11.2.2 SIMD4 Mode of Operation

With a register mapping of <src0> to doublewords 0-3 of *r2*, <src1> to doublewords 4-7 of *r2* and <dst> to doublewords 0-3 of *r3*, the example 3D graphics API Shader instruction can be translated into the following GENX instruction:

```
add (4)   r3<4>.xyz:f   r2<4>.yzwx:f   r2.4<4>.zwxy:f   {NoMask}
```

Without diving too much into the syntax definition of a GENX instruction, it is clear that a GENX instruction also takes two source operands and one destination operands. The second term, (4), is the execution size that determines the number of data elements processed by the SIMD instruction. It is similar to the term SIMD Width used in the literature. Each operand is described by the register region parameters such as '<4>' and data type (e.g. "*:f*"). These will be detailed in Section 11.3. The instruction option field, {NoMask}, ensure that the execution occurs for the execution channels shown in the instruction, instead of, possibly, being masked out by the conditional masks of the thread (See Instruction Summary chapter for definition of *MaskCtrl* instruction field).

The operation of this GENX instruction is illustrated in Figure 11-2. In this example, both source operands share the same physical GRF register r2. The two are distinguished by the subregister number. The source swizzles control the routing of source data elements to the parallel adders corresponding to the destination data elements. The shaded areas in the destination register r3 are not modified. In particular, doublewords 4-7 are unchanged as the execution size is 4; doubleword 3 is unchanged due to the destination mask setting.

In this mode of operation, there is only one program flow – any branch decision will be based on a scalar condition and apply to the whole vector of four elements. Option {*NoMask*} ensures that the instruction is not subject to the masks. In fact, most of the instructions in a thread should have {*NoMask*} set.

Even though the execution only performs four parallel add operations, the GENX instruction still executes in 2 cycles (with no useful computation in the second cycle).

Figure 11-2. A SIMD4 Example