ISSN 2278 – 0211 (Online)

Improved Granularity of HW/SW Co-Simulation by using Pipelined Instruction Set Simulator

Ritika Chandan
M.Tech Student, Department of Information and Technology
Chandigarh Group of Colleges, Landran, Punjab, India
Sachin Majithia
Assistant Professor, Department of Information and Technology
Chandigarh Group of Colleges, Landran, Punjab, India
Vivek Sharma
VTI VPTeam, Intel Mobile Communications Pvt. Ltd., Bangalore, Karnataka, India

Abstract:
Processor simulator models are the integral part of nearly all modern SOC/ASIC virtual platforms, as they let the software development team to progress without the dependency on the silicon tape-out time. Important considerations while developing an Instruction Set Simulator are speed and accuracy, which generally comes at the cost of decreased granularity of the different stages of instruction life cycle. Mainstream processor ISS virtual models do not implement the pipelines, as it's a bottleneck to the performance and increases code complexity when implemented in the languages as C, C+. This eradicates ISS model's scope for system level program development like compiler design and architecture exploration of new processors, limiting its use only for pre-silicon software testing. In this paper we present a novel technique to implement pipelines using multi-threaded SystemC high level hardware modeling language, that provides better granular accuracy and enhanced instruction execution performance efficiency to the processor ISS over a similarly modeled no pipeline version of the ISS. Classic RISC 3 stage pipeline is taken as a reference for this design and implementation.

Key words: *Pipelining, SystemC modeling language, ARM Instruction Set Simulator, Instruction Life-cycle Granularity, Virtual Platform, Performance Evaluation, Parallel Architectures, VCD Analyzer, and Branch Delays*

1. Introduction

Virtual systems are the key to success for any new SOC/ASIC design. They give the ability to verify functionality, algorithms and design logic without going into a time consuming tape-out process. They are also used to develop necessary software before the actual hardware hits out of the semiconductor fabrication unit, which decreases the time to market of the product and hence gives it a competitive edge. Pipelining is an integral part of any current generation processor, but not so much for their virtual models, maybe because there is no true parallel processing available in the languages generally preferred to model such high level virtual simulators.3-stage pipelining requires three instructions to be going through three different operations (fetch, decode and execute) at the same time when the pipeline is full, these stages are as follows:
Stage-1: Fetch Instruction - processor fetches the instruction from the memory subsystem, the instruction address is provided by the Program Counter register, i.e. incremented by the address increment method to fetch the next instruction when required by the processor.
Stage-2: Decode Instruction - processor now decodes the machine instruction which was fetched in stage-1. The decoding is done by the instruction decoder which reverses the instruction encoding protocol to identify the exact type of instruction, a sample instruction encoding scheme used for ARM data processing instructions is described in Table-I. The value of the fetched 32 bit instruction is matched with each of the instruction encoding field to get the type of instruction and then further action can be taken in the execution stage. The only time ALU may get utilized in decode stage is at the time of branch instructions, when the offset address to jump to, is calculated in the decode stage itself, which may lead to pipeline hazards in the actual hardware while executing the software which is never tested on a pipelined virtual model of the target machine.

Stage-3: Execute Instruction – processor executed the decoded instruction in this stage and any general purpose register as well as program status registers (CPSR bits) involved in the instruction execution along with the processor internal states are updated. This also includes the memory load and store operation, or program counter updates if there is any branch or the destination register for the executed instruction is PC. The instruction which goes through this life-cycle will take in general 3 processor clock cycles, assuming that each stage takes exactly 1 cycle long to finish.
This is clear that the ISS model need to have a parallel processing ability to model such a feature of concurrent notion of time, for simulating parallel processes of hardware. This feature is implemented and verified very thoroughly in SystemC modeling language and hence becomes the language of choice for implementation of the pipeline logic in a virtual model. The three stage pipeline logic demands that for a set of non-branched instructions, at a time T, instruction N should be in execution stage, instruction N+1 should be in the decoding stage and instruction N+2 should be in the fetch stage. In SystemC we may implement this by the use of thread processes SystemC is specifically designed to be used as a virtual platform modeling and testing language, some major components of SystemC is the ability to accurately model input/output ports, communication channels, functional algorithms of hardware - processes (method and threads), etc.

2. Problem Statement

Mainstream processor ISS virtual models do not implement the pipelines, as it’s a bottleneck to the performance and increases code complexity when implemented in the languages as C, C++, which are the language of choice for fast models of processor, but thisreduces the granularity of instruction lifecycle from its major three stages (Fig. 1), To only a single stage in the ISS (Fig. 2).

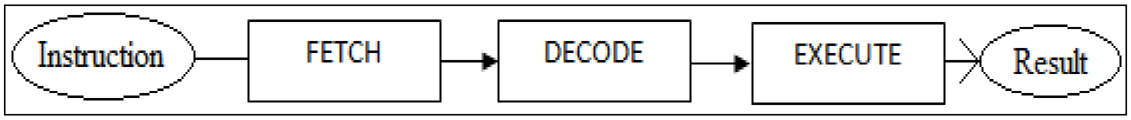

Figure 1: Three Stage Granularity of Instruction Life Cycle

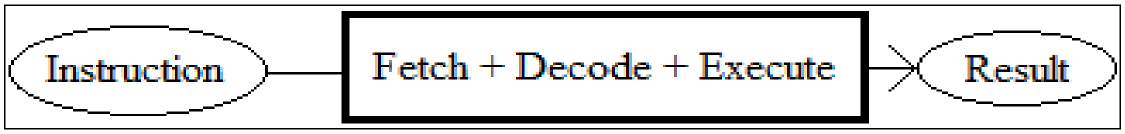

Figure 2: Single Stage Granularity of Instruction Life Cycle

Reducing the granularity results in eradication of the ISS model’s scope for architecture exploration of new processors and system programming for them e.g. development of new compiler design and verification, both of which are dependent on the granular accuracy of the target machine, limiting the current ISS model’s use only for pre-silicon software testing and execution.

3. Proposed Solution

Implement the 3 stages of pipeline in the ISS model, making its granular accuracy better and improving its instruction execution efficiency. Algorithm for implementation for 3 stage pipelines is described by the Fig. 3, which shows the communication flow between the multithreaded processes (Fetch, Decode, Execute, Address increment unit) and SystemC simulation kernel, it also shows the event notification and synchronization technique used by the threads to avoid out of order execution of the pipelines on multicore machines, giving a deterministic simulation behavior on every platform, in addition to this, the figure also describes the calling mechanism of each thread as per the language construct rules.
SystemC implementation of Fetch operation: This operation requires the instruction at the address stored in the program counter to be fetched from the memory subsystem. This communication part is handled by the SystemC-TLM blocking transport, for which a TLM simple initiator socket is implemented in the arm-vm module. Fetch function is registered as a SystemC thread, and all its functionality is implemented in an infinite while loop, which uses wait function to give control back to the scheduler, a wait of one clock cycle is used. As fetch is a thread and is automatically invoked by the kernel, it is also responsible for invoking the decode thread, which is sensitive on the event begin decode and will not get invoked unless the event is triggered.
SystemC implementation of Decode operation: This operation requires the fetched instruction to be checked for different encoding schemes of different types of instructions as per the ARM instruction encoding scheme, once identified, the thread labels the instruction type for the execute thread to use for its operation.

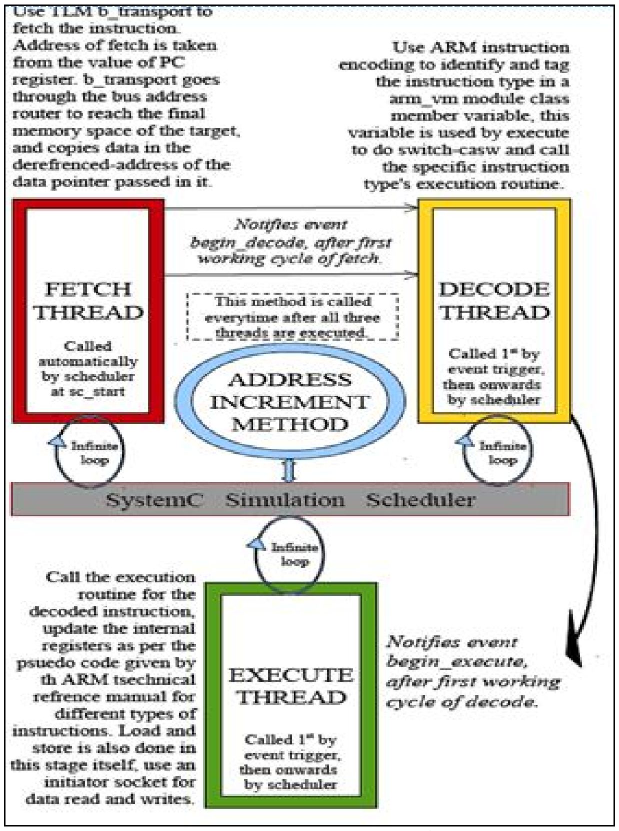
Use TLM b_transport to fetch the instruction. Address of fetch is taken from the value of PC register. b_transport goes through the bus address router to reach the final memory space of the target, and copies data in the derefrenced-address of the data pointer passed in it.
Use ARM instruction encoding to identify and tag the instruction type in a arm_vm module class member variable, this variable is used by execute to do switch-casw and call the specific instruction type's execution routine.
Notifies event begin_decode, after first working cycle of fetch.
FETCH THREAD
Called automatically by scheduler at sc_start
This method is called everytime after all three threads are executed.
ADDRESS INCREMENT METHOD
DECODE THREAD
Called 1st by event trigger, then onwards by scheduler
Infinite loop
SystemC Simulation Scheduler
Infinite loop
Call the execution routine for the decoded instruction, update the internal registers as per the psuedo code given by th ARM tsechnical refrence manual for different types of instructions. Load and store is also done in this stage itself, use an initiator socket for data read and writes.
EXECUTE THREAD
Called 1st by event trigger, then onwards by scheduler
Notifies event begin_execute, after first working cycle of decode.

Figure 3: Implementation Algorithm and Flow of SystemC 3 Stage Pipeline.

SystemC implementation of Execute operation: This operation uses the value of decoded instruction to call the execution routines for the specific instruction type. It does the memory read writes in case of load store instructions, and updates the register file of ISS if any register operation has been executed.
SystemC implementation of Address increment method: Once an instruction is executed, processor waits for the next instruction to execute, thus there must remain a mechanism to update the address of the next instruction before the fetch thread comes out of its wait state. This is done by using SystemC Method process, a method process acts just like a function call, but one which can be triggered by an event and also can be executed in a delta delay between the execution of next cycle of simulation. It increments the instruction address by adding four bytes to it, as each ARM mode instruction is 4 bytes long. One catch here is that if the program counter is updated in the execute state (e.g. by a branch instruction), then program counter must not increment its value, so that the updated instruction address remains valid.

4. Methodology

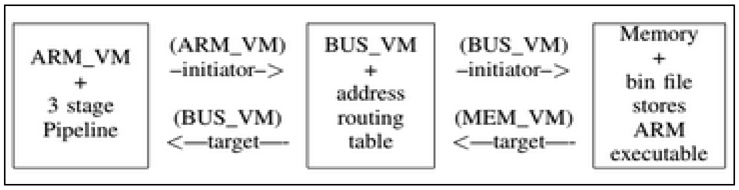
ARM_VM + 3 stage Pipeline
(ARM_VM) –initiator–>
(BUS_VM) <—target—
BUS_VM + address routing table
(BUS_VM) –initiator–>
(MEM_VM) <—target—
Memory + bin file stores ARM executable

Figure 4: ARM Virtual system used for simulating instructions

The virtual system used for simulating the ARM instructions to evaluate the performance of the virtual model of 3 stage pipeline, consisted of the subsystems as shown in Fig. 4. Where ARM_VM is the ISS, BUS_VM is the interconnect bus and Memory is the virtual model of memory where the instructions are loaded. Communication is performed using TLM2.0 protocol complaint blocking transactions. The whole virtual platform is simulated by SystemC kernel which goes through various phases as described in Fig. 5.

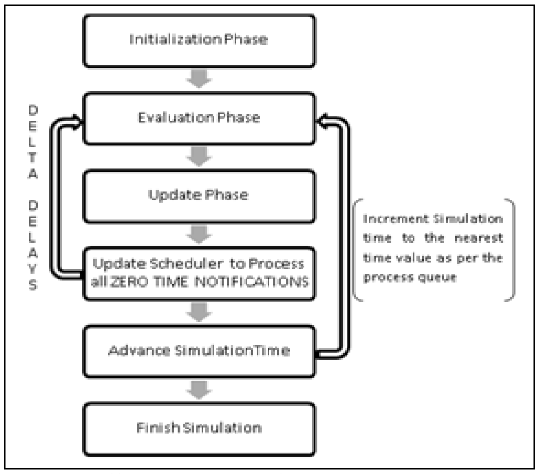
Initialization Phase
Evaluation Phase
Update Phase
Update Scheduler to Process all ZERO TIME NOTIFICATIONS
Advance SimulationTime
Finish Simulation
DELTA DELAYS
Increment Simulation time to the nearest time value as per the process queue

Figure 5: Different phases of SystemC simulation kernel

These phases are invisible to the user and the software instructions simulating over the virtual system and efforts are underway by the language working group to take advantage of host system's multi-core processors to give a realistic parallel processing solution for multithreaded simulations like that of pipelines

To analyze the execution of the instructions on the ISS, we also require a value change trace tool - GTKWave, which creates and updates a graph with the values of of all signals which it is tracing at the exact time of change event, As the ISS has provided signals to monitor the phases of instruction being executed, the instruction value is also displayed in the graph every time any change occurs. The value change graph of non-pipelined ISS model is given by Fig. 6, The graph displays the Time axis in nanoseconds, and the flow of instruction as it goes through different stages of its lifecycle. In this graph, it displays the instruction 0xE3A00010 being fetched at 0ns, at 1ns it goes into decode mode, at 2ns it goes into execute mode, at time 3ns, a new instruction 0xE3A01020 starts its own lifecycle. It is clear that no other instruction is being handled by the ISS while one is being processed.

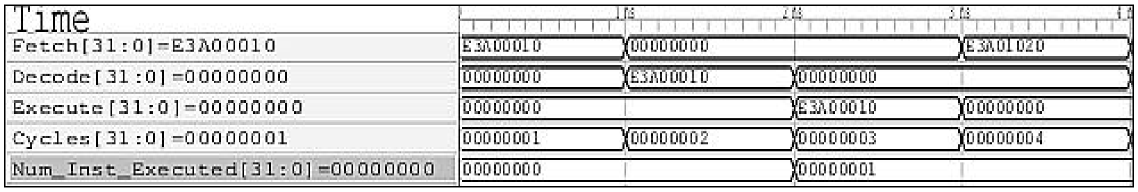

Figure 6: Single Stage ISS execution value change dump trace graph.

For the 3 stage pipelined ISS, the graph is given in Fig. 7, which displays that the instruction 0xE3A00010 being fetched at 0ns, at 1ns it goes into decode mode, and a new instruction 0xE3A01020 starts its fetch phase at the same time, at 2ns instruction-1 goes into execute phase, instructions-2 goes in decode phase and a new instruction-3 0xE3A02030 starts its fetch phase, and the ISS continues the concurrent phases of individual instructions, providing the necessary granularity of a three stage pipelined real processor.

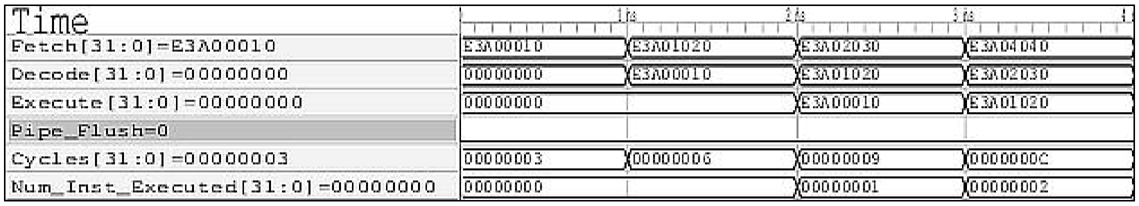
Time
Fetch[31:0]=E3A00010
Decode[31:0]=00000000
Execute[31:0]=00000000
Pipe_Flush=0
Cycles[31:0]=00000003
Num_Inst_Executed[31:0]=00000000
E3A00010 E3A01020 E3A02030 E3A04040
00000000 E3A00010 E3A01020 E3A02030
00000000 E3A00010 E3A01020
00000003 00000006 00000009 0000000C
00000000 00000001 00000002

Figure 7: Three Stage ISS execution value change dump trace graph

5. Conclusion

Analysis of the value change dump graphs generated by the GTKWave trace dump tool Fig. 6, and Fig. 7, shows that the pipelined ISS model executes the instructions concurrently in all three stages, and verifies the implementation of 3 stage pipeline in the ISS processor model, using the results given by the trace tool, we can conclude that the newly implemented 3 stage pipelined ISS virtual model had improved the granular accuracy of every instruction it executed, and thus giving a functionally accurate base for verification of hidden pipeline hazards in the software, which may remain undetected until the time of silicon tape-out and hardware porting, if the software is never executed on a granular pipelined ISS virtual model, and saving the expensive cost of debugging these hard to catch bugs in software at a later stage.

6. References

1. Alex Heunhe Han, Young-Si Hwang, Young-Ho An, So-Jin Lee, Ki-Seok Chung, "Virtual ARM Platform for Embedded System Developer", 2008.
2. AKohler and M. Radetzki, "A SystemC TLM2 model of communication in wormhole switched Networks-On-Chip" in Proc. of the Forum on Specification and Design Languages, 2009.
3. Alexander Peter, Ramesh K. Karne, Alexander L. Wijesinha, Patrick Appiah-Kubi"Transforming a Bare PC Application to Run on an ARM Device", 2013
4. Azucena, N, "High Level Implementation of an ARM7 Microprocessor with Multi-Core Capabilities", March 2006.
5. Christiensen C. Arandilla, Joseph Bernard A. Constantino, Alvin Oliver M. Glove Anastacia P. Ballesil-Alvarez, Joy Alinda P. Reyes, "High-Level Implementation of the 5-stage Pipelined ARM9TDM Core", 2010
6. Eriko Nurvitadhi, James C. Hoe, Timothy Kam, Shih-Lien L. Lu, "Automatic pipelining from transactional datapath specifications", Proceedings of the Conference on Design, Automation and Test in European Design and Automation Association, 3001 Leuven, Belgium, 2010, pp.10011004.
7. G.Schirner and R.Doemer, "Quantitative analysis of the speed/accuracy Trade-Off in Transaction Level Modeling", ACM Trans. Embed. Comput. Syst., vol. 8, no. 1, 2009
8. Joseph Yiu, "The Definitive Guide to the ARM CortexM3", Newnes publishers, AUG-2007.
9. Luna, A.,"Implementation of the Complete ARM7TDMI-S Instruction Set on a Debug-Capable Core", October 2008.
10. L.S. Indrusiak and O. M. dos Santos, "Fast and Accurate Transaction Level Model of a Wormhole Network-on-Chip with Priority Preemptive Virtual Channel Arbitration", in Proc. of the Conference on Design, Automation and Test in Europe, 2011.
11. Mitesh J. Limachia, Dr. Nikhil J. Kothari,"Modelling of ARM Cortex-M3 processor Core using SystemC", 2011.
12. Mona Safar, Magdy A. El-Moursy, Ashraf Salem, Mohamed Abdel salam, "TLM Based Approach for Architecture Exploration of Multicore Systems-on-chip", 2011-12.
13. ARM Limited, "ARMv7-M Architecture Application Level Reference Manual", ARM DDI 0405C.
14. SystemC Language reference manual 2.0
15. GTKwave analyzer reference manual 3.3