Chapter 2
Program Control

This chapter provides overview on the new concepts of the program control in the SC2000. This is a preliminary description of the proposed features to be implemented in the SC2000 family in order to reduce the deeper pipeline latency impact. The complexity of design versus gain of all new ideas will be examined and later define the actual implementation method.

In the following sections three subjects are raised to be considered for the architecture of the SC2000 program control:

1. Methods to reduce COF latency.
2. Hardware loops support (for both legacy and new code).
3. Precise exception.

Before starting, here are several concepts:

- Most likely that the fetch set will be defined as 256 bits for the SC2400 derivatives, and 128 bits for the SC2200 derivatives, which result in a 256/128 rows in the PDU (Program Dispatch Unit) queues.
- The SC2000 family will have pre-fetch and target buffers (for instruction pre-fetching of both continues code and branch target) along with HW loop buffers (aimed mainly at power reduction during short loops). One of the design targets is to share the hardware loop mechanism with the pre-fetch buffers.

2.1 Methods to Reduce COF Latency

The SC2000 is targeted to ramp-up frequency with 10 pipeline stages. As it is described in previous chapters, program memory access from address to data fetch has a 3 cycles latency, and the dispatch stage is separated from the decode stage. This by itself increase the latency cycles penalty, compared to the SC1000, from 2 cycles to 4 cycles for JMP (Jump) instruction, and from 3 cycles to 4 cycles for BR (Branch) instruction. See example in Table 2-1:

Table 2-1. JMP Latency

Operation	Instruction Cycle									
	1	2	3	4	5	6	7	8	9	10
Program Address			Target							
Read Memory				Target						
Fetch					Target					
VLES Dispatch	JMP					Target				
Decode		JMP					Target			

The shaded cycles are the latency of the JMP instruction. Notice that delayed COF instruction or lpmarkB execute 1 or 2 instructions in these cycles. Table 2-2 lists COF cycle count (latency + 1) differences between the SC1000 and the SC2000 cores.

Table 2-2. COF Cycle Count

COF Instruction	SC1000 Cycle Count		SC2000 Cycle Count[1]	
	Taken	Not Taken[2]	Taken	Not Taken[2]
BRA, BRAD, BSR, BSRD	4		5	
BF, BFD, BT, BTD, JF, JFD, JT, JTD	4	0	5/7[3]	0
JMP, JMPD, JSR, JSRD	3		5	
RTE, RTED, RTSTK, RTSTKD, RTERI, RTERID	5/6[4]		8	
RTS, RTSD	3/5/6[5]		5/8[6]	
BREAK, CONT, CONTD	4		5	
SKIPLS	4	0	5	0
TRAP	4/5[7]		5	

1. Using a pre-branch or a pre-jmp instructions can reduce the cycle count in some cases.
2. This is relevant only for conditional COF. COF which are not conditional are considered always as Taken.
3. In worst case extreme scenario, cycle count may reach 7 in the rare case of conditional COF following T-bit update in DALU, and most of the execution sets before/after the COF instruction have maximum length.
4. The additional cycle in the count is dependent on because the shadow SP is not valid.
5. If RAS is valid then cycle count is in minimum value, else the cycle count depends on the state of the shadow SP (see [4])
6. If RAS is valid then cycle count is 5.
7. The cycle count is dependent on the machine state and may vary.

The COF Latency can be reduced significantly if target is pre-fetched. For this purpose the SC2000 introduces new pre-fetch instructions and a pre-fetch buffers mechanism.

2.1.1 Pre-Fetch Buffers

The SC2000 as the SC1000 uses a variable length execution sets (VLES) and a fixed length program fetch sets. This requires a mechanism of sequencer and storage in the PDU. For a continues busy pipeline operation, i.e. there is no queue starvation, there is a need for 5 fetch-sets storage in the queue.

The SC2000 uses additional queues, pre-fetch buffers, to reduce COF latency. With additional pre-branch or pre-jmp instructions, the whole mechanism can reduce latency to zero, by an early fetch of a target address. These buffers may also be used as a target cache for a re-visited target, e.g. a subroutine that is used more than once.

2.1.2 Conditional Branch and Static Prediction

In the SC1000 there are jump and branch instructions which are executed depending upon the state of the T-bit in the status Register (SR). The SC1000 gives advantage to the not taken case of conditional COF instruction, executing them with zero latency. For example JT instruction when the T bit is not set, result no latency in the pipe.

Though the pipe of SC2000 is deeper, the SC2000 maintains zero latency for a not taken conditional COF instruction. In addition the SC2000 introduces also a zero latency for taken conditional COF and for unconditional COF instructions, using a smart pre-fetch instruction with a 'taken' prediction flag. Because of the SC2000 deeper pipeline, instruction should be able to be canceled after they were dispatched (speculative execution). For example in the following case demonstrating JT after CMP instruction:

Table 2-3. JT instruction

Operation	Instruction Cycle									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Program Address			A	T	T1	A1				
Read Memory					T					
Fetch						T				
VLES Dispatch	CMP	JT	i1	i2	i3	i4				
Decode		CMP	JT	i1	i2	i3				
Address Generation			CMP	JT	i1	i2				
aCcess Memory				CMP	JT	i1				
Execution					CMP	JT				
Mac						CMP				
T-bit		-	-	-	-	valid				

In the example above, instructions i1-i4 (the shaded cycles) need to be killed or rewind if the resolution of the CMP (Compare) instruction is TRUE. As described the latest stage that is necessary to kill is the aCcess Memory stage. It means that the access to data memory that was started already by sending the address, should be canceled. All the post-modifications or arithmetic of the address registers, should be restored.

Note that the timing of the Taken (with prediction and a use of a pre-fetch instruction) and the Not-Taken cases is the same in most cases, exclude cases of prediction Taken with a resolution of Not-Taken where the code section near by the COF instruction, contains very long execution sets.

Lets consider the following example:

Table 2-4. JT instruction in a Hardware Loop

Operation	Instruction Cycle									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Program Address			A	T	SA	SA+1				
Read Memory					T	SA				
Fetch						T				
VLES Dispatch	CMP	JT	i1	LA-2	LA-1	LA				
Decode		CMP	JT	i1	LA-2	LA-1				
Address Generation			CMP	JT	i1	LA-2				
aCcess Memory				CMP	JT	i1				
Execution					CMP	JT				
Mac						CMP				
T-bit		-	-	-	-	valid				

A hardware loop is considered in the above case. Restriction L.C.3 + L.C.5 “A Bc or Jc is not allowed at LPA-1 or LPB-1 of a loop”, force to have i1 between the JT instruction and LA-2, where the lpmarkB is encoded. We see that if we need to cancel the lpmarkB actions because of a TRUE resolution of the T-bit, then it would better for us to write the hardware loop flags not earlier than Address Generation stage.

2.2 Hardware Loop Support

The SC2000 keeps the efficiency of the SC1000 loop execution. The SC2000 core keeps the same programming model of the SC1000 with the hardware loop mechanism. It supports up to 4 nesting levels with 4 SA (Start Address) registers, and 4 LC (Loop Counters).

In order to reduce power consumption, the SC2000 executes 5 to 8 fetch sets[1] of the 1st or 2nd inner nesting levels from an internal buffers. In this way the core does not need any program memory fetches while executing the most inner nesting levels.

Long loops which are not using the internal buffers need a special care.

In the SC1000, change of flow at the end of a long loop from the LA (Last Address) to the SA (Start Address) is done using lpmarkB instruction, located at LA-2. lpmarkB execution cause the program to perform a JMP-like instruction with 2 delay slots, causing a zero latency execution (see latency numbers in Table 2-1). Since the SC2000 architecture has 3 cycles latency in this case, a naive implementation will cause a single stall (latency) cycle on each cycle of the HW (which is unacceptable).

The SC2000 architecture offer two solutions to avoid this penalty:

For a binary backward compatible code, with lpmarkB, a dynamic branch prediction buffer will guaranty zero latency of loop branch in all cycles but the first one (which will have one cycle penalty).

For new, re-compiled or re-assembled code, long loops will have lpmarkA instruction at address LA, with a special pre-fetch instruction located at LA-3, guarantying zero latency in all cases.

1. Final number is TBD

2.3 Precise Exception

In a precise exception the machine stops execution and jumps to code section pointed to from the exception table. The term precise is defined such that the exception timing is guaranteed to be synchronous with the instruction execution. I means that the core need to kill the current execution set and following ones, and to be able to restore the status when exception is returned from.

The SC2000 is support precise exception with the restriction that no use is done with lpmarkB and COF with delay slots[1]. Therefore precise exception is supported only for new code compiled/assembled code, following the rule of no delay slots in it.

1. Supporting precise exeption in an un-interrupted sequence as COF with delay slot and/or lpmarkB COF is much more difficult.