [CLKS] followed by a primary clock edge [CLKP]. When expressed as relative time: $LCCP_{REL} = N \times ILUT$, where 'N' is any integer value between one and infinity (∞) and is equal to the value of both the primary and secondary clock domains, and 'ILUT' is a unit of measure established by the FASM relative timing (ILUT) model. Units of absolute time: $LCCP_{ABS} \leq$ Tpd(STE), clk-su (max) having units of measure expressed as time, such as in seconds or nanoseconds.

- *Synchronized trans-domain exit transfer delay 'Tpd(STX), clk-su'*. The timing parameter that defines the longest common combinatorial path (LCCP) which is allotted for a control transfer signal exiting the primary clock domain to the secondary clock domain. It is measured between identified rising edges of the primary and secondary clocks. The RTL transfer region along this path is limited to one (1) look-up table.

  When generated by a synchronized unity trans-domain SYSCON, the timing parameter is measured between a primary clock edge [CLKP] which is identified by the primary exit phase signal [EXTP], and a secondary clock edge [CLKS] which is identified by the secondary exit phase signal [EXTS]. When expressed as relative time: $LCCP_{REL}$ = N x ILUT, where 'N' is any integer value between one and infinity (∞) and is equal to the value of both the primary and secondary clock domains, and 'ILUT' is a unit of measure established by the FASM relative timing (ILUT) model. Units of absolute time: $LCCP_{ABS} \leq$ Tpd(STX), clk-su (max) having units of measure expressed as time, such as in seconds or nanoseconds.

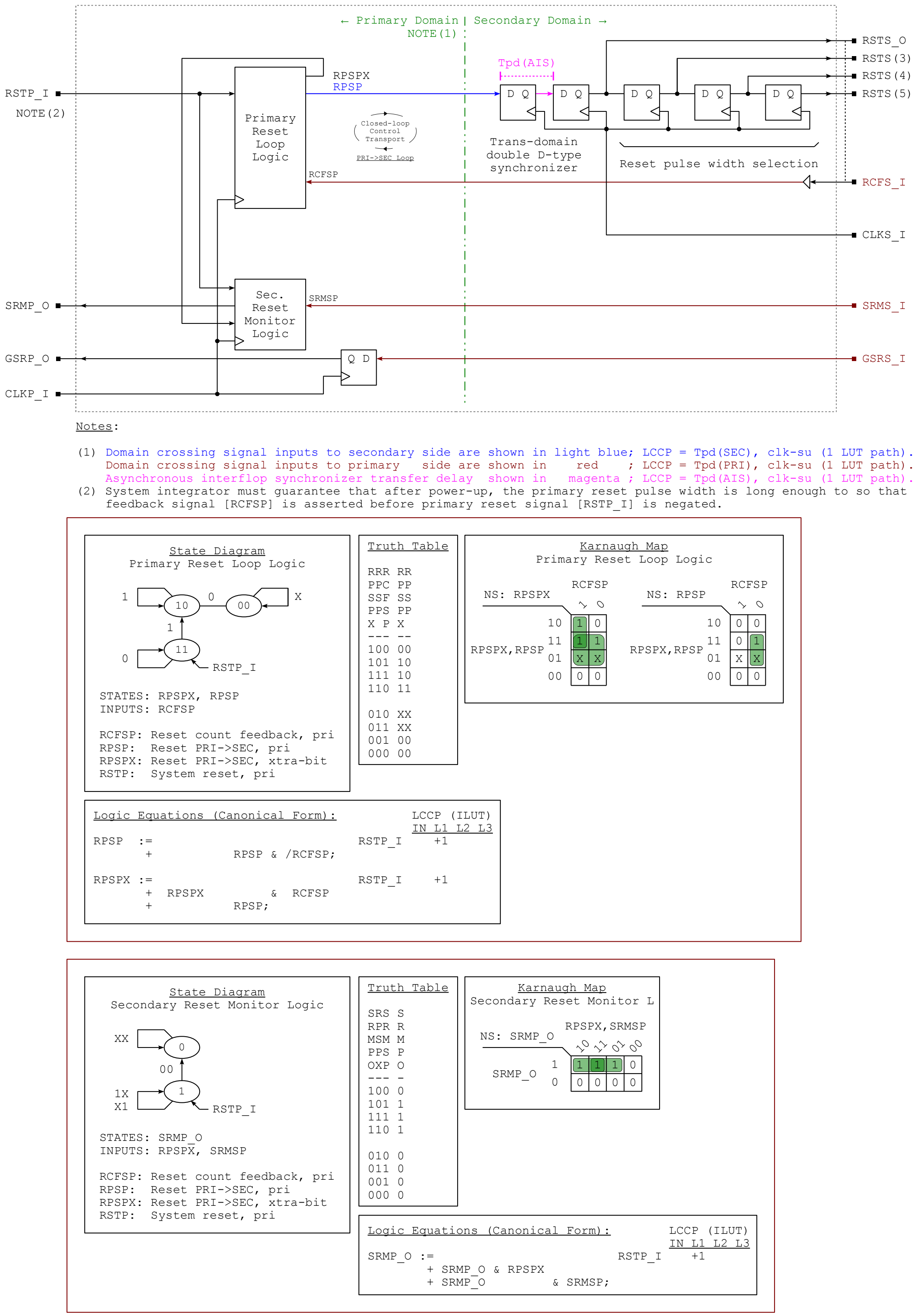
Reset Bridge

Because the timing method for the 1:1 primary-to-secondary clock arrangement requires phases in both the primary and secondary domains, it is necessary to initialize the secondary phase generator before signals are allowed to exit the primary clock domain. This reset method requires that there are no explicitly defined entry phases. A signal entering the primary clock domain may do so between any first secondary clock [CLKS] and the next primary clock [CLKP].

The use of secondary and global reset monitors signals are handled in a similar way to the approaches used in the synchronized non-unity trans-domain SYSCON describe elsewhere in this report.

Figure 47 shows the internal view of the reset bridge for this SYSCON. Because clock phases are generated in the secondary domain, the reset signal must initialize the secondary phase generator before signals may exit the primary domain. Figure 48 shows the reset timing with an initial PRI->SEC cycle for the synchronized unity trans-domain reset bridge SYSCON.

This section provides a CLASS Machine design representation for this type of SYSCON. It has no special portability requirements other than two low-skew clock distribution networks, and synchronization timing parameter Tpd(AIS), clk-su.



Notes:

(1) Domain crossing signal inputs to secondary side are shown in light blue; LCCP = Tpd(SEC), clk-su (1 LUT path).
Domain crossing signal inputs to primary side are shown in red ; LCCP = Tpd(PRI), clk-su (1 LUT path).
Asynchronous interflop synchronizer transfer delay shown in magenta ; LCCP = Tpd(AIS), clk-su (1 LUT path).

(2) System integrator must guarantee that after power-up, the primary reset pulse width is long enough to so that feedback signal [RCFSP] is asserted before primary reset signal [RSTP_I] is negated.

State Diagram
Primary Reset Loop Logic

STATES: RPSPX, RPSP
INPUTS: RCFSP

RCFSP: Reset count feedback, pri
RPSP: Reset PRI->SEC, pri
RPSPX: Reset PRI->SEC, xtra-bit
RSTP: System reset, pri

Truth Table

RPSPX	RPSP	RCFSP	RPSPX	RPSP
1	0	0	0	0
1	0	1	1	0
1	1	1	1	0
1	1	0	1	1
0	1	0	X	X
0	1	1	X	X
0	0	1	0	0
0	0	0	0	0

Karnaugh Map
Primary Reset Loop Logic

NS: RPSPX		RCFSP	
		1	0
RPSPX,RPSP	10	1	0
	11	1	1
	01	X	X
	00	0	0

NS: RPSP		RCFSP	
		1	0
RPSPX,RPSP	10	0	0
	11	0	1
	01	X	X
	00	0	0

```
Logic Equations (Canonical Form):                LCCP (ILUT)
                                                 IN  L1  L2  L3
RPSP   :=                             RSTP_I      +1
       +             RPSP  & /RCFSP;

RPSPX  :=                             RSTP_I      +1
       +  RPSPX            &  RCFSP
       +             RPSP;
```

State Diagram
Secondary Reset Monitor Logic

STATES: SRMP_O
INPUTS: RPSPX, SRMSP

RCFSP: Reset count feedback, pri
RPSP: Reset PRI->SEC, pri
RPSPX: Reset PRI->SEC, xtra-bit
RSTP: System reset, pri

Truth Table

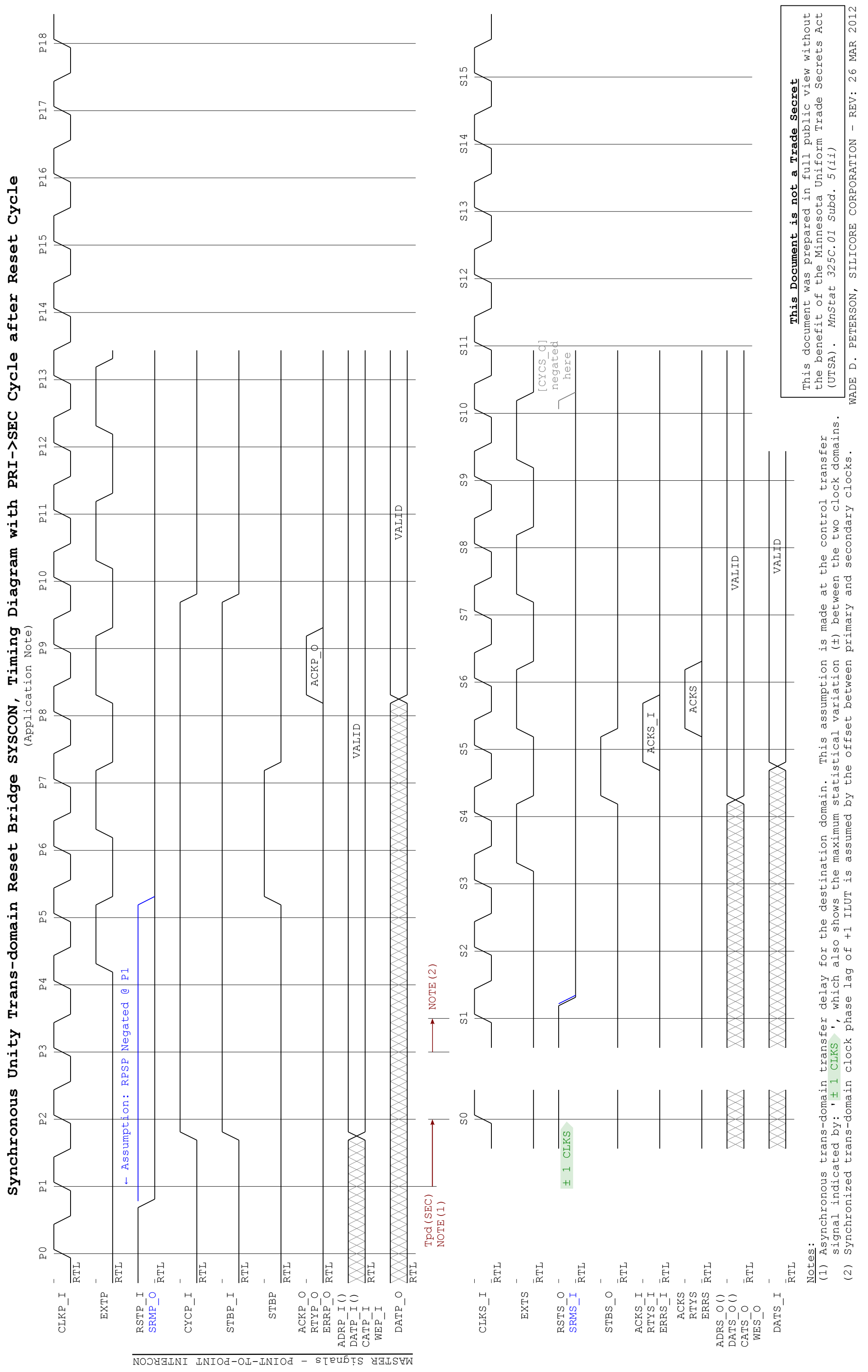
SRMPO	RPSPX	SRMSP	SRMPO
1	0	0	0
1	0	1	1
1	1	1	1
1	1	0	1
0	1	0	0
0	1	1	0
0	0	1	0
0	0	0	0

Karnaugh Map
Secondary Reset Monitor L

NS: SRMP_O		RPSPX,SRMSP			
		10	11	01	00
SRMP_O	1	1	1	1	0
	0	0	0	0	0

```
Logic Equations (Canonical Form):                LCCP (ILUT)
                                                 IN  L1  L2  L3
SRMP_O :=                                RSTP_I    +1
          + SRMP_O & RPSPX
          + SRMP_O          & SRMSP;
```

Figure 47. Internal view of the reset bridge for the synchronized unity trans-domain SYSCON.


Synchronous Unity Trans-domain Reset Bridge SYSCON, Timing Diagram with PRI->SEC Cycle after Reset Cycle
(Application Note)

Notes:
(1) Asynchronous trans-domain transfer delay for the destination domain. This assumption is made at the control transfer signal indicated by: '± 1 CLKS', which also shows the maximum statistical variation (±) between the two clock domains.
(2) Synchronized trans-domain clock phase lag of +1 ILUT is assumed by the offset between primary and secondary clocks.

This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Subd. 5(ii)*

WADE D. PETERSON, SILICORE CORPORATION - REV: 26 MAR 2012

Figure 48. Reset timing for the synchronized unity trans-domain SYSCON.

Example: Synchronized Unity Trans-domain BRIDGE

Figures 49 and 50 show performance benchmark cycles for the synchronized unity trans-domain BRIDGE. It is used when there is a primary-to-secondary clock frequency divisor ratio of 1:1.

This SYSCON can be made as a CLASS Machine design representation. There are no special portability requirements other than requirements for two low-skew clock distribution networks, and adherence to the following two timing parameters: 'Tpd(STE), clk-su' and 'Tpd(STX), clk-su'. It has a closed-loop $LCCP_{REL}$ = +2 ILUT, leading balance $LCCP_{REL}$ = +2 ILUT, with no lagging balance.

Performance Benchmarks for the Synchronized Unity Trans-domain BRIDGE

Figure 51 is a synopsis of performance benchmarks for the synchronized unity trans-domain BRIDGE example. It was acquired using the graphical technique described elsewhere in this report.

Figure 51(a) shows the closed-loop control transport clock efficiency for PRI->SEC and SEC->PRI cycles using packed and unpacked interconnections on the destination side. All graphs apply to the READ, WRITE, DUPLEX and EVENT data transport operations. Noteworthy results include:

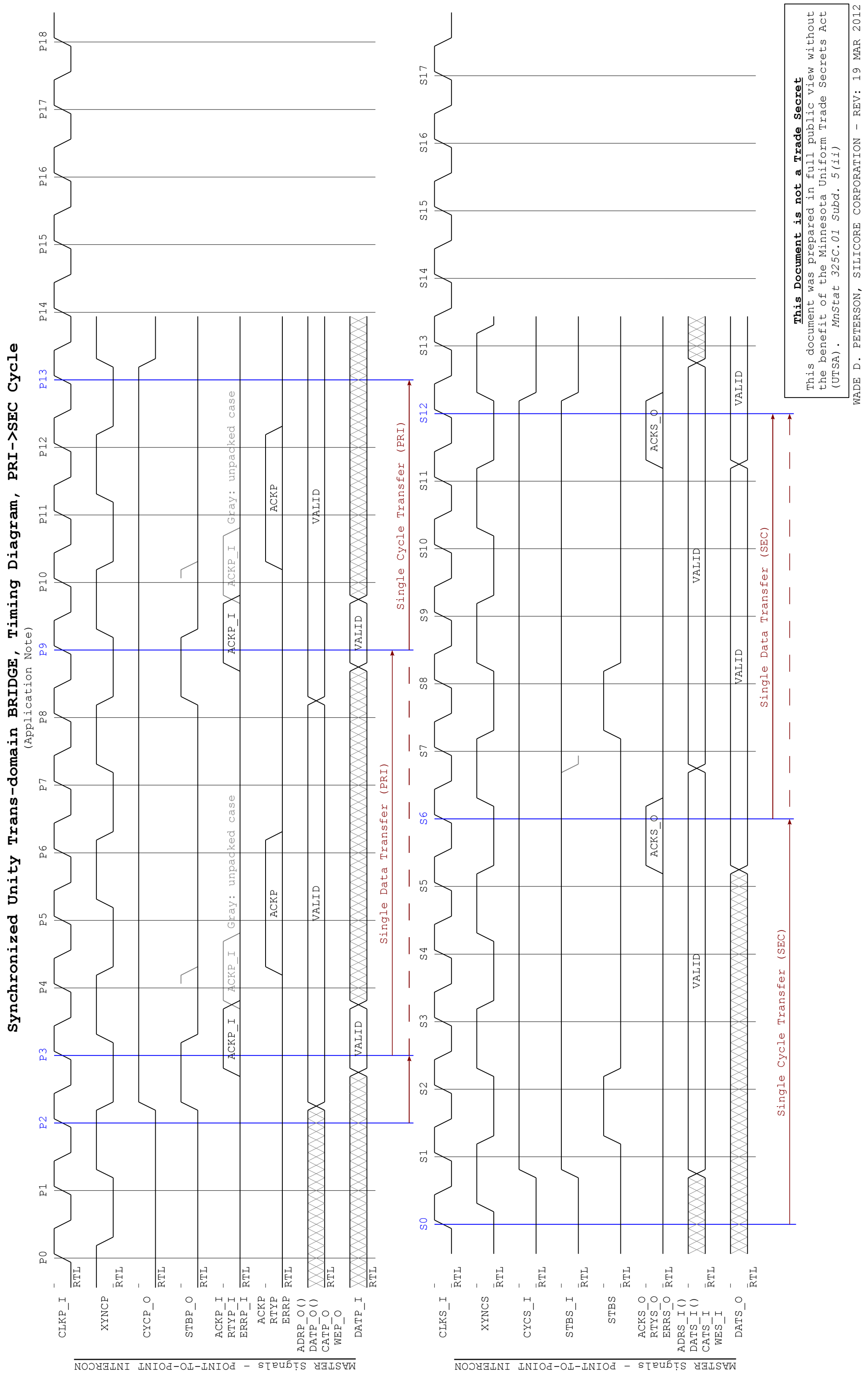
- *Each data transfer phase is identical*. So long as the remote MASTER and SLAVE interfaces are repeatable, each phase of a multi-phase data transfer requires the same number of clock cycles to complete.

- *Unidirectional and bidirectional operations are the same*. The clock efficiency is the same regardless of whether it is a unidirectional or a bidirectional BRIDGE.

- *Overall clock efficiencies are low*. The overall clock efficiency compares data transfers across a BRIDGE versus across a point-to-point interconnection. It is useful for understanding the effects of blocking. This metric assumes a baseline reference for each point-to-point transfer of one (1) packed and two (2) unpacked clock cycles.

  A packed PRI->SEC BRIDGE transfer requires six (6) primary clock cycles to complete. Comparison to the baseline reference results in a clock efficiency of 6:1, or about 17%. An unpacked transfer also requires six (6) primary clock cycles, and results in a clock efficiency of 6:2, or about 33%.

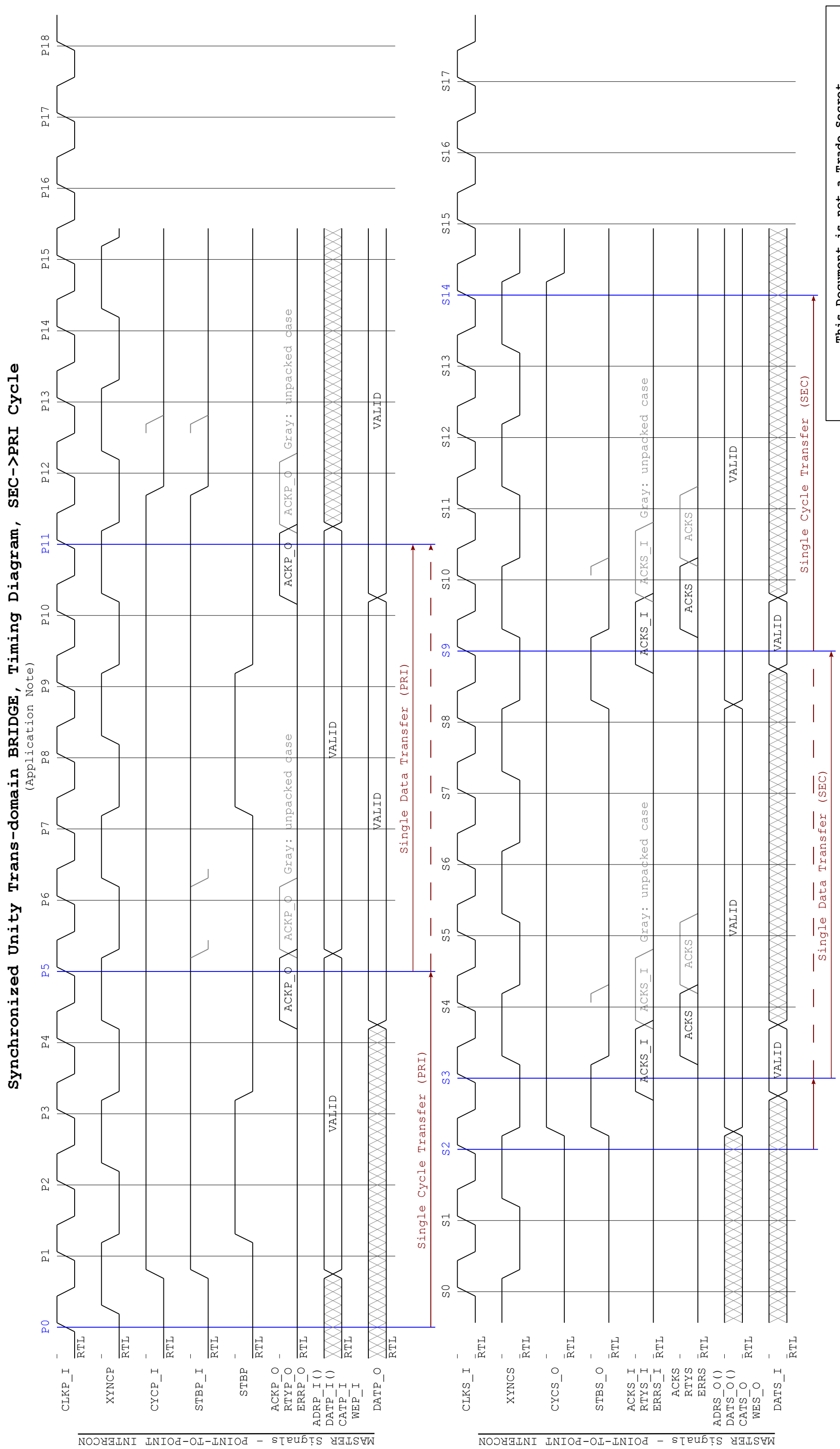
  Note that this BRIDGE is less efficient than the common clock BRIDGE, but compares favorably to it.



This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Subd. 5(ii)*

WADE D. PETERSON, SILICORE CORPORATION - REV: 19 MAR 2012

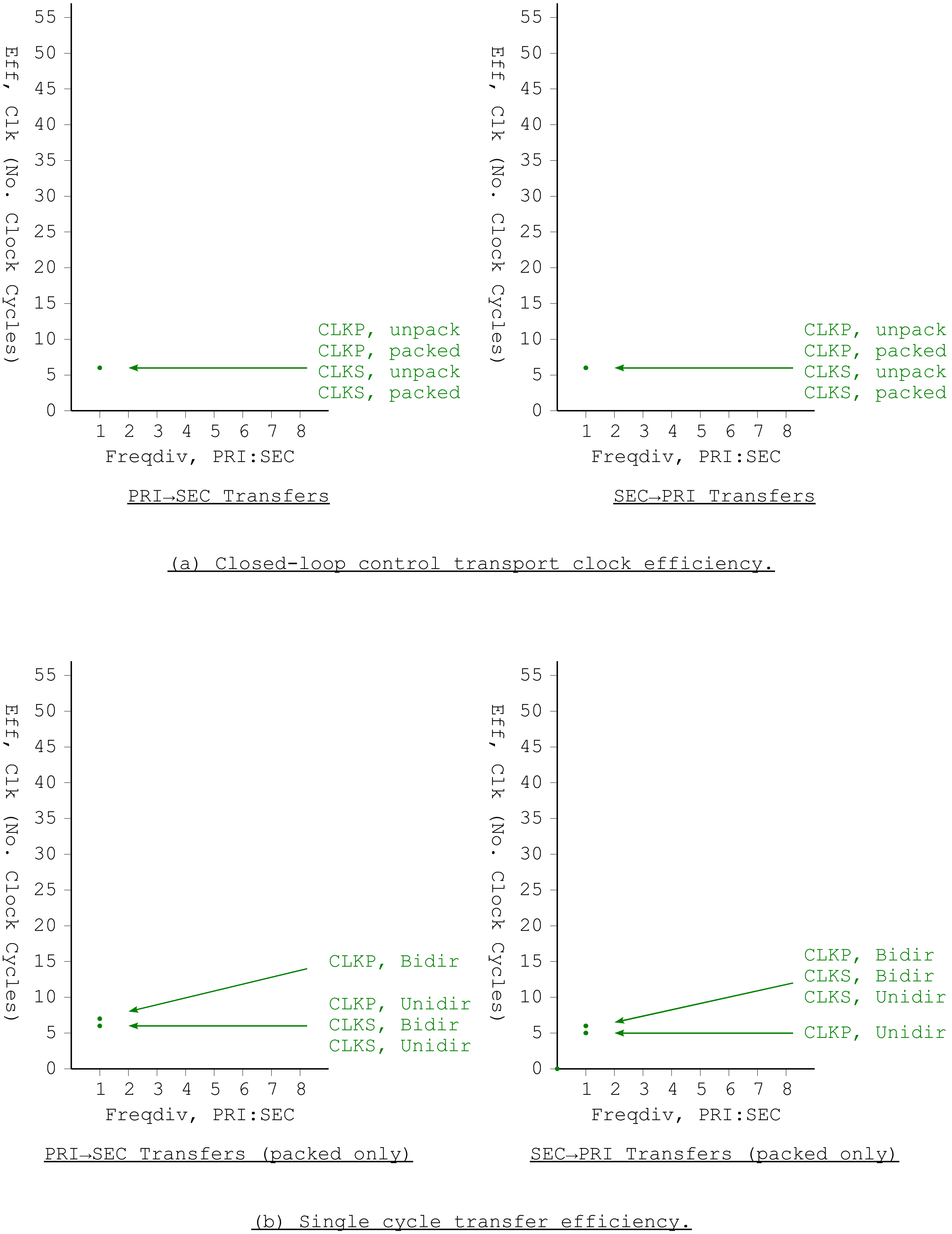
Figure 49. PRI->SEC performance benchmark for the synchronized unity trans-domain BRIDGE.



This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). *MnStat 325C.01 Subd. 5(ii)*

WADE D. PETERSON, SILICORE CORPORATION - REV: 19 MAR 2012

Figure 50. SEC->PRI performance benchmark for the synchronized unity trans-domain BRIDGE.



(a) Closed-loop control transport clock efficiency.

(b) Single cycle transfer efficiency.

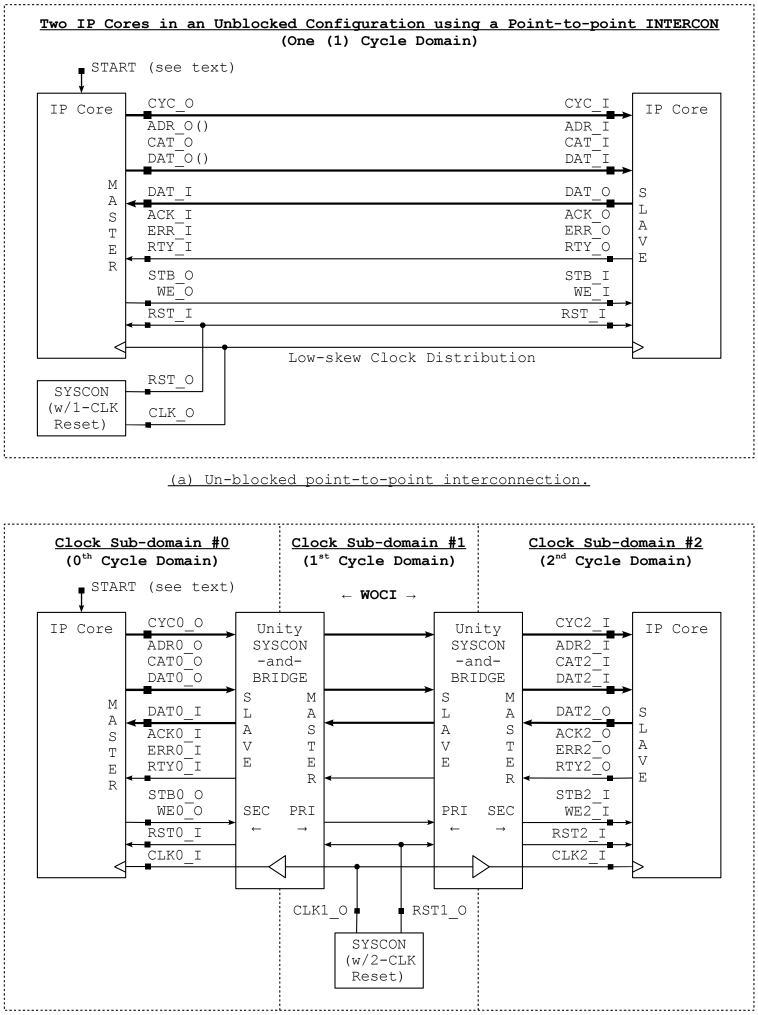
Figure 51. Performance benchmarks for the synchronized unity trans-domain BRIDGE.

Figure 51(b) shows the single cycle transfer efficiency. This measures the time duration in which the cycle signal [CYC] is asserted on the primary and secondary sides of the BRIDGE. Only packed cycles are considered. Noteworthy results include:

- *No syncup uncertainty*. Unlike the synchronized non-unity trans-domain BRIDGE, there is no syncup uncertainty.
- *The deadlock arbiter has no effect*. The deadlock arbiter does not affect clock efficiency.
- *No disconnect penalty*. There is no 'disconnect penalty' associated with the cycle signal [CYC] in either the unidirectional or bidirectional BRIDGE topologies.

Example: Blocking the WISHBONE Off-chip Component Interconnection (WOCI)

Figure 52 shows a blocking example for the WISHBONE Off-chip Component Interconnection (WOCI).



(a) Un-blocked point-to-point interconnection.

(b) Blocked point-to-point interconnection for the WOCI.

Figure 52. Blocking example for the WISHBONE Off-chip Component Interconnection (WOCI).

Chapter 4 - The Asynchronous Trans-domain BRIDGE

The asynchronous trans-domain BRIDGE allows the transport of WISHBONE DATUM INTERLOCKED bus cycles between two clock domains that operate from two independent clock sources[43]. This allows different parts of a value-added data process to operate at different clock frequencies.

Before launching into the asynchronous BRIDGE case, it is useful to review the concepts of metastability and the synchronization of signals.

Metastability and the Synchronization of Signals

Metastability refers to an unstable condition which occurs in a D-type flip-flop[44] output when an input signal violates its set-up and hold time (with respect to the system clock). This condition usually happens when a signal is first synchronized to a clock domain.

In portable CLASS Machine design representations, the effects of the metastable state be eliminated with a trans-domain double D-type synchronizer. However, this technique is only applied to critical control transport signals – those where the set-up and hold time of a flip-flop is likely to be violated. These signals, once synchronized, can be used to extend the domain crossing time for other non-critical signals, thereby reducing the total logic size. The critical and non-critical domain crossing signals are readily identified by inspection of the timing diagrams.

Figure 53(a) shows how metastability might effect an RTL segment when the setup and hold times of a flip-flop are violated. At the left side of the upper diagram the setup and hold times are met, and so a stable (monotonic) output results at flip-flop Q1. At the right side (shown in light blue) the setup and hold times are violated, which may result in a metastable condition.

The duration of the metastable condition is statistical, meaning that sometimes it resembles a high speed glitch, a chaotic 'fuzz' or a periodic high speed periodic oscillation. The actual response is dependent upon the semiconductor technology, the output load conditions, the operating conditions, the amount of power supply noise[45] and possibly other criteria. Eventually the flip-flop stabilizes itself to a high or low condition, but the final state is based upon statistical chance.

The duration of the metastable state is proportional to the astable frequency (internal closed-loop toggle frequency) of the flip-flop, and so it's a function of the target device technology. In asynchronous BRIDGE applications this duration is assumed to be less than the period of one system clock cycle. This makes it non-viable as a switching signal because the output of the flip-flop is not stable for the entire period of the clock cycle[46].

[43] Unless they are justified under the rules of the WISHBONE specification, two clock domains which are synchronized by a phase-lock loop are also considered to be coupled by asynchronous means.

[44] This report concerns itself with the D-type flip-flop, but all types of flip-flop exhibit metastability.

[45] Metastability is one of the few times where power supply noise yields a positive result. Once the metastable condition starts, power supply noise can force the flip-flop output into its high or low state, thus diminishing the effects of the problem.

[46] The duration of the metastable condition is statistical, meaning that under the right set of circumstances it could last

Figure 53(b) shows a solution based upon a double D-type synchronizer. This is a double register that always provides a stable (monotonic) output at Q2. It always resides along a domain-crossing signal path in the destination domain.

Trans-domain Double D-type Synchronizer

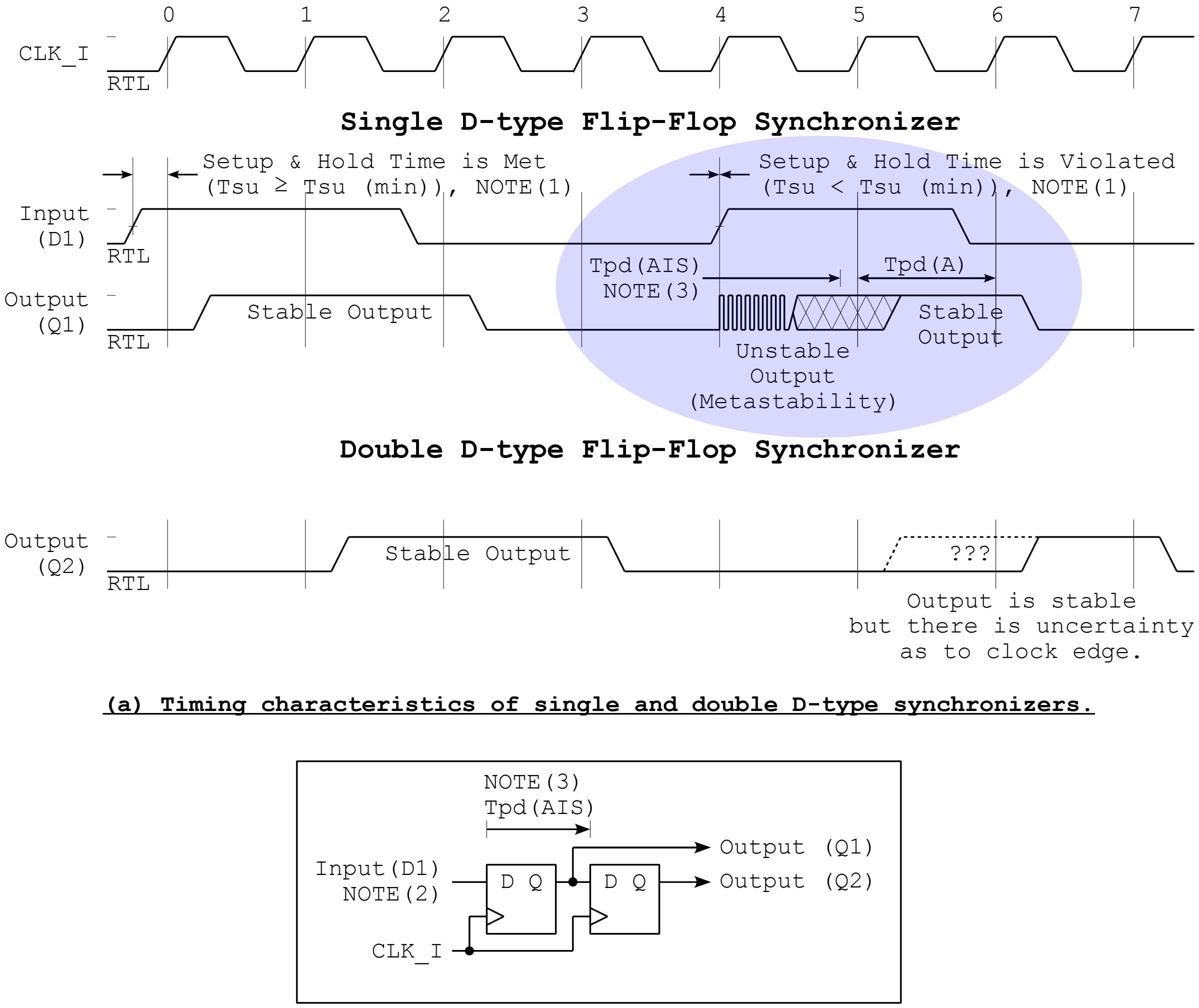
Figure 54 shows a CLASS Machine design representation for the trans-domain double D-type synchronizer. It is a method for controlling synchronization, timing and metastability in a signal that crosses an asynchronous trans-domain boundary. It is generally applied to a critical timing path within a control transport structure. It uses three flip-flops which are arranged in series along the signal path: an exit flip-flop located in the first domain and two synchronization flip-flops located in the second domain.

For example, a signal crossing a trans-domain boundary from a primary to a secondary clock domain is first registered by a flip-flop in the primary domain. This serves as the terminus to the clock sub-domain in the primary domain. The signal then crosses the trans-domain boundary and is registered by a first synchronization flip-flop. Timing in this segment is controlled by the trans-domain transfer delay. Because the set-up and hold times of the first synchronization flip-flop cannot be controlled, it is subject to metastable conditions. These are filtered by a second synchronization flip-flop. Timing between the between the first and second synchronization flip-flops is controlled by the asynchronous interflop synchronizer transfer delay 'Tpd(AIS), clk-su'. The last flip-flop also serves as a terminus to the clock sub-domain located in the secondary domain.

Asynchronous Interflop Synchronizer Transfer Delay 'Tpd(AIS), clk-su'

The asynchronous interflop synchronizer transfer delay 'Tpd(AIS), clk-su' is a timing parameter that defines the longest common combinatorial path (LCCP) between the two final flip-flops in a trans-domain double D-type synchronizer. The purpose of this delay is to minimize the effects of metastability by causing a router to minimize the distance between these two synchronizer flip-flops, and to maximize the stabilization time during a metastable event. The RTL transfer region along this path is limited to one (1) look-up table. When expressed as relative time: $LCCP_{REL}$ = +1 ILUT, where 'ILUT' is a unit of measure established by the FASM relative timing (ILUT) model. Units of absolute time: $LCCP_{ABS} \leq$ Tpd(AIS), clk-su (max) having units of measure expressed as time, such as in seconds or nanoseconds.

more than one clock cycle. However, if the closed-loop control transport structures shown this report are used, the asynchronous BRIDGE should tolerate this rare situation, albeit at a temporary reduction of data transfer speed.



(a) Timing characteristics of single and double D-type synchronizers.

(b) Double D-type sychronizer circuit

NOTEs:
(1) 'Tsu (min)' refers to the minimum setup & hold time specified by the target device manufacturer.

(2) Setup & hold timing parameter 'Tsu (min)' is violated on a statistical basis when D-type flip-flop inputs are operated asynchronously with respect to the clock input. As Tsu (min) approaches zero, the relative frequency of occurrence and duration of the metastable output conditions becomes less severe, but they are never eliminated entirely.

(3) Asynchronous interflop synchronizer transfer delay (Tpd(AIS)) is applied to the path between the two flip-flops.

Figure 53. How metastability might effect an RTL segment.

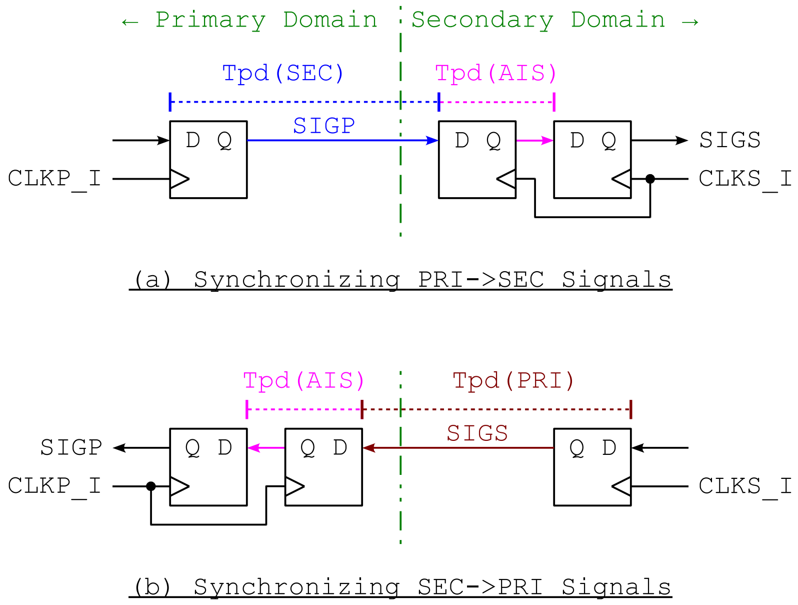


Figure 54. Class Machine design representation for the trans-domain double D-type synchronizer.

Trans-domain Transfer Delay

The trans-domain transfer delay is the timing parameter along a signal path between a source register and a destination register which are located in two different clock domains. Unless otherwise indicated, the value depends upon the direction of signal travel. The RTL transfer region along this path is limited to one (1) look-up table. It is applied in different ways depending upon the context:

- At a *synchronized trans-domain boundary* it equals one of the following: (i) the destination clock domain transfer delay 'Tpd(DOM), clk-su'[47], (ii) the synchronized trans-domain entry transfer delay 'Tpd(STE), clk-su)' or (iii) the synchronized trans-domain exit transfer delay 'Tpd(STX), clk-su'.

- At an *asynchronous trans-domain boundary* it equals that of the destination clock domain transfer delay 'Tpd(DOM), clk-su'.

Estimating Total Synchronization Time

The total synchronization time – the amount of time required to synchronize a signal through a trans-

[47] ***clock domain transfer delay 'Tpd(DOM), clk-su'***
The timing parameter that defines the longest common combinatorial path (LCCP), and the minimum duration of each periodic clock cycle, in a clock domain region. If multiple clock domains exist, then each will have its own unique specification. When expressed as relative time: $LCCP_{REL}$ = N x ILUT, where 'N' is any integer value between one and infinity (∞) and 'ILUT' is a unit of measure established by the FASM relative timing (ILUT) model. When expressed as actual time: $LCCP_{ABS} \leq$ Tpd(DOM),clk-su (max) having units of measure expressed as time, such as in seconds or nanoseconds.

domain double D-type synchronizer – is found by adding three values: (i) the routed delay between the source domain flip-flop and the first destination domain flip flop; (ii) the time required to synchronize the signal to a new clock edge; and (iii) a one clock delay between the destination flip-flops in the double D-type synchronizer:

Total synchronization time = routed delay + synchronization delay + 1 Clk

At an asynchronous trans-domain boundary the two clocks are asynchronous, and so the total time is statistical. This means that it takes on a minimum (best case) value, a typical (mean) value and a maximum (worst case) value.

The routed delay varies between some infinitesimally small value, and a maximum of one (1) destination propagation delay 'Tpd(DOM), clk-su'. It is assumed that the route will exhibit either a flat or Gaussian (normal distribution) between minimum and maximum values, across a range of target devices and routing runs with different seed numbers.

The synchronization delay is the time required to synchronize the signal to a new clock edge. It is exclusive of the routing delay, and depends upon the exact phase relationship between the primary and secondary clocks at the time the trans-domain signal is launched. The minimum will be one (1.0) destination propagation delay, the average will be one and one-half (1.5) destination propagation delays, and the maximum will be two (2.0) destination propagation delays. Over a sufficiently long time interval it is assumed that this will exhibit a 'flat' probability density function, meaning that any particular run will produce an evenly distributed delay between the minimum and maximum values.

The total trans-domain transfer delay is the sum of the routed delay, the synchronization delay and the one (1) clock double D-type synchronizer delay. Thus, the total minimum will be two (2.0) destination propagation delays, the average will be two and one-half (2.5) destination propagation delays, and the maximum will be three (3.0) destination propagation delays. Over a sufficiently long time interval it is assumed that this will also exhibit a 'flat' probability density function, meaning that any particular run will produce an evenly distributed delay between the minimum and maximum values.

Example: Asynchronous Reset Bridge SYSCON

The asynchronous reset bridge SYSCON is a WISHBONE functional module for re-transmitting the system reset signal [RST] across a clock domain boundary. Since the clock sources for the two clock domains are independent, this SYSCON only handles the system reset signals. It is used in conjunction with an asynchronous BRIDGE or REPEATER functional modules.

The primary side of the reset bridge SYSCON will always assert or negate the system reset input signal [RSTP_I] before the secondary side [RSTS_O] does. It can be used with any combination of clock frequencies, which means that there can be a considerable delay between the negation of the primary reset input signal [RSTP_I], and the assertion of the secondary reset input signal [RSTS_I]. Until the secondary side is reset the domain crossing signals from that side are assumed to take a transitory (non-deterministic) state, which could result in an erratic state of logic on the primary side. Further complications might arise when a remote MASTER on the primary side of the bridge begins a bus cycle before the secondary side is initialized.

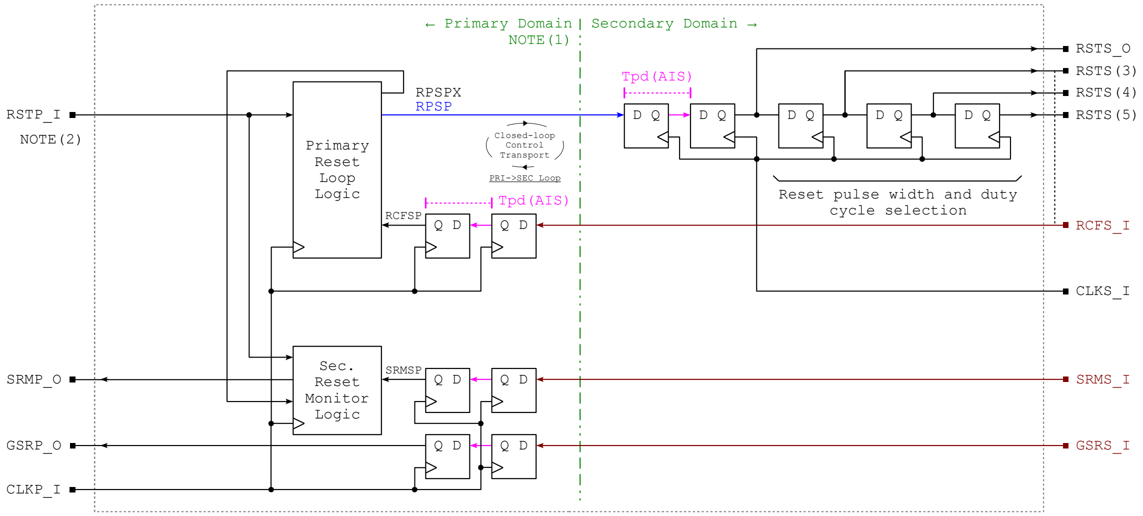
These problems are overcome by secondary and global reset monitors similar to those employed in the synchronized non-unity trans-domain SYSCON described above. They provide the following solutions:

- *Staged reset*. The reset signals on both sides of the BRIDGE cannot be negated at the same time, regardless of the clock frequencies used. However, within a hierarchy of domain authority the resets on the two sides are staged in a predictable manner to prevent transitory states from producing erratic behavior on the two WISHBONE interfaces.

- *Early remote MASTER cycle*. Bus cycles from remote MASTERs can take place immediately after a local (primary or secondary) reset signal is negated, but are should be ignored by the BRIDGE SLAVE interfaces until the secondary reset monitor signals are negated.

This SYSCON can be made as a CLASS Machine design representation. There are no special portability requirements other than two low-skew clock distribution networks, and adherence to the following transfer delays at an asynchronous trans-domain boundary: 'Tpd(AIS), clk-su' and the destination clock domain transfer delay 'Tpd(DOM), clk-su'. In the SYSCON example, closed-loop $LCCP_{REL}$ = +1 ILUT, leading balance $LCCP_{REL}$ = +1 ILUT, with no lagging balance. This makes it suitable for the highest clock domain in any system.

Figure 55 shows the schematic and logic diagrams for the asynchronous reset bridge SYSCON. Figure 56 shows reset timing diagrams for combined PRI->SEC and SEC->PRI cycles in a bidirectional BRIDGE, at a clock divisor ratio of about 3:1.



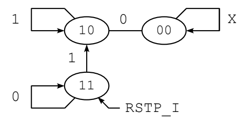
Notes:

(1) Domain crossing signal inputs to secondary side are shown in light blue; LCCP = Tpd(SEC), clk-su (1 LUT path).
Domain crossing signal inputs to primary side are shown in red ; LCCP = Tpd(PRI), clk-su (1 LUT path).
Asynchronous interflop synchronizer transfer delay shown in magenta ; LCCP = Tpd(AIS), clk-su (1 LUT path).

(2) System integrator must guarantee that after power-up, the primary reset pulse width is long enough to so that feedback signal [RCFSP] is asserted before primary reset signal [RSTP_I] is negated.

State Diagram

Primary Reset Loop Logic



STATES: RPSPX, RPSP
INPUTS: RCFSP

RCFSP: Reset count feedback, pri
RPSP: Reset PRI->SEC, pri
RPSPX: Reset PRI->SEC, xtra-bit
RSTP: System reset, pri

Truth Table

RRR	RR
PPC	PP
SSF	SS
PPS	PP
X P	X
---	--
100	00
101	10
111	10
110	11
010	XX
011	XX
001	00
000	00

Karnaugh Map

Primary Reset Loop Logic

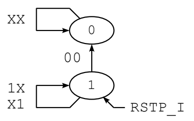
NS: RPSPX		RCFSP 1	0
RPSPX,RPSP	10	1	0
	11	1	1
	01	X	X
	00	0	0

NS: RPSP		RCFSP 1	0
RPSPX,RPSP	10	0	0
	11	0	1
	01	X	X
	00	0	0

```
Logic Equations (Canonical Form):                LCCP (ILUT)
                                                 IN L1 L2 L3
RPSP  :=                              RSTP_I     +1
       +           RPSP & /RCFSP;

RPSPX :=                              RSTP_I     +1
       +  RPSPX          &  RCFSP
       +           RPSP;
```

State Diagram

Secondary Reset Monitor Logic



STATES: SRMP_O
INPUTS: RPSPX, SRMSP

RCFSP: Reset count feedback, pri
RPSP: Reset PRI->SEC, pri
RPSPX: Reset PRI->SEC, xtra-bit
RSTP: System reset, pri

Truth Table

SRS	S
RPR	R
MSM	M
PPS	P
OXP	O
---	-
100	0
101	1
111	1
110	1
010	0
011	0
001	0
000	0

Karnaugh Map

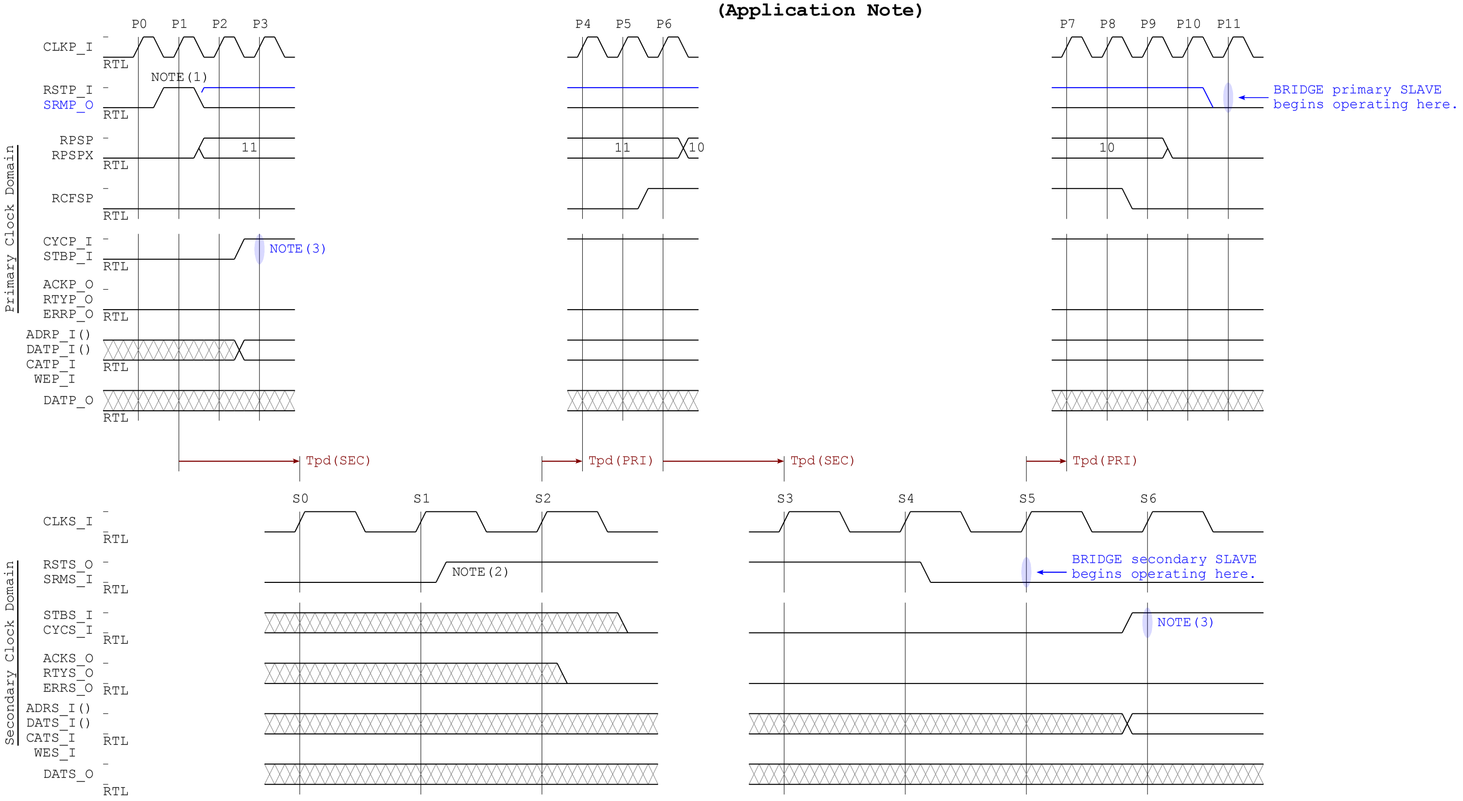
Secondary Reset Monitor L

NS: SRMP_O		RPSPX,SRMSP 10	11	01	00
SRMP_O	1	1	1	1	0
	0	0	0	0	0

```
Logic Equations (Canonical Form):           LCCP (ILUT)
                                            IN L1 L2 L3
SRMP_O :=                            RSTP_I    +1
          + SRMP_O & RPSPX
          + SRMP_O          & SRMSP;
```

Figure 55. Schematic and logic diagrams for the asynchronous reset bridge SYSCON.

Asynchronous Reset Bridge SYSCON, Timing Diagram, PRI->SEC, Unidir. (3:1) (Application Note)



Notes:
(1) Timing diagram assumes that the primary system reset input [RSTP_I] has been asserted previously in response to a power-up reset. At that time the system integrator guaranteed that (after power-up) the primary reset pulse width was sufficiently long to guarantee that feedback signal [RCFSP] had been asserted before primary reset [RSTP_I] was negated.
(2) The timing diagram assumes that feedback signal [RCFS_I] is tied to [RSTS(3)]. This guarantees that a BRIDGE is reset for at least one clock pulse before the feedback [RCFS_I] is returned to the primary side.
(3) Remote MASTER starts a cycle after local reset signal is negated.

This Document is not a Trade Secret
This document was prepared in full public view without the benefit of the Minnesota Uniform Trade Secrets Act (UTSA). MnStat 325C.01 Subd. 5(ii)

WADE D. PETERSON, SILICORE CORPORATION - REV: 26 MAR 2012

Figure 56. Timing diagram for the asynchronous reset bridge SYSCON.

Example: Asynchronous Trans-domain BRIDGE

An asynchronous trans-domain BRIDGE connects two clock domain regions where the clocks are not synchronized.

It is an automatic BRIDGE, meaning that it responds automatically to bus cycles presented to its SLAVE interface.

It is a bidirectional BRIDGE, meaning it has two MASTER and two SLAVE interfaces in opposition which propagate bus cycles in two directions. SLAVE interfaces on a bidirectional BRIDGE will also resolve a deadly embrace by asserting the retry output signal [RTY_O]. However, it can also be configured as a unidirectional BRIDGE through the use of a minimization exclusion.

The BRIDGE example supports all combinations of DATUM INTERLOCKED cycles, which include:

```
CONGA{
        DATUM INTERLOCKED( READ() ),
        DATUM INTERLOCKED( WRITE() ),
        DATUM INTERLOCKED( DUPLEX() ),
        DATUM INTERLOCKED( EVENT() )
};
```

This section provides a CLASS Machine design representation for this type of BRIDGE. There are no special portability requirements other than requirements for two low-skew clock distribution networks, and adherence to the following transfer delays at an asynchronous trans-domain boundary: 'Tpd(AIS), clk-su' and the destination clock domain transfer delay 'Tpd(DOM), clk-su'. It has a closed-loop $LCCP_{REL}$ = +2 ILUT, leading balance $LCCP_{REL}$ = +2 ILUT, with no lagging balance.

<u>Closed-loop Control Transport Structure</u>

Figure 57 shows the general topology of the closed-loop control transport structure. Logic equations and timing diagrams are shown in Figures 58-64.

The closed-loop control transport structure for a BRIDGE always begins at the strobe output signal [STB_O] of a remote MASTER interface, and ends with the terminal signal (e.g. [ACK_I]) at the same MASTER. The detailed operating description which follows is for a primary-to-secondary (PRI->SEC) transaction on an automatic, unidirectional BRIDGE, where the remote interfaces are connected by point-to-point interconnections. The following steps complete the closed-loop control transport process:

1. *<u>Remote MASTER.</u>* Remote MASTER addresses BRIDGE SLAVE with a DATUM INTERLOCKED cycle, causing primary strobe input signal [STBP_I] to be asserted.

2. *<u>Primary SLAVE Cycle Logic</u>*. Primary SLAVE Cycle Logic generates a trans-domain cycle signal [CPSP] at the beginning of the CONGA cycle. All trans-domain signals are synchronized to the destination clock domain using a double D-type synchronizer to prevent metastability problems.

If a BLOCK INTERLOCKED cycle is requested (i.e. CONGA address tag [CATP_I] is asserted), then the error output signal [ERRP_O] is returned to the remote MASTER, and the remote interconnection is never requested.

3. *Primary SLAVE Loop Logic*. Primary SLAVE Loop Logic generates a trans-domain strobe signal [EPSP] at the beginning of the data transport. This signal, in combination with [EPSSS] from the far side of the BRIDGE, forms a four-edge handshaking protocol. The asserted signal propagates across the clock domain boundary, and is registered at the secondary side within the time specified by the propagation delay entry specification Tpd(SEC), clk-su, before conditioning by a double D-type synchronizer.

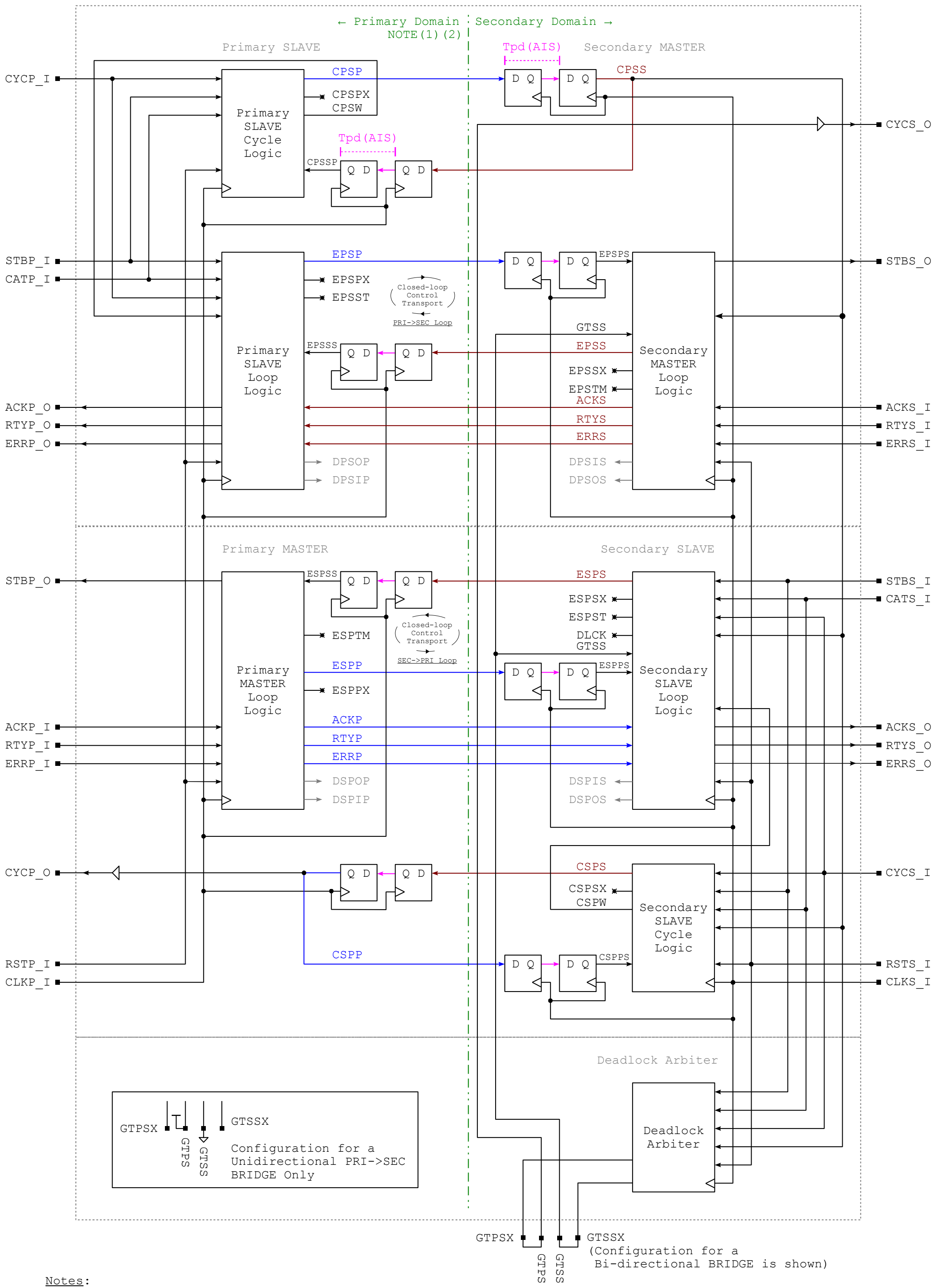
4. *Secondary MASTER Loop Logic*. Secondary MASTER Loop Logic responds by asserting secondary strobe output signal [STBS_O]. The remote SLAVE interface responds by asserting a terminal signal (e.g. [ACKS_I]), which causes the Secondary MASTER Loop Logic to respond by asserting loop signal [EPSS]. It also asserts one of three related domain-crossing signals. Thus, if acknowledge input signal [ACKS_I] is received from the remote SLAVE, then the secondary acknowledge signal [ACKS] is asserted, if [RTYS_I] then [RTYS] or if [ERRS_I] then [ERRS]. The asserted signal propagates back across the clock domain boundary, and is registered at the primary side within the time specified by the propagation delay entry specification Tpd(PRI), clk-su.

5. *Primary SLAVE Loop Logic*. Primary SLAVE Loop Logic monitors loop input signal [EPSSS], and responds by asserting a related terminal signal back to the remote MASTER. Thus, if secondary acknowledge signal [ACKS] is asserted, then the acknowledge output signal [ACKP_O] is sent to the remote MASTER; if [RTYS] then [RTYS_O] or if [ERRS] then [ERRS_O].

6. *Remote MASTER*. Remote MASTER terminates the cycle causing primary strobe input signal [STBP_I] to be negated. Alternatively, the MASTER can begin another phase of a multi-phase DATUM INTERLOCKED cycle.

When configured as a bidirectional BRIDGE, the control transport structure also implements a deadlock arbiter, which resides in the secondary clock domain. Deadly embrace (deadlock) is a situation that might occur when both sides of the BRIDGE attempt to access the other at the same time. This is prevented when the deadlock arbiter recognizes the situation and responds by allowing one side to complete its bus cycle, and terminates the other by asserting the retry output signal [RTYP_O] or [RTYS_O].

Ownership of the bidirectional BRIDGE is requested when a SLAVE monitors that both the cycle input signal [CYC_I] and strobe input signal [STB_I] are asserted. Once the deadlock arbiter grants ownership, a request from the opposite side of the BRIDGE is rejected by asserting the retry output signal [RTYP_O] or [RTYS_O]. In the event that requests are obtained simultaneously from both sides, the deadlock arbiter grants the higher priority to the primary interface. Ownership is maintained until the related cycle input signal [CYCP_I] or [CYCS_I] is negated.



Notes:

(1) Domain crossing signal inputs to secondary side are shown in light blue; LCCP = Tpd(SEC), clk-su (1 LUT path).
Domain crossing signal inputs to primary side are shown in red ; LCCP = Tpd(PRI), clk-su (1 LUT path).
Asynchronous interflop synchronizer transfer delay shown in magenta ; LCCP = Tpd(AIS), clk-su (1 LUT path).

(2) Domain crossing signal inputs which are not terminated by a double D-type synchronizer are guaranteed to have at least two (destination) clock periods available for clk-su.

Figure 57. Control transport structure for the asynchronous trans-domain BRIDGE.

```
Logic Equations (Canonical Form), Primary Side:                                         LCCP (ILUT)
                                                                                        IN L1 L2 L3       Notes / Example LUT Functions
CPSP   :=           CPSP                               &  CYCP_I & /RSTP_I                    +2     L2=(1A,1B,0,0,0);
NOTE(1) +                  /CPSSP & [STBP_I & /CATP_I] &  CYCP_I & /RSTP_I;                          1A=(CPSP,CYCP_I,RSTP_I,0,0);
                                                                                                     1B=(CPSSP,STBP_I,CATP_I,CYCP_I,RSTP_I);

CPSPX  :=  CPSPX &  CPSP                                         & /RSTP_I                 +1        Self-evident
        +  CPSPX         &  CPSSP                                & /RSTP_I
        +           CPSP &  CPSSP                      &  CYCP_I & /RSTP_I;

CPSW   :=                   CPSPX &    CPSP & /CYCP_I & /RSTP_I                               +2     L2=(1A,1B,0,0,0);
        +   CPSW & CPSSP                              & /RSTP_I;                                     1A=(CPSPX,CPSP,CYCP_I,RSTP_I,0);
                                                                                                     1B=(CPSW,CPSSP,RSTP_I,0,0);
EPSP   := /EPSPX                  &  EPSST & /RSTP_I                                          +2     L2=EPSPX,EPSP,EPSST(1A),EPSST(1B),RSTP_I);
        +           EPSP          & /EPSST & /RSTP_I;

EPSPX  :=  EPSPX         &  EPSSS          & /RSTP_I                                       +1        Self-evident
        +  EPSPX &  EPSP                   & /RSTP_I
        +           EPSP &  EPSSS          & /RSTP_I;

EPSST   =  CYCP_I & STBP_I & /CATP_I & /ACKP_O & /RTYP_O & /ERRP_O & /CPSW;              Intermed.  EPSST(1A)=(CYCP_I,STBP_I,CATP_I,ACKP_O,0)
                                                                                                    EPSST(1B)=(RTYP_O,ERRP_O,CPSW,0,0);
DPSOP   =  EPSPX &  EPSP &  EPSST                                                             +2    L2=(1A,EPSST(1A),EPSST(1B),SIG-IN,SIG-FB);
NOTE(4) + /EPSPX & /EPSP &  EPSST;                                                                  1A=(EPSPX,EPSP,0,0,0);

DPSIP   =  EPSPX & /EPSP & /EPSSS                                                          +1       L1=(EPSPX,EPSP,EPSSS,SIG-IN,SIG-FB);
NOTE(4) + /EPSPX &  EPSP &  EPSSS;

ACKP_O := /ACKP_O & /EPSPX &  EPSP &  EPSSS & ACKS & /RSTP_I                                  +2    L2=(1A,ACKS,0,0,0);
        + /ACKP_O &  EPSPX & /EPSP & /EPSSS & ACKS & /RSTP_I;                                       1A=(ACKP_O,EPSPX,EPSP,EPSSS,RSTP_I);

RTYP_O := /RTYP_O & /EPSPX &  EPSP &  EPSSS & RTYS & /RSTP_I                                  +2    L2=(1A,RTYS,0,0,0);
        + /RTYP_O &  EPSPX & /EPSP & /EPSSS & RTYS & /RSTP_I;                                       1A=(RTYP_O,EPSPX,EPSP,EPSSS,RSTP_I);

ERRP_O := /ERRP_O & /EPSPX &  EPSP &  EPSSS & ERRS                            & /RSTP_I       +2    L2=(1A,1B,ERRS,0,0);
        + /ERRP_O &  EPSPX & /EPSP & /EPSSS & ERRS                            & /RSTP_I             1A=(ERRP_O,EPSPX,EPSP,EPSSS,RSTP_I);
NOTE(2) + /ERRP_O                                  & CYCP_I & STBP_I & CATP_I & /RSTP_I;            1B=(ERRP_O,CYCP_I,STBP_I,CATP_I,RSTP_I);

ESPP   :=  ESPPX         &  ESPTM          & /RSTP_I                                          +2    L2=(ESPPX,ESPP,ESPTM(1A),ESPTM(1B),RSTP_I);
        +           ESPP & /ESPTM          & /RSTP_I;

ESPPX  :=  ESPPX & /ESPP                   & /RSTP_I                                       +1       Self-evident
        +  ESPPX                  &  ESPSS & /RSTP_I
        +          /ESPP          &  ESPSS & /RSTP_I;

STBP_O := /STBP_O & /ESPPX & /ESPP &  ESPSS                               & /RSTP_I           +2    L2=(1A,1B,0,0,0);
        + /STBP_O &  ESPPX &  ESPP & /ESPSS                               & /RSTP_I                 1A=(STBP_O,ESPPX,ESPP,ESPSS,RSTP_I);
        +  STBP_O                           & /ACKP_I & /RTYP_I & /ERRP_I & /RSTP_I;                1B=(STBP_O,ACKP_I,RTYP_I,ERRP_I,RSTP_I);

ESPTM   =  STBP_O & ACKP_I                                                               Intermed.  ESPTM(1A)=(STBP_O,ACKP_I,RTYP_I,ERRP_I,0);
        +  STBP_O          & RTYP_I                                                                 ESPTM(1B)=(STBP_O,ESPPX,ESPP,0,0);
        +  STBP_O                   & ERRP_I
        + /STBP_O                            &  ESPPX & /ESPP
        + /STBP_O                            & /ESPPX &  ESPP;

ACKP   :=          STBP_O & ACKP_I                                                         +1       Self-evident
        +  ACKP & /STBP_O          &  ESPPX &  ESPP
        +  ACKP & /STBP_O          & /ESPPX & /ESPP;

RTYP   :=          STBP_O & RTYS_I                                                         +1       Self-evident
        +         /STBP_O          &  ESPPX & /ESPP
        +         /STBP_O          & /ESPPX &  ESPP
        +  RTYP & /STBP_O          &  ESPPX &  ESPP
        +  RTYP & /STBP_O          & /ESPPX & /ESPP;

ERRP   :=          STBP_O & ERRP_I                                                         +1       Self-evident
        +  ERRP & /STBP_O          &  EPSSX &  EPSS
        +  ERRP & /STBP_O          & /EPSSX & /EPSS;

DSPOP   =  ESPPX & /ESPP &  ESPTM                                                             +2    L2=(1A,ESPTM(1A),ESPTM(1B),SIG-IN,SIG-FB));
NOTE(4) + /ESPPX &  ESPP &  ESPTM;                                                                  1A=(ESPPX,ESPP,0,0,0);

DSPIP   =  ESPPX &  ESPP &  ESPSS                                                          +1       Self-evident
NOTE(4) + /ESPPX & /ESPP &  ESPSS;
```

Notes:

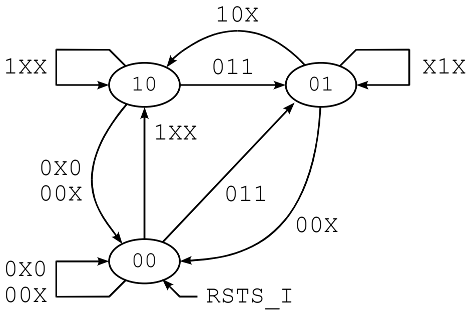
```
(1) Domain crossing signal inputs to secondary side are shown in light blue; LCCP = Tpd(SEC), clk-su (1 LUT path).
    Domain crossing signal inputs to primary   side are shown in    red    ; LCCP = Tpd(PRI), clk-su (1 LUT path).
    Domain crossing signal inputs which are not terminated by a double D-type synchronizer (not shown) are guaranteed to have
    at least two (destination) clock periods available for clk-su.
(2) The interface responds with an error when BLOCK INTERLOCKED cycles are directed at the SLAVE.
(3) Bidirectional operation:           [GTSS] = [GTSSX]; [GTPS] = [GTPSX].
    Unidirectional PRI->SEC operation: [GTPS] = '1'; [CYCS_I],[STBS_I],[CATS_I],[ACKP_I],[RTYP_I],[ERRP_I] = '0'.
    Unidirectional SEC->PRI operation: [GTSS] = '1'; [CYCP_I],[STBP_I],[CATP_I],[ACKS_I],[RTYS_I],[ERRS_I] = '0'.
(4) LCCP estimate also applies to clock enable on a data transport register. Each equation describes one signal, such as a data bit.  In the
    equations, 'SIG-IN' represents an input signal, and 'SIG-FB' represents a feedback path.
```

Figure 58. Logic equations (p. 1 of 2) for the asynchronous trans-domain BRIDGE.

```
Logic Equations (Canonical Form), Secondary Side:                                                   LCCP (ILUT)
                                                                                                    IN L1 L2 L3       Notes / Example LUT Functions
CSPS   :=           CSPS                                       &  CYCS_I & /RSTS_I                        +2     L2=(1A,1B,0,0,0);
NOTE(1) +                 /CSPPS & [/CPSS & STBS_I & /CATS_I] &  CYCS_I & /RSTS_I;                              1A=(CYCS_I,RSTS_I,0,0,0);
                                                                                                                 1B=(CSPS,CSPPS,CPSS,STBS_I,CATS_I);
CSPSX  :=  CSPSX &  CSPS                                                 & /RSTS_I                     +1        Self-evident
        +  CSPSX         &  CSPPS                                        & /RSTS_I
        +           CSPS &  CSPPS                              &  CYCS_I & /RSTS_I;

CSPW   :=                   CSPSX &    CSPS & /CYCS_I & /RSTS_I                                           +2     L2=(1A,1B,0,0,0);
        +   CSPW & CSPPS                              & /RSTS_I;                                                 1A=(CSPSX,CSPS,CYCS_I,RSTS_I,0);
                                                                                                                 1B=(CSPW,CSPPS,RSTS_I,0,0);
ESPS   :=                   ESPS          & /ESPST          & /RSTS_I
        +                   ESPS                    &  CPSS & /RSTS_I                                     +2     L2=(1A,1B,1C,ESPST(1A),ESPST(1B))
        +  DLCK &  ESPSX                  &  ESPST  & /CPSS & /RSTS_I                                            1A=(DLCK,ESPSX,CPSS,RSTS_I,0);
        + /DLCK & /ESPSX                  &  ESPST  & /CPSS & /RSTS_I;                                           1B=(ESPS,CPSS,RSTS_I,0,0);
                                                                                                                 1C=(ESPS,RSTS_I,0,0,0);
ESPSX  :=          ESPSX &  ESPS                                & /RSTS_I                              +1        Self-evident
        +          ESPSX         & ESPPS                        & /RSTS_I
        +                   ESPS & ESPPS                        & /RSTS_I;

DLCK   := /DLCK &  ESPSX &  ESPS          &  ESPST &  CPSS & /RSTS_I                                      +2     L2=(1A,ESPST(1A),ESPST(1B),0,0);
        + /DLCK & /ESPSX & /ESPS          &  ESPST &  CPSS & /RSTS_I;                                            1A=(DLCK,ESPSX,ESPS,CPSS,RST_I);

ESPST   = /CPSS                 &  CYCS_I & STBS_I & /CATS_I & /ACKS_O & /RTYS_O & /ERRS_O           Intermed.   ESPST(1A)=(STBS_I,CATS_I,ACKS_O,RTYS_O,ERRS_O);
        +          GTSS & /CSPW &  CYCS_I & STBS_I & /CATS_I & /ACKS_O & /RTYS_O & /ERRS_O;                      ESPST(1B)=(CPSS,GTSS,CSPW,CYCS_I,0);

ACKS_O := /ACKS_O & /ESPSX &  ESPS &  ESPPS & ACKP & /RSTS_I                                        I1    +2     L2=(1A,ACKP,0,0,0);
        + /ACKS_O &  ESPSX & /ESPS & /ESPPS & ACKP & /RSTS_I;                                                    1A=(ACKS_O,ESPSX,ESPS,ESPPS,RSTS_I);

RTYS_O := /RTYS_O & /ESPSX &  ESPS &  ESPPS & RTYP                & /RSTS_I                         I1    +2     L2=(1A,1B,1C,ESPST(1B),RTYP);
        + /RTYS_O &  ESPSX & /ESPS & /ESPPS & RTYP                & /RSTS_I                                      1A=(RTYS_O,ESPSX,ESPS,ESPPS,RSTS_I);
        + /RTYS_O & /ESPSX & /ESPS                 & ESPST & CPSS & /RSTS_I                                      1B=(RTYS_O,ESPSX,ESPS,CPSS,RSTS_I);
        + /RTYS_O &  ESPSX &  ESPS                 & ESPST & CPSS & /RSTS_I;                                     1C=(GTSS,CSPW,CYCS_I,0,0);

ERRS_O := /ERRS_O & /ESPSX &  ESPS &  ESPPS & ERRP                            & /RSTS_I             I1    +2     L2=(1A,1B,ERRP,RSTS_I,0);
        + /ERRS_O &  ESPSX & /ESPS & /ESPPS & ERRP                            & /RSTS_I                          1A=(ERRS_O,ESPSX,ESPS,ESPPS,RSTS_I);
NOTE(2) + /ERRS_O                                  & CYCS_I & STBS_I & CATS_I & /RSTS_I;                         1B=(ERRS_O,CYCS_I,STBS_I,CATS_I,RSTS_I);

DSPIS   =  ESPSX & /ESPS &  ESPPS                                                                      +1        L1=(ESPSX,ESPS,ESPPS,SIG-IN,SIG-FB);
NOTE(4) + /ESPSX &  ESPS &  ESPPS;

DSPOS   =  ESPSX &  ESPS &  ESPST                                                                         +2     L1=(1A,ESPST(1A),ESPST(1B),SIG-IN,SIG-FB));
NOTE(4) + /ESPSX & /ESPS &  ESPST;                                                                               1A=(ESPSX,ESPS,0,0,0);

EPSS   :=  EPSSX         &  EPSTM                          & /RSTS_I                                      +2     L1=(EPSSX,EPSS,EPSTM(1A),EPSTM(1B),RSTS_I);
        +           EPSS & /EPSTM                          & /RSTS_I;

EPSSX  :=  EPSSX & /EPSS                           & /RSTS_I                                           +1        Self-evident
        +  EPSSX                           & /CPSS & /RSTS_I
        +  EPSSX                   & EPSPS         & /RSTS_I
        +          /EPSS &         & EPSPS &  CPSS & /RSTS_I;

STBS_O := /STBS_O & /EPSSX & /EPSS &  EPSPS &  CPSS & /GTSS                               & /RSTS_I       +2     L2=(1A,1B,EPSSX,EPSS,EPSPS);
        + /STBS_O &  EPSSX &  EPSS & /EPSPS &  CPSS & /GTSS                               & /RSTS_I              1A=(STBS_O,CPSS,GTSS,RSTS_I,0)
        +  STBS_O                                          & /ACKS_I & /RTYS_I & /ERRS_I & /RSTS_I;              1B=(STBS_O,ACKS_I,RTYS_I,ERRS_I,RSTS_I);

EPSTM   =  STBS_O & ACKS_I                                                                           Intermed.   EPSTM(1A)=(STBS_O,ACKS_I,RTYS_I,ERRS_I,0);
        +  STBS_O         & RTYS_I                                                                               EPSTM(1B)=(STBS_O,EPSSX,EPSS,0,0);
        +  STBS_O                  & ERRS_I
        + /STBS_O                            &  EPSSX & /EPSS
        + /STBS_O                            & /EPSSX &  EPSS;

ACKS   :=          STBS_O & ACKS_I                                                                     +1        Self-evident
        +  ACKS & /STBS_O          &  EPSSX &  EPSS
        +  ACKS & /STBS_O          & /EPSSX & /EPSS;

RTYS   :=          STBS_O & RTYS_I
        +         /STBS_O          &  EPSSX & /EPSS                                                    +1        Self-evident
        +         /STBS_O          & /EPSSX &  EPSS
        +  RTYS & /STBS_O          &  EPSSX &  EPSS
        +  RTYS & /STBS_O          & /EPSSX & /EPSS;

ERRS   :=          STBS_O & ERRS_I                                                                     +1        Self-evident
        +  ERRS & /STBS_O          &  EPSSX &  EPSS
        +  ERRS & /STBS_O          & /EPSSX & /EPSS;

DPSIS   =  EPSSX &  EPSS & /EPSPS                                                                      +1        Self-evident
NOTE(4) + /EPSSX & /EPSS &  EPSPS;

DPSOS   =  EPSSX & /EPSS &  EPSTM                                                                         +2     L2=(1A,EPSTM(1A),EPSTM(1B),SIG-IN,SIG-FB));
NOTE(4) + /EPSSX &  EPSS &  EPSTM;                                                                               1A=(EPSSX,EPSS,0,0,0);

GTPSX  :=   /GTSSX &   CPSS           & /RSTS_I                                                        +1        Self-evident
NOTE(3) +              CPSS & /CYCS_I & /RSTS_I;

GTSSX  :=    GTSSX          &  CYCS_I                       & /RSTS_I                                     +2     L2=(1A,1B,0,0,0);
NOTE(3) +             /CPSS &  CYCS_I &  [STBS_I & /CATS_I] & /RSTS_I;                                           1A=(GTSSX,CYCS_I,RSTS_I,0,0);
                                                                                                                 1B=(CPSS,CYCS_I,STBS_I,CATS_I,RSTS_I);
```

Figure 59. Logic equations (p. 2 of 2) for the asynchronous trans-domain BRIDGE.

State Diagram
Deadlock Arbiter



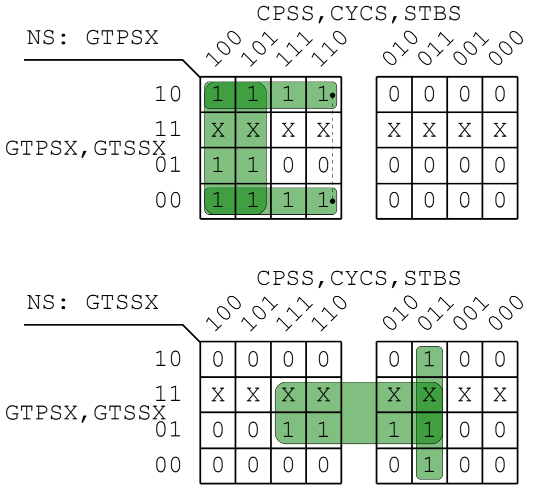
STATES: GTPSX,GTSSX
INPUTS: CPSS,CYCS,STBS

CPSS: Cycle, PRI->SEC
CYCS: Cycle, SEC
GTPSX: Grant, PRI->SEC
GTSSX: Grant, SEC->PRI
STBS: Strobe, SEC

Truth Table

GGCCS	GG
TTPYT	TT
PSSCB	PS
SSSSS	SS
XX	XX
-----	--
10000	00
10001	00
10011	01
10010	00
10110	10
10111	10
10101	10
10100	10
11100	XX
11101	XX
11111	XX
11110	XX
11010	XX
11011	XX
11001	XX
11000	XX
01000	00
01001	00
01011	01
01010	01
01110	01
01111	01
01101	10
01100	10
00100	10
00101	10
00111	10(*)
00110	10
00010	00
00011	01
00001	00
00000	00

Karnaugh Map
Deadlock Arbiter



NS: GTPSX

CPSS,CYCS,STBS

GTPSX,GTSSX	100	101	111	110	010	011	001	000
10	1	1	1	1	0	0	0	0
11	X	X	X	X	X	X	X	X
01	1	1	0	0	0	0	0	0
00	1	1	1	1	0	0	0	0

NS: GTSSX

CPSS,CYCS,STBS

GTPSX,GTSSX	100	101	111	110	010	011	001	000
10	0	0	0	0	0	1	0	0
11	X	X	X	X	X	X	X	X
01	0	0	1	1	1	1	0	0
00	0	0	0	0	0	1	0	0

General Assumptions

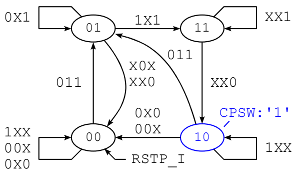
Initial secondary cycle must show [CYCS_I] and [STBS_I] both asserted because signal [CYC] is broadcast to all SLAVEs, without address decoding.

STBS = STBS_I & /CATS_I;

(*) Primary has priority over secondary.

Figure 60. Deadlock arbiter for the asynchronous trans-domain BRIDGE.

State Diagram
Primary SLAVE Cycle Logic
NOTE(1)



```
STATES: CPSPX, CPSP
INPUTS: CPSSP, STBP, CYCP_I

CPSP:   Cycle PRI->SEC, pri
CPSPX:  Cycle PRI->SEC, pri x
CPSSP:  Cycle PRI->SEC, return
CPSW:   Cycle PRI->SEC, wait
CYCP_I: Primary cycle input
STBP:   Valid primary strobe
```

Karnaugh Map
Primary SLAVE Cycle Logic

NS: CPSP

CPSPX,CPSP \ CPSSP,STBP,CYCP_I	100	101	111	110	010	011	001	000
10	0	0	0	0	0	1	0	0
11	0	1	1	0	0	1	1	0
01	0	1	1	0	0	1	1	0
00	0	0	0	0	0	1	0	0

NS: CPSPX

CPSPX,CPSP \ CPSSP,STBP,CYCP_I	100	101	111	110	010	011	001	000
10	1	1	1	1	0	0	0	0
11	1	1	1	1	1	1	1	1
01	0	1	1	0	0	0	0	0
00	0	0	0	0	0	0	0	0

Truth Table

```
CCC C CC
PPPSY PP
SSSTC SS
PPSBP PP
X PPI X
----- --
10000 00
10001 00
10011 01
10010 00

10110 10
10111 10
10101 10
10100 10

11100 10
11101 11
11111 11
11110 10

11010 10
11011 11
11001 11
11000 10

01000 00
01001 01
01011 01
01010 00

01110 00
01111 11
01101 11
01100 00

00100 00
00101 00
00111 00
00110 00

00010 00
00011 01
00001 00
00000 00
```

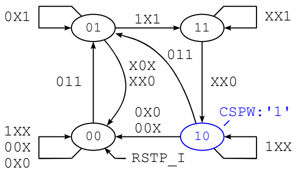
Notes:

(1) The purpose of this state machine is to hold the deadlock arbiter and the destination bus during a CONGA cycle.

The purpose of the wait signal [CPSW] is to delay a new closed-loop control transport cycle from starting during back-to-back assertions of the cycle signal [CYCP_I].

The following substitution is assumed:
STBP = STBP_I & /CATP_I;

State Diagram
Secondary SLAVE Cycle Logic
NOTE(1)



```
STATES: CSPSX, CSPS
INPUTS: CSPPS, STBS, CYCS_I

CSPS:   Cycle PRI->SEC, pri
CSPSX:  Cycle PRI->SEC, pri x
CSPPS:  Cycle PRI->SEC, return
CSPW:   Cycle PRI->SEC, wait
CYCS_I: Primary cycle input
STBS:   Valid primary strobe
```

Karnaugh Map
Secondary SLAVE Cycle Logic

NS: CSPS

CSPSX,CSPS \ CSPPS,STBS,CYCS_I	100	101	111	110	010	011	001	000
10	0	0	0	0	0	1	0	0
11	0	1	1	0	0	1	1	0
01	0	1	1	0	0	1	1	0
00	0	0	0	0	0	1	0	0

NS: CSPSX

CSPSX,CSPS \ CSPPS,STBS,CYCS_I	100	101	111	110	010	011	001	000
10	1	1	1	1	0	0	0	0
11	1	1	1	1	1	1	1	1
01	0	1	1	0	0	0	0	0
00	0	0	0	0	0	0	0	0

Truth Table

```
CCC C CC
SSSSY SS
PPPTC PP
SSPBS SS
X SSI X
----- --
10000 00
10001 00
10011 01
10010 00

10110 10
10111 10
10101 10
10100 10

11100 10
11101 11
11111 11
11110 10

11010 10
11011 11
11001 11
11000 10

01000 00
01001 01
01011 01
01010 00

01110 00
01111 11
01101 11
01100 00

00100 00
00101 00
00111 00
00110 00

00010 00
00011 01
00001 00
00000 00
```

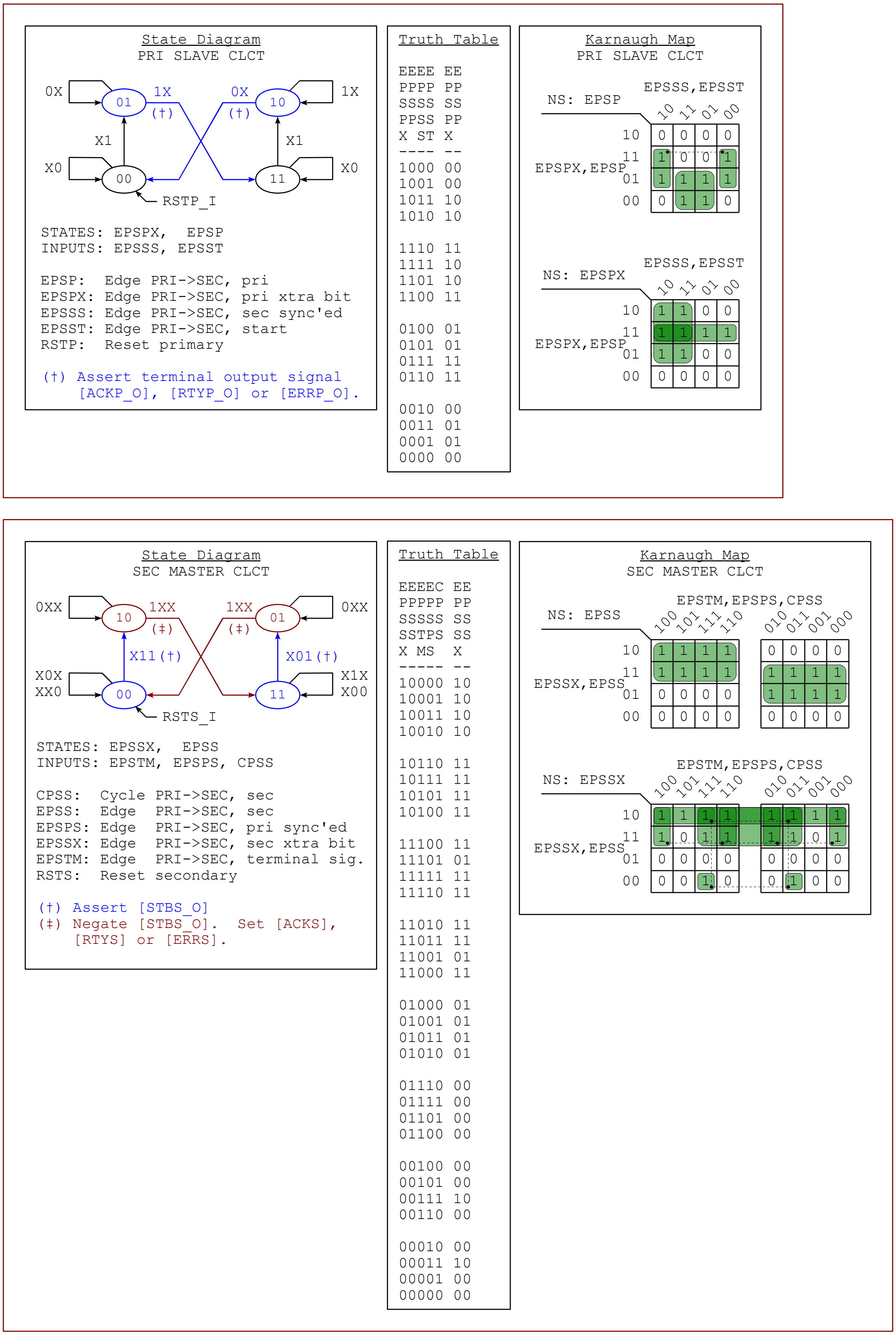
Notes:

(1) The purpose of this state machine is to hold the deadlock arbiter and the destination bus during a CONGA cycle.

The purpose of the wait signal [CSPW] is to delay a new closed-loop control transport cycle from starting during back-to-back assertions of the cycle signal [CYCS_I].

The following substitution is assumed:
STBS = /CPSS & STBS_I & /CATS_I;

Figure 61. SLAVE Cycle Logic for the asynchronous trans-domain BRIDGE.



STATES: EPSPX, EPSP
INPUTS: EPSSS, EPSST

EPSP: Edge PRI->SEC, pri
EPSPX: Edge PRI->SEC, pri xtra bit
EPSSS: Edge PRI->SEC, sec sync'ed
EPSST: Edge PRI->SEC, start
RSTP: Reset primary

(†) Assert terminal output signal
[ACKP_O], [RTYP_O] or [ERRP_O].

Truth Table

```
EEEE EE
PPPP PP
SSSS SS
PPSS PP
X ST X
---- --
1000 00
1001 00
1011 10
1010 10

1110 11
1111 10
1101 10
1100 11

0100 01
0101 01
0111 11
0110 11

0010 00
0011 01
0001 01
0000 00
```

Karnaugh Map
PRI SLAVE CLCT

NS: EPSP

EPSPX,EPSP \ EPSSS,EPSST	10	11	01	00
10	0	0	0	0
11	1	0	0	1
01	1	1	1	1
00	0	1	1	0

NS: EPSPX

EPSPX,EPSP \ EPSSS,EPSST	10	11	01	00
10	1	1	0	0
11	1	1	1	1
01	1	1	0	0
00	0	0	0	0

STATES: EPSSX, EPSS
INPUTS: EPSTM, EPSPS, CPSS

CPSS: Cycle PRI->SEC, sec
EPSS: Edge PRI->SEC, sec
EPSPS: Edge PRI->SEC, pri sync'ed
EPSSX: Edge PRI->SEC, sec xtra bit
EPSTM: Edge PRI->SEC, terminal sig.
RSTS: Reset secondary

(†) Assert [STBS_O]
(‡) Negate [STBS_O]. Set [ACKS],
[RTYS] or [ERRS].

Truth Table

```
EEEEC EE
PPPPP PP
SSSSS SS
SSTPS SS
X MS  X
----- --
10000 10
10001 10
10011 10
10010 10

10110 11
10111 11
10101 11
10100 11

11100 11
11101 01
11111 11
11110 11

11010 11
11011 11
11001 01
11000 11

01000 01
01001 01
01011 01
01010 01

01110 00
01111 00
01101 00
01100 00

00100 00
00101 00
00111 10
00110 00

00010 00
00011 10
00001 00
00000 00
```

Karnaugh Map
SEC MASTER CLCT

NS: EPSS

EPSSX,EPSS \ EPSTM,EPSPS,CPSS	100	101	111	110	010	011	001	000
10	1	1	1	1	0	0	0	0
11	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1
00	0	0	0	0	0	0	0	0

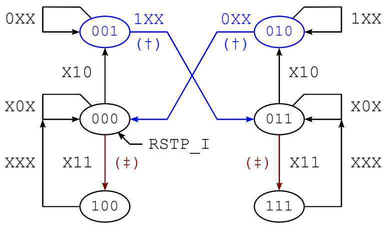
NS: EPSSX

EPSSX,EPSS \ EPSTM,EPSPS,CPSS	100	101	111	110	010	011	001	000
10	1	1	1	1	1	1	1	1
11	1	0	1	1	1	1	0	1
01	0	0	0	0	0	0	0	0
00	0	0	1	0	0	1	0	0

Figure 62. PRI->SEC Closed-loop Control Logic for the asynchronous trans-domain BRIDGE.

State Diagram

SEC SLAVE CLCT



```
STATES: DLCK,  ESPSX, ESPS
INPUTS: ESPPS, ESPST, CPSS

ESPS:  Edge SEC->PRI, sec
ESPSX: Edge SEC->PRI, sec xtra bit
ESPPS: Edge SEC->PRI, pri sync'ed
ESPST: Edge SEC->PRI, start
RSTS:  Reset secondary

Transition events (other logic):

(†) Assert terminal output signal
    [ACKS_O], [RTYS_O] or [ERRS_O]
    for one [CLKS] pulse.

(‡) Deadlock - assert [RTYS_O]
    for one [CLKS] pulse.
```

Truth Table

SEC SLAVE CLCT

```
DEEEEC DEE | DEEEEC DEE
LSSSSP LSS | LSSSSP LSS
CPPPPS CPP | CPPPPS CPP
KSSPSS KSS | KSSPSS KSS
 X ST   X  |  X ST   X
------ --- | ------ ---
100000 000 | 010000 000
100001 000 | 010001 000
100011 000 | 010011 000
100010 000 | 010010 000

100110 000 | 010110 010
100111 000 | 010111 010
100101 000 | 010101 010
100100 000 | 010100 010

101100 XXX | 011100 011
101101 XXX | 011101 011
101111 XXX | 011111 111
101110 XXX | 011110 010

101010 XXX | 011010 010
101011 XXX | 011011 111
101001 XXX | 011001 011
101000 XXX | 011000 011

111000 011 | 001000 001
111001 011 | 001001 001
111011 011 | 001011 001
111010 011 | 001010 001

111110 011 | 001110 011
111111 011 | 001111 011
111101 011 | 001101 011
111100 011 | 001100 011

110100 XXX | 000100 000
110101 XXX | 000101 000
110111 XXX | 000111 100
110110 XXX | 000110 001

110010 XXX | 000010 001
110011 XXX | 000011 100
110001 XXX | 000001 000
110000 XXX | 000000 000
```

Karnaugh Map

SEC SLAVE CLCT

NS: ESPS

DLCK,ESPSX,ESPS \ ESPPS,ESPST,CPSS	100	101	111	110	010	011	001	000
100	0	0	0	0	0	0	0	0
101	X	X	X	X	X	X	X	X
111	1	1	1	1	1	1	1	1
110	X	X	X	X	X	X	X	X
010	0	0	0	0	0	0	0	0
011	1	1	1	0	0	1	1	1
001	1	1	1	1	1	1	1	1
000	0	0	0	1	1	0	0	0

NS: ESPSX

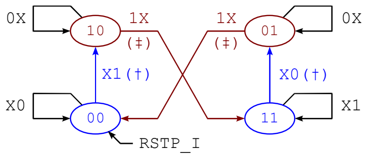
DLCK,ESPSX,ESPS \ ESPPS,ESPSTT,CPSS	100	101	111	110	010	011	001	000
100	0	0	0	0	0	0	0	0
101	X	X	X	X	X	X	X	X
111	1	1	1	1	1	1	1	1
110	X	X	X	X	X	X	X	X
010	1	1	1	1	0	0	0	0
011	1	1	1	1	1	1	1	1
001	1	1	1	1	0	0	0	0
000	0	0	0	0	0	0	0	0

NS: DLCK

DLCK,ESPSX,ESPS \ ESPPS,ESPSTT,CPSS	100	101	111	110	010	011	001	000
100	0	0	0	0	0	0	0	0
101	X	X	X	X	X	X	X	X
111	0	0	0	0	0	0	0	0
110	X	X	X	X	X	X	X	X
010	0	0	0	0	0	0	0	0
011	0	0	1	0	0	1	0	0
001	0	0	0	0	0	0	0	0
000	0	0	1	0	0	1	0	0

State Diagram

PRI MASTER CLCT



```
STATES: ESPPX,  ESPP
INPUTS: ESPTM, ESPSS

ESPP:  Edge  SEC->PRI, pri
ESPSS: Edge  SEC->PRI, sec sync'ed
ESPPX: Edge  SEC->PRI, pri xtra bit
ESPTM: Edge  SEC->PRI, terminal sig.
RSTS:  Reset secondary

Transition events (other logic):

(†) Assert [STBP_O]
(‡) Negate [STBP_O].  Set [ACKS],
    [RTYS] or [ERRS].
```

Truth Table

PRI MASTER CLCT

```
EEEE EE
SSSS SS
PPPP PP
PPTS PP
X MS X
---- --
1000 10
1001 10
1011 11
1010 11

1110 01
1111 11
1101 11
1100 01

0100 01
0101 01
0111 00
0110 00

0010 00
0011 10
0001 10
0000 00
```

Karnaugh Map

PRI MASTER CLCT

NS: ESPP

ESPPX,ESPP \ ESPTM,ESPSS	10	11	01	00
10	1	1	0	0
11	1	1	1	1
01	0	0	1	1
00	0	0	0	0

NS: ESPPX

ESPPX,ESPP \ ESPTM,ESPSS	10	11	01	00
10	1	1	1	1
11	0	1	1	0
01	0	0	0	0
00	0	1	1	0

Figure 63. SEC->PRI Closed-loop Control Logic for the asynchronous trans-domain BRIDGE.

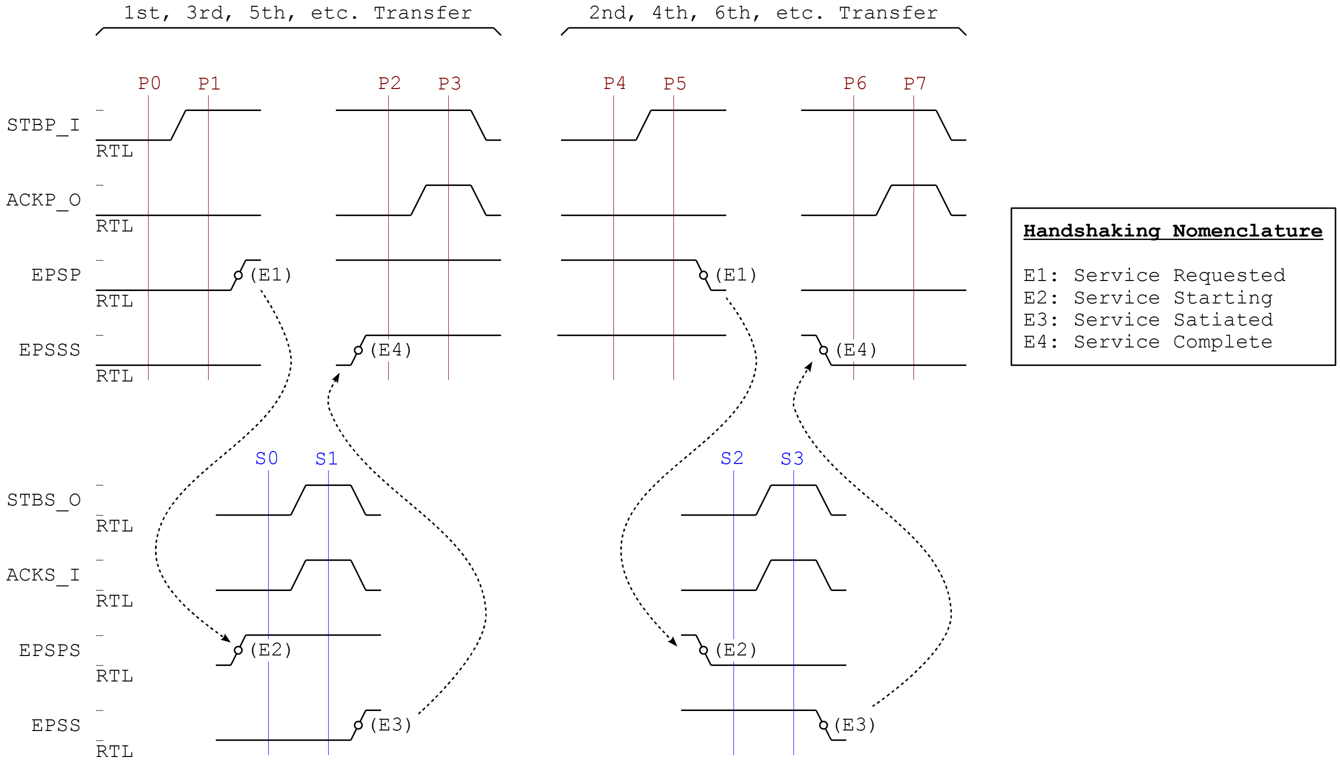


Figure 64. PRI->SEC internal timing detail for the asynchronous trans-domain BRIDGE.

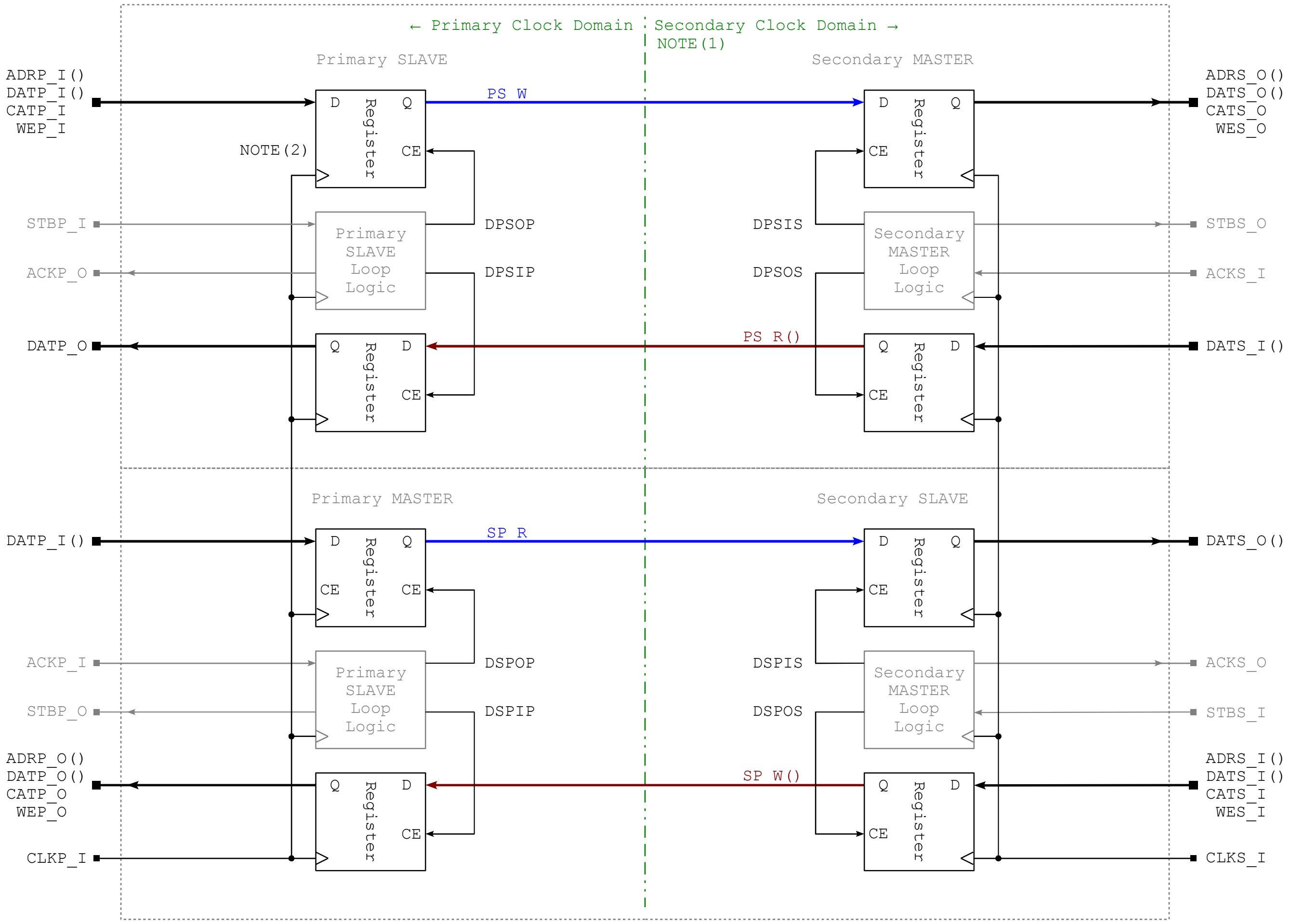
Data Transport Structure

Figure 65 shows the general topology for the data transport structure. That diagram demonstrates how identical registers and logic are used as generic transports for address bus [ADR], data bus [DAT], CONGA address tag [CAT] and write enable [WE] signals. The blocks shown for the primary and secondary loop logic are the same as those shown in the diagram above. For brevity, the terminal signals associated with retry and error (e.g. [RTYS] and [ERRS]) are not shown, as they are not used to control any of the registers in this diagram.

Note that the write enable input signals [WEP_I] and [WES_I] do not impact the flow of data through the BRIDGE.

System Reset Operation

The asynchronous trans-domain BRIDGE is designed to work in conjunction with an asynchronous reset bridge SYSCON described above. The two sides of the BRIDGE respond to primary and secondary reset input signals [RSTP_I] and [RSTS_I].



Notes:

```
(1) Domain crossing signal inputs to secondary side are shown in light blue; LCCP = Tpd(SEC), clk-su (1 LUT path).
    Domain crossing signal inputs to primary   side are shown in    red     ; LCCP = Tpd(PRI), clk-su (1 LUT path).

(2) Typical bitwise register equations for DAT(): DATPS_W(00) := DATP_I (00) &  DPSOP
                                                               + DATPS_W(00) & /DPSOP;

                                                  DATS_O(00)  := DATSP_R(00) &  DSPIS
                                                               + DATS_O (00) & /DSPIS;
```

Figure 65. Data transport structure for the asynchronous trans-domain BRIDGE.

Performance Benchmarks for the Asynchronous Trans-domain BRIDGE

Figure 66 is a synopsis of performance benchmarks for the asynchronous trans-domain BRIDGE example. This data was acquired using the graphical technique described elsewhere in this report.

Figure 66(a) shows the closed-loop control transport clock efficiency for PRI->SEC and SEC->PRI cycles using packed and unpacked interconnections on the destination side. All graphs apply to the READ, WRITE, DUPLEX and EVENT data transport operations. Noteworthy results include:

- *The length of a data transfer phase varies*. Assuming that the remote MASTER and SLAVE interfaces are repeatable, each phase of a multi-phase data transfer varies in length. This is due to an uncertainty caused by synchronization cycles (syncup) within the double D-type synchronizer at the domain crossing boundary.

- *Unidirectional and bidirectional operations are the same*. On the average, the clock efficiency is the same regardless of whether it is a unidirectional or a bidirectional BRIDGE.

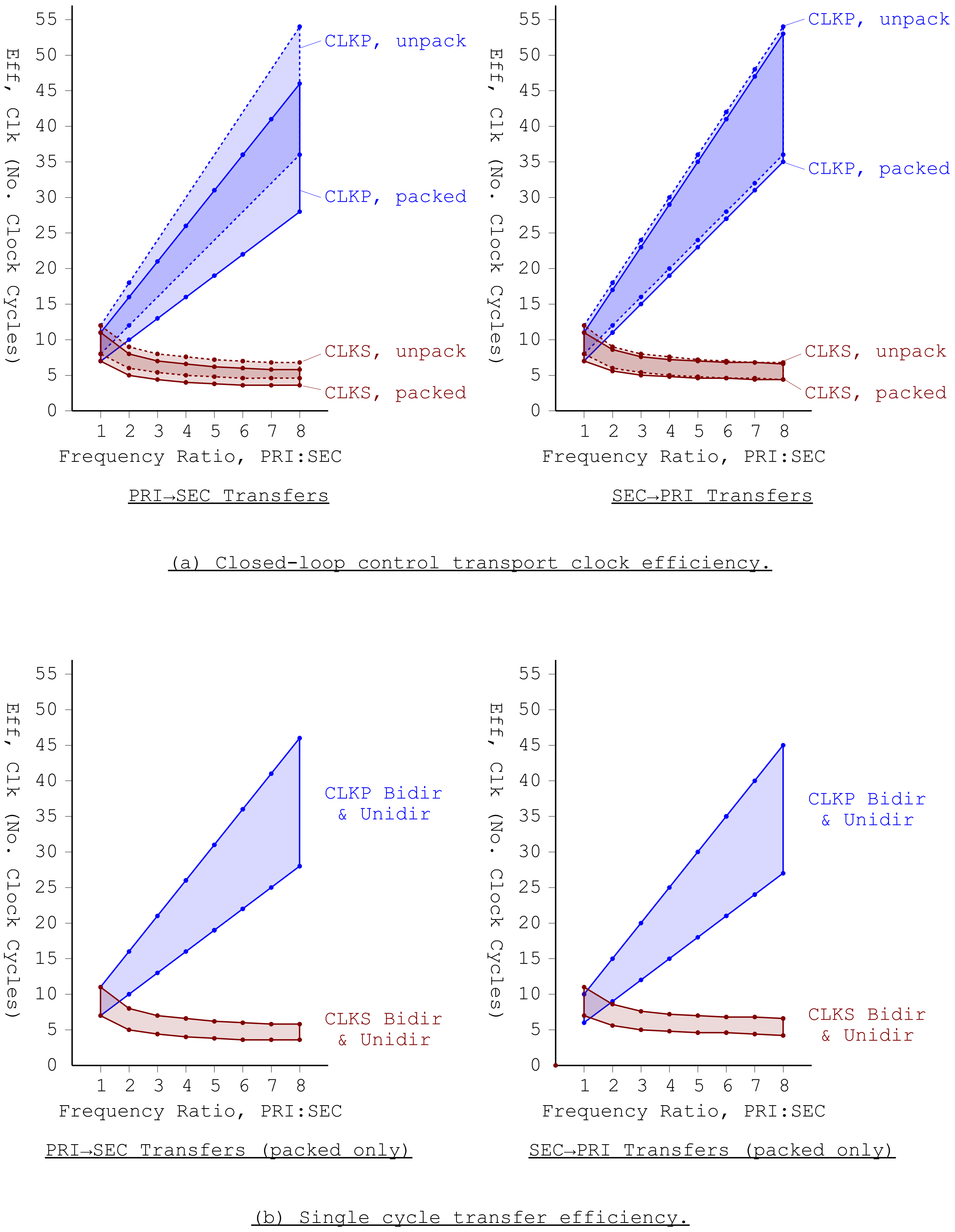
- *Overall clock efficiencies are low*. The overall clock efficiency compares data transfers across a BRIDGE versus across a point-to-point interconnection. It is useful for understanding the effects of blocking. This metric assumes a baseline reference for each point-to-point transfer of one (1) packed and two (2) unpacked clock cycles.

  A packed PRI->SEC BRIDGE transfer at a divisor ratio of approximately 4:1 requires an average of twenty-on (21) primary clock cycles to complete. Comparison to the baseline reference results in a clock efficiency of 21:1, or about 5%. An unpacked transfer requires an average of twenty-four (24) primary clock cycles, and results in a clock efficiency of 24:2, or about 8%.

Figure 66(b) shows the single cycle transfer efficiency. This measures the time duration in which the cycle signal [CYC] is asserted on the primary and secondary sides of the BRIDGE. Only packed cycles are considered. Noteworthy results include:

- *The deadlock arbiter has no effect*. The deadlock arbiter does not affect clock efficiency.

- *No disconnect penalty*. There is no 'disconnect penalty' associated with the cycle signal [CYC] in either the unidirectional or bidirectional BRIDGE topologies.



(a) Closed-loop control transport clock efficiency.

(b) Single cycle transfer efficiency.

Figure 66. Performance benchmarks for the asynchronous trans-domain BRIDGE.

Appendix – Handshaking Efficiency Study

This appendix provides an evaluation of the handshaking protocols used by the WISHBONE DATUM INTERLOCKED cycles for bridge[48] applications. It compares and contrasts the relative control transport clock efficiencies for the following three cases:

- Unblocked point-to-point interconnection (no-bridge baseline reference).
- A point-to-point interconnection blocked with one common-clock bridge.
- A point-to-point interconnection blocked with two common-clock bridges.

The handshaking modes used in the study include the well-established WISHBONE 3-Edge (AF) and 3-Edge (RF) handshake protocols. These are compared and contrasted to the traditional 4-Edge (TR) handshake protocol.

The study does not distinguish between the READ, WRITE, DUPLEX and EVENT data transport transactions. All use the same control transport handshaking modes.

3-Edge(AF) Interlocked Handshaking Mode

The 3-Edge(AF)[49] interlocked handshaking mode shown in Figure A-1 is a method for synchronizing WISHBONE MASTER and SLAVE interfaces during DATUM INTERLOCKED cycles. It is capable of an interlocked data transfer on every system clock [CLK] cycle, where both MASTER and SLAVE interfaces can throttle the data transfer rate by adding wait-states. The SLAVE interface asserts and negates a terminal signal asynchronously with respect to both the leading and trailing edges of the strobe input signal [STB_I].

From the perspective of the MASTER interface, the 3-Edge(AF) interlocked handshaking mode is forward and backward compatible with the 3-Edge(RF) interlocked handshaking mode. During the 3-Edge(AF) interlocked handshaking mode, the MASTER operates a finite state machine that drives the strobe output signal [STB_O], and monitors the acknowledge input signal [ACK_I]. If the strobe and acknowledge signals are tied together, then a single transfer will occur at every system clock [CLK] edge.

3-Edge(RF) Interlocked Handshaking Mode

The 3-Edge(RF)[50] interlocked handshaking mode shown in Figure A-2 is a method for synchronizing WISHBONE MASTER and SLAVE interfaces during DATUM INTERLOCKED cycles. It is capable of an interlocked data transfer on every second system clock [CLK] cycle, where both MASTER and SLAVE interfaces can throttle the data transfer rate by adding wait-states. In response to the leading edge of the strobe input signal [STB_I] the SLAVE interface asserts a terminal signal using a register, and in response to the trailing edge it negates a terminal signal asynchronously. It is useful for converting a 3-Edge(AF) cycle into a split transaction protocol, when necessary.

[48] The lowercase 'bridge' nomenclature is used throughout the appendix because non-standard (handshake only) structures are used.

[49] 'AF' is an acronym for *Asynchronous Feedback.*

[50] 'RF' is an acronym for *Registered Feedback.*

From the perspective of the MASTER interface, the 3-Edge(AF) interlocked handshaking mode is forward and backward compatible with the 3-Edge(RF) interlocked handshaking mode.

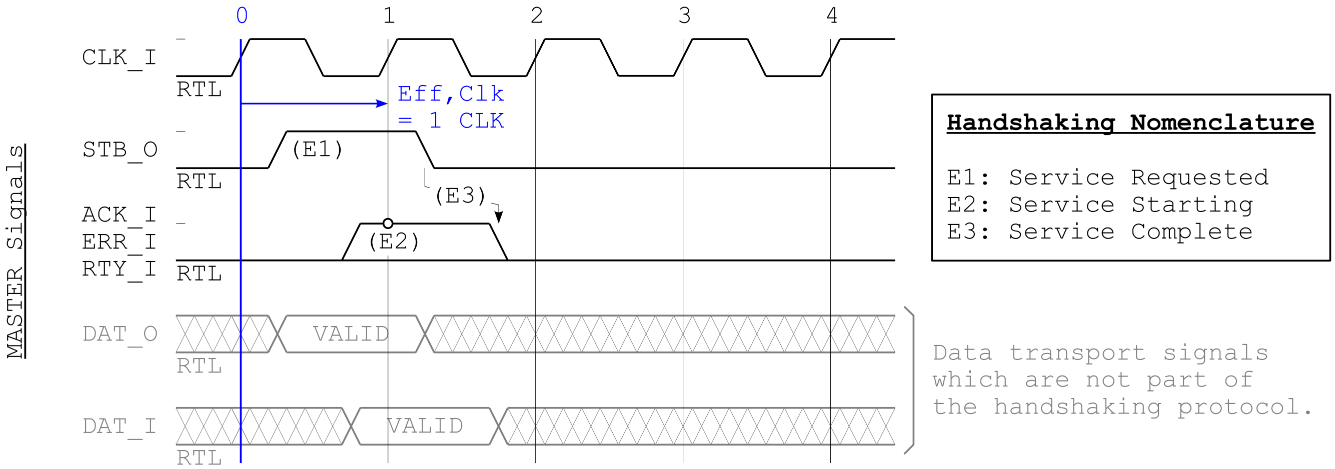


Figure A-1. Timing diagram for a 3-Edge(AF) DATUM INTERLOCKED cycle.

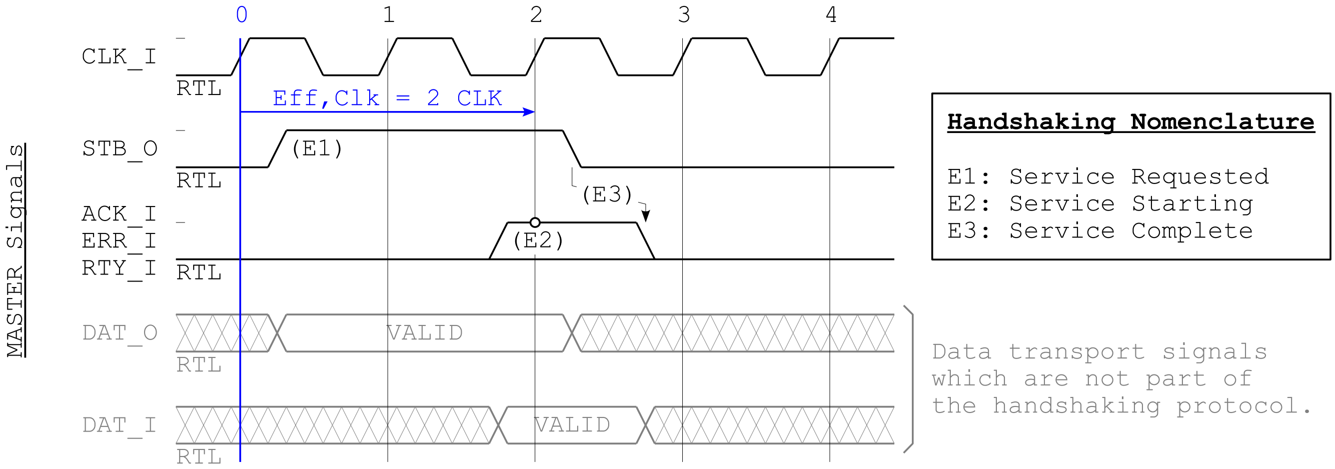
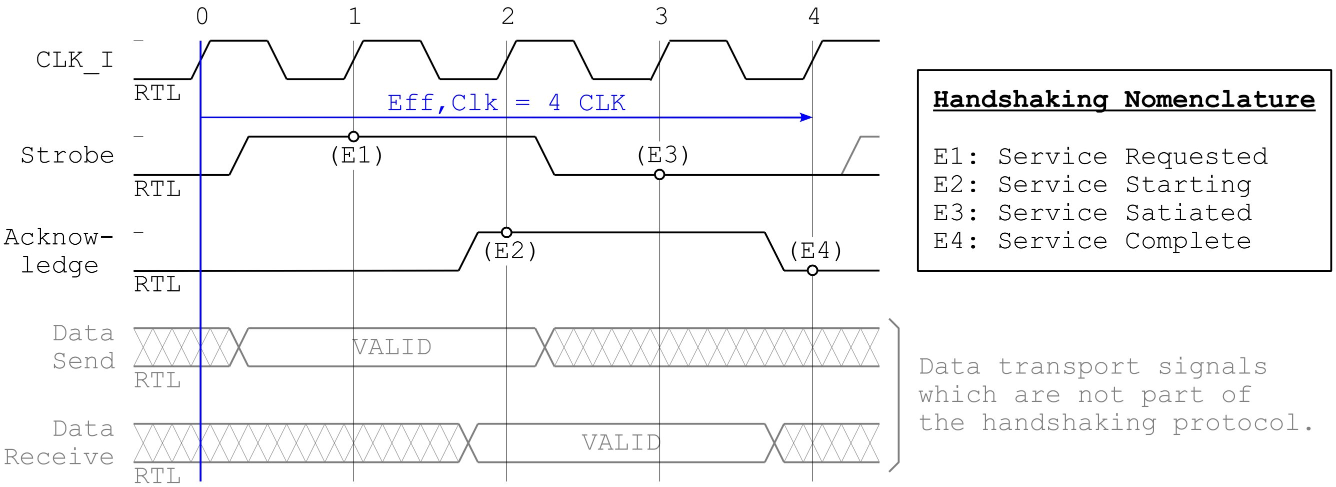


Figure A-2. Timing diagram for a 3-Edge(RF) DATUM INTERLOCKED cycle.

4-Edge(TR) Interlocked Handshaking Mode

The 4-Edge(TR)[51] interlocked handshaking mode shown in Figure A-3 is a method for interlocking data transfers between a sender and a receiver. It is a de facto industry standard, but is not used in the WISHBONE specification except generally for internal structures and benchmarking purposes.



Handshaking Nomenclature

E1: Service Requested
E2: Service Starting
E3: Service Satiated
E4: Service Complete

Data transport signals which are not part of the handshaking protocol.

Figure A-3. Timing diagram for a 4-Edge(TR) interlocked data transfer.

Test Methodology

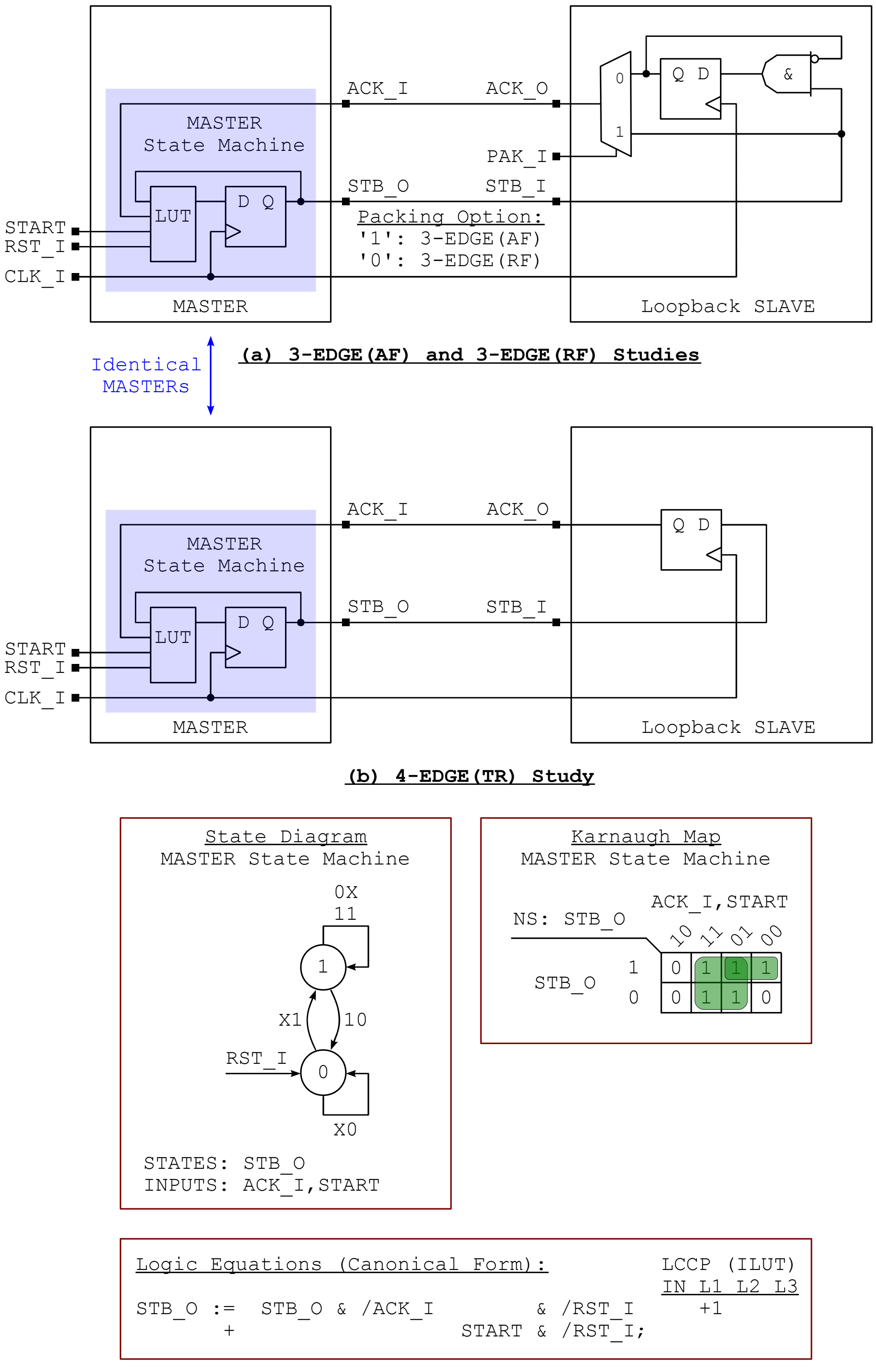
Figure A-4 shows the standard logic used for the MASTER and SLAVE interfaces during the test. Figure A-5 shows the bridge structures.

Note that the same strobe state machine is used for all three handshaking modes (3-Edge (AF), 3-Edge (RF) and 4-Edge (TR)). This state machine excites the strobe output signal [STB_O] in response to a start input signal [START], and then monitors acknowledge input signal [ACK_I] before terminating the cycle.

The standard WISHBONE packed and unpacked interconnections are simulated by the same SLAVE loopback interface. 3-Edge (AF) and 3-Edge (RF) cycles are selected at the remote SLAVE interface by the packing (balancing) option signal [PAK_I], which is assumed to be a implemented as a minimization exclusion. The traditional 4-Edge (TR) interconnection is modeled by a separate SLAVE interface.

These control transport structures are intended to complement a data transport structure. That is, the relationship between the control signals are meant to instruct the system components on how to process the associated data structure(s). Generally, this refers to READ, WRITE, DUPLEX and EVENT transactions.

[51] 'TR' is an acronym for *TRaditional.*



(a) 3-EDGE(AF) and 3-EDGE(RF) Studies

(b) 4-EDGE(TR) Study

Karnaugh Map
MASTER State Machine

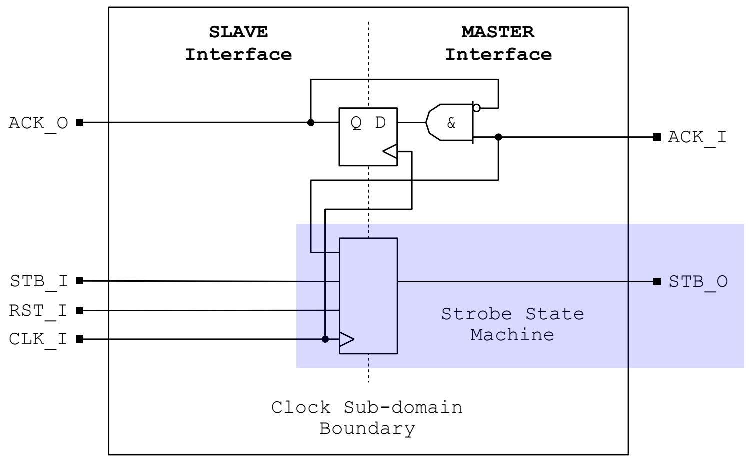
NS: STB_O

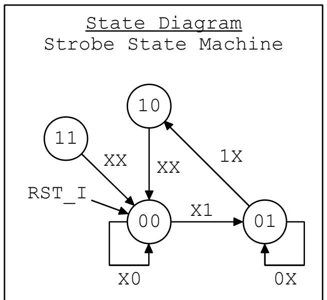
STB_O	ACK_I,START: 10	11	01	00
1	0	1	1	1
0	0	1	1	0

```
Logic Equations (Canonical Form):              LCCP (ILUT)
                                               IN L1 L2 L3
STB_O :=  STB_O & /ACK_I            & /RST_I      +1
      +                       START & /RST_I;
```

(c) MASTER State Machine

Figure A-4. MASTER and SLAVE interfaces used for the handshaking efficiency study.





State Diagram
Strobe State Machine

STATES: WAIT, STB_O
INPUTS: ACK_I, STB_I

ACK: Acknowledge
STB: Strobe
WAIT: Wait

Karnaugh Map
Strobe State Machine

NS: WAIT

ACK_I,STB_I

WAIT, STB_O	10	11	01	00
10	0	0	0	0
11	0	0	0	0
01	1	1	0	0
00	0	0	0	0

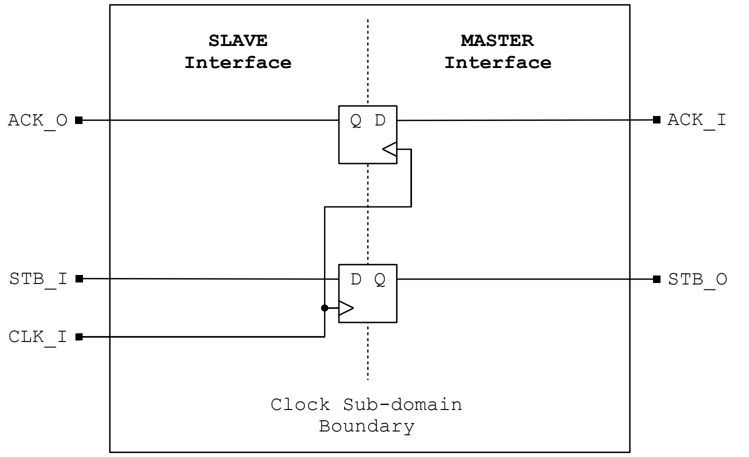
NS: STB_O

ACK_I,STB_I

WAIT, STB_O	10	11	01	00
10	0	0	0	0
11	0	0	0	0
01	0	0	1	1
00	0	1	1	0

```
Logic Equations (Canonical Form):                          LCCP (ILUT)
                                                           IN L1 L2 L3
WAIT  := /WAIT &  STB_O &  ACK_I & /RST_I;                    +1

STB_O := /WAIT &  STB_O & /ACK_I          & /RST_I            +1
       + /WAIT & /STB_O          &  STB_I & /RST_I;
```

(a) Common-clock bridge for 3-Edge(AF) and 3-Edge(RF) cycles.



(b) Common-clock bridge for 4-Edge(TR) cycles.

Figure A-5. Bridge structures used for the handshaking efficiency study.

Figure A-6 identifies the service locations (indications) in each handshaking waveform. These are identified as 'E1-E4'. Note that both the three (3) and four (4) edge handshaking protocols result in identical service indications. The 4-Edge (TR) handshake also provides an indication of 'service satiated'. While 'service satiated' does have value in some situations, it is not required in the regular WISHBONE data transport structure and so has been omitted from it.

```
      Handshaking Nomenclature

        3-EDGE(AF)/3-EDGE(RF)

E1: MASTER Service Requested
E2: SLAVE  Service Starting
E3: BOTH   Service Complete

              4-EDGE(TR)

E1: MASTER Service Requested
E2: SLAVE  Service Starting
E3: MASTER Service Satiated
E4: SLAVE  Service Complete
```

Figure A-6. Service locations.

Unbridged Point-to-point Interconnection (Baseline Reference)

The MASTER and SLAVE interfaces are first evaluated over a point-to-point interconnection as a baseline reference. This produces the three relative timing diagrams shown in Figure A-7. There, each diagram shows its closed-loop control transport clock efficiency, which are summarized below.

Point-to-point Interconnection Blocked with Common-clock Bridges

The point-to-point test is repeated by blocking it with one common-clock bridge, and then two common-clock bridges in series. The two-bridge case creates three clock sub-domain regions which are identified as #0, #1 and #2. Figures A-8, A-9 and A-10 show the timing relationship for the two bridge cases using the 3-Edge (AF), 3-Edge (RF) and 4-Edge (TR) handshaking modes respectively. Each measures the closed-loop control transport clock efficiency (CE) using the graphical measurement technique described elsewhere in this report.

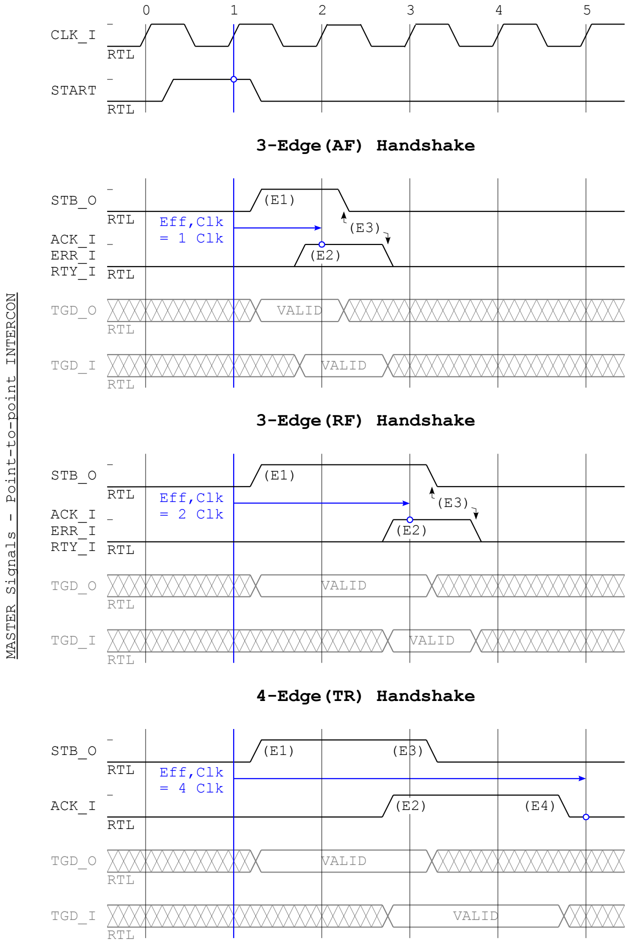


Figure A-7. Timing for the unbridged point-to-point interconnection (baseline reference).

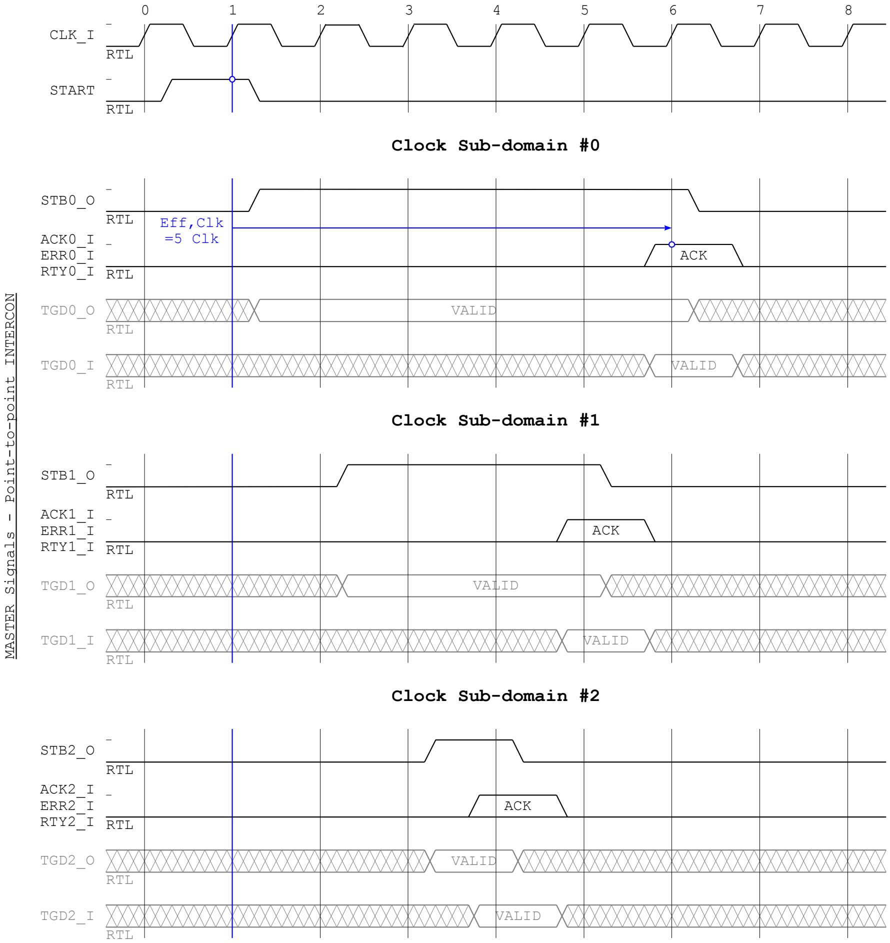


Figure A-8. Timing diagram for the 3-Edge (AF) cycle over a two-bridge interconnection.

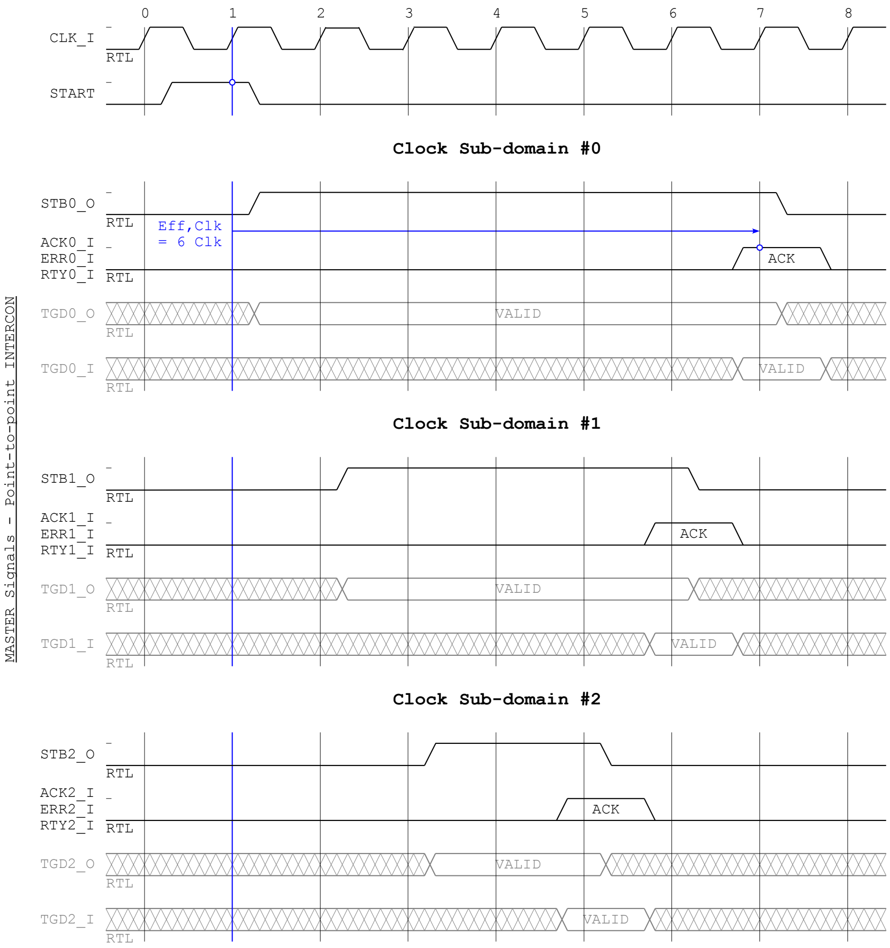


Figure A-9. Timing diagram for the 3-Edge (RF) cycle over a two-bridge interconnection.

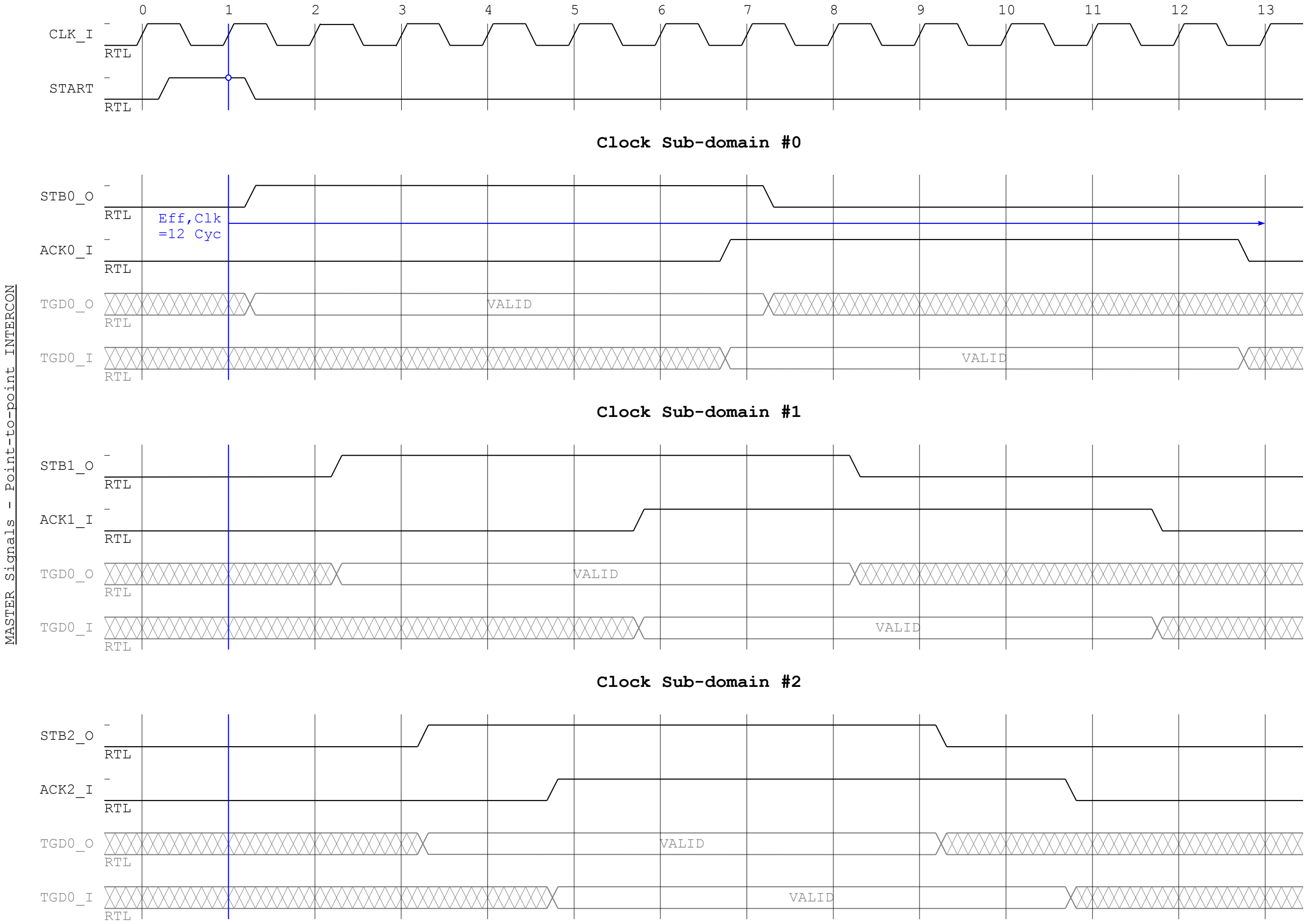


Figure A-10. Timing diagram for the 4-Edge (TR) cycle over a two-bridge interconnection.

Results and Conclusions

The closed-loop control transport clock efficiency (Eff, Clk) for the three blocking cases: (i) unblocked point-to-point interconnection (no-bridge baseline reference); (ii) point-to-point interconnection blocked with one common-clock bridge and (iii) point-to-point interconnection blocked with two common-clock bridges, are summarized in Table A-1.

Table A-1. Handshake comparison under three bridge conditions.

Handshaking Type	No Bridge Eff, Clk	1 Bridge Eff, Clk	2 Bridge Eff, Clk
Standard WISHBONE: 3-Edge (AF)	1	3	5
Standard WISHBONE: 3-Edge (RF)	2	4	6
Traditional: 4-Edge (TR)	4	8	12

In conclusion, the standard WISHBONE 3-Edge (AF) handshaking provides the best (lowest) clock efficiency in bridge applications.

A corollary to this conclusion is that the SLAVE interface on an automatic bridge will always insert at least one wait-state. This is because the SLAVE interface always asserts a terminal signal using a registered output, and thus conforms to the definition for the 3-Edge(RF) handshaking signal.

Bibliography

Peterson, Wade D. (2001a) SLC1657 8-Bit RISC uC Core Technical Reference Manual. Silicore Corporation, 2001, 2003, 2005. 194 pages < manl1657.pdf >

Peterson, Wade D. (2001b). WISHBONE Public Domain Library for VHDL. Silicore Corporation, 08 OCT 2001. 23 pages + library files. < manual: wishlibs.pdf; library: WBVHDLIB.zip >

Peterson, Wade D. (2008) The Creation of Interoperable Integrated Circuits Using Clean Room Design Practice. Silicore Corporation, 23 Feb 2008. 118 pages < CleanRoomDsnPrac.pdf >

Peterson, Wade D. (2010) WISHBONE SoC Proposed Rev. C (Major) Amendments (Unpublished)[52]. Silicore Corporation, 17 NOV 2010. 231 pages < WishboneRevC_Amend_17NOV2010.pdf >

Peterson, Wade D. (2012) The Consolidated Dictionary of WISHBONE Terminology, First Edition (Unpublished). Silicore Corporation, 27 MAR 2012. 132 pages < WbDictionary_27MAR2012.pdf >

WISHBONE Rev. A.0 (1999) Wishbone Interconnection for Portable IP Cores Specification, Revision A. Silicore Corporation, 16 JUN 1999. 46 pages < wbspec_a0.pdf >

WISHBONE Rev. B.0 (2001) WISHBONE System-On-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B. Silicore Corporation, 05 JAN 2001. 87 pages < wbspec_b0.pdf >

WISHBONE Rev. B.1 (2001) WISHBONE System-On-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.1. Silicore Corporation, 05 JAN 2001. 87 pages < wbspec_b1.pdf >

WISHBONE Rev. B.1 (2001+) WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.1 (Stewards Change Copy). Silicore Corporation, 09 AUG 2001. 103 pages < wbspec_b1_chgtrk.pdf >

WISHBONE Rev. B.2 (2001) WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.2. Silicore Corporation, 10 OCT 2001. 109 pages < wbspec_b2.pdf >

WISHBONE Rev. B.2 (2001+) WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.2 (Stewards Change Copy). Silicore Corporation, 30 NOV 2001. 123 pages < wbspec_b2_chgtrk.pdf >

Ω

[52] Deposited with the Registrar of Copyright, U.S. Copyright Office on or about 18 NOV 2010; USPS Certified Mail Receipt No. 7010 0290 0002 1159 4500. Copyright registration fee submitted by USPS Money Order No. 17620266947 in the amount of US $50.00; fee was accepted and cashed by (endorsement signature): LC COPYRIGHT '0 031 736 124 5'; Cancelled by: 'FRB CLEVELAND 12/28/2010'. Fee verification was requested on or about 16 MAR 2011 using USPS Money Order Inquiry (Form 5401), Ser. No. 17620266947.

> ***Secure Services Tech. v. Time and Space Proc., 722 F.Supp. 1354 (E.D.Va. 1989)***
> "Under the copyright laws, copyright registration is presumed to have occurred from the moment the owner of the copyrightable material delivers his application and filing fee to the Copyright Office." (722 F.Supp 1354, 1364)