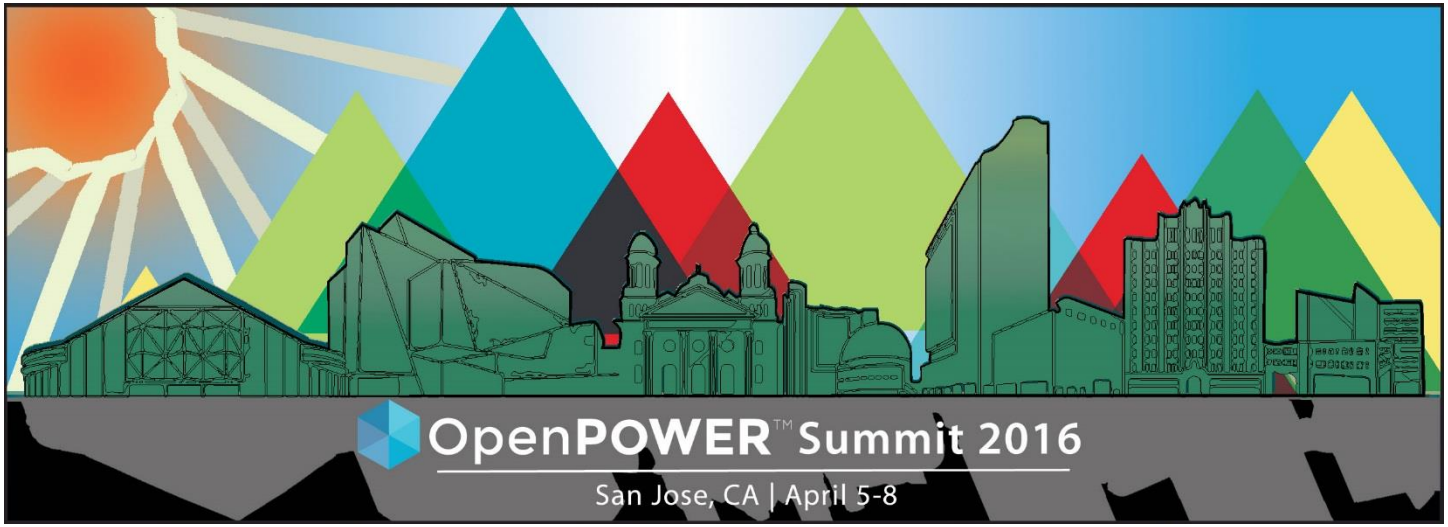
OpenPOWER™

A Methodology for Ensuring Compliance to the Power Architecture

Laurent Fournier, Senior Manager
Hardware Verification Technologies
IBM Research - Haifa

Revolutionizing the Datacenter



Join the Conversation #OpenPOWERSummit

What is a test?

Resource initialization

Instruction sequence

Predicted Results

```
* Genesys Pro Test File
* Produced on: Tue Oct 16 16:53:55 2001
* Def filename: /afs/haifa/home/adir/visit/01_5/seminar/407/tutorial.def
:
TEST 0
H  Seed: 8 System: PowerPC Version:  Format:
INITIALIZATIONS: DATA MEMORY (MEMORY)
D A7988120 64DA43CE                          * TABLE EA=43E4C120 WIMG=2
:
CLUSTER 0
PROCESS 0
INITIALIZATIONS: REGISTERS
R CR               0E97168E
R CCR0             002EE007
:
PHASE 0    INSTRUCTIONS
I E06F9004 FCE3982A * EA=C6339004 WIMG=3 fadd F7,F3,F19
I E06F9008 C0879C00 * EA=C6339008 WIMG=3 lfs F4,0x9C00(G7)
*   EA=43E4C120 (New) RA=A7988120 WIMG=2
I E06F900C FCC1282A * EA=C633900C WIMG=3 fadd F6,F1,F5
I E06F9010 FCA0C02A * EA=C6339010 WIMG=3 fadd F5,F0,F24
I E06F9014 C06D1889 * EA=C6339014 WIMG=3 lfs F3,0x1889(G13)
*   EA=A558E86C (New) RA=AE28E86C WIMG=3
EPILOGUE
* Begin macro Epilogue_Sequence
I E06F9018 4BFFD002 * EA=C6339018 WIMG=3 ba 0x3FFD000
*   EA=FFFFD000 (New) RA=FFFFD000 WIMG=0
:
RESULTS: REGISTERS
R CR               0E97168E
R CCR0             002EE007
:
END_OF_PROCESS
END_OF_CLUSTER
RESULTS: DATA MEMORY (MEMORY)
D A7988120 64DA43CE
:
END_OF_TEST
```

Definition and Goal

- What is an Architecture Compliance Suite (ACS) ?
  - Suite of tests showing that Architecture has been correctly followed
- What do we need it for ?
  - OpenPower standpoint: a Certificate
  - Developer standpoint: an assistance to confirm understanding

Common Pitfall

- The ACS is NOT a verification suite
  - It is developed independently of the microarchitecture (source of most bugs !)
- Simple example: the ADD instruction
  - Special cases: Zero and Overflow
  - Possible special microarchitecture case of increment by one is not checked (X = X+1)

The Architecture

- Instruction definition
  - Fixed Point, Branch, Load-store, Floating Point, etc.
- Mechanisms
  - Address Translation
  - Interrupt
  - Transactional Memory
  - Memory Model (MultiProcessing)
  - Etc.

ACS Constituents

- A test for each specified behavior
  - For instruction behavior, single-instruction tests
  - For mechanisms, can require multiple instructions
- Misinterpretations
  - Internal
    - A certain type of ADD instruction is NOT setting the overflow bit
  - External
    - Catch possible mistakes due to specific behavior different in other architectures (or in previous Power Architectures)

Instruction-driven scenario

Fixed-Point Arithmetic Instructions

- ***Setting CR0 (Condition Register Field 0)***

  Recording instructions: addic. addc[o]. subfc[o]. adde[o]. subfe[o]. addme[o]. ...

  Non-recording instructions: addic subfic addc[o] subfc[o] adde[o] subfe[o] addme[o] ...

Instructions tested	Observability preconditions	Expected result
I-3.3.9.CR0.1 Recording instructions set CR0 correctly when result is zero		
Recording instructions	First three bits of CR0 = 110	First three bits of CR0 = 001
I-3.3.9.CR0.4 Non-recording instructions do not change CR0 when result is zero		
Non-recording instructions	First three bits of CR0 != 001	First three bits of CR0 unchanged

Mechanism-driven scenarios

Interrupt Priorities

- ***Combinations of interrupts caused by the same instruction***
  - The scenario in this group should be tested for every one of the instruction groups A-J (Section III-S.6.8)
  - For each instruction group, the scenario should be tested for every pair I_{high}, I_{low} of interrupts caused by the instruction group, such that I_{high} has higher priority than I_{low}

Instructions tested	**Compliance conditions**	**Expected result**
III-S.6.8.Instr.1 For exceptions caused by an instruction in the designated group, interrupts are generated according to priority		
Representative of instruction group	• Exception I_{high} is created • Exception I_{low} is created • Interrupt I_{high} is not disabled • Interrupt I_{low} is not disabled	Interrupt I_{high} is generated

A Compliance Enigma

Storage Access Ordering (Section II.1.7.1)
Load: Instructions in Sections 3.3.2, 4.6.2 of Book I, lhbrx, lwbrx, ldbrx, lq ...
Store: Instructions in Sections 3.3.3, 4.6.3 of Book I, sthbrx, stwbrx, stdbrx…

Instruction sequence	Observability preconditions	Expected results
II-1.7.1.Access.1 Two processors (P1 and P2) perform stores and loads to/from two non-overlapping Caching Inhibited and Guarded storage locations (A, B)		
P1 P2 Store to A Load from B Store to B Load from A	• Storage Attributes of the A and B memory locations are set as follows: -- "Caching Inhibited" -- "Guarded" • Initial values of corresponding bytes of the source registers used by Stores and of locations A and B should be different.	If P2 loads from B a new value stored by P1, then P2 must load also a new value from A

Remarks and Numbers

- Around 300 pages
  - 5000 tests
    - 80% instruction driven tests
    - 95% single instruction tests
- Need to cope with various types of unspecified behavior:
  - Undefined
  - Optional
  - Multiple options
- ACS running failure
  - Force to rerun same tests ?

Summary

- An ACS is a huge suite of tests to verify architecture compliance
  - Not to be confused with verification suite
- Instruction definition and mechanisms are comprehensively targeted
  - Specified behavior
  - Misinterpretations
- Compliance is well tackled for instruction specification and most mechanisms
  - A few mechanisms (especially MP –related) are difficult to cover with the same rigor