SC100 Fetch Set and Program Bus Behavior

StarCore LLC
Kevin Shay
September 3, 2003
Rev 1.0

The following is a programming level discussion of fetch set buffer behavior with respect to program bus activity on the SC100 architecture. It is intended for understanding behavior from a programming model prospective and does not exactly detail the inner workings of the pipeline.

Section 5.1.1.1 of the SC140 Core Reference Manual discusses the Pre-Fetch and Fetch stages of the SC100 architecture. The Fetch stage reads a fetch set from program memory into the core for VLES parsing and processing in the Dispatch stage. Fetch sets are 128-bits (8 16-bit words) in length and accesses to them over the program bus are aligned on 8 word bounds. Because the SC100 architecture doesn't restrict the alignment of the start of a VLES instruction group, a VLES can start at any word in a fetch set and extend over the 8-word fetch set boundary. To allow proper dispatch without a timing penalty of instructions in a VLES that crosses a fetch set boundary, the Fetch stage contains two 128-bit fetch set buffers. For proposes of discussion we will refer to the two buffers as the primary buffer and secondary buffer.

The start of the current VLES used in the Dispatch stage is always found in the primary buffer. If the VLES is found to extend beyond the end of the primary buffer, the Dispatch stage continues to parse using the secondary buffer. In this case or if the VLES ends on the last word of the primary buffer, the primary buffer's contents are marked as invalid for the next Fetch stage to resolve. Note that VLES grouping rules limit a VLES to 8 words. Thus, a VLES extending beyond the primary buffer will never extend beyond the secondary buffer. It is also important to note that if a VLES extends beyond the primary buffer but the secondary buffer is invalid, a one-cycle delay is imposed while the remaining portion of the VLES is loaded into the secondary buffer.

The Fetch stage must resolve invalid buffer states once Dispatch is complete and instruct the Pre-fetch stage for data. The following are the valid cases and the action used to resolve:

1. Primary invalid, Secondary valid – Secondary buffer contents moved to primary buffer. Fetch read access requested and fetch set placed in Secondary buffer. Both buffers marked as valid.
2. Primary valid, Secondary invalid – Fetch read access requested and fetch set placed in Secondary buffer. Both buffers marked as valid.
3. Primary invalid, Secondary invalid – Fetch read access requested and fetch set placed in Primary buffer. Primary marked as valid. Secondary marked as invalid. Next cycle will resolve this state (unless the fetch set was an 8 word VLES in which case Dispatch will mark the as invalid also).
4. Primary valid, Secondary valid – No fetch action required. No program activity will be seen.

Therefore, the behavior of the pipeline is such that it will always try to have fetched one fetch set ahead of the fetch set with the start of the current VLES. Note that change-of-flow instructions (and exceptions) mark both the primary and secondary buffers as invalid.

There are several observable implications of the dual fetch buffer behavior:

1. Because code is typically not composed of only 8 word VLES, program accesses are not in general needed every cycle. If code is composed only of VLES with single, 16-bit instructions, program accesses only occur every 8th VLES.
2. If a change-of-flow (or exception) destination is to a VLES that extends over a fetch set boundary, a one-cycle stall is introduced.
3. Short loops can use dual buffers to implement looping behavior that does not require program bus activity. For short loops, no program bus activity will be present. This also explains why short loops are limited to 2 VLES.
4. Program bus activity cannot be used to isolate real-time debugging information without some post processing. Activity on the bus is typically a couple of VLES ahead of the current execution point.