Formal Verification on the TriCore CPU

Dr. Tim Blackmore
Formal Verification Tool Activities

1. Complete formal verification using ‘operational properties’
   - Targeting verification of a unit e.g. fetch unit
   - Completely verifies unit (data path, protocol, sideband signals)

2. Formal property checking
   - Usually developed by designers to increase confidence in release

3. Auto-generated properties
   - E.g. Code coverage closure, exclusions for fault simulations

4. Post-silicon debug
   - Generate corner-case waveforms to explore and characterise unexpected behaviour
Operation-based complete formal verification

- Methodology developed by OneSpin based on 3 steps
  1. Temporal description (SVA/PSL) of each ‘operation’ or transaction
     - E.g. read, write, read_and_write, no_read_or_write
  2. Operations are joined end-on-end by means of a key state
     - With this step have moved from individual operations to arbitrary sequences of operations
  3. For each operation all relevant outputs described in all cycles
     - So all outputs described for all possible sequences of operations
     - I.e. complete description of output behaviour

- RTL checked against operations with formal tool -> complete formal verification
- OneSpin tool incorporates formal completeness check
- Have also used methodology to write assertions for simulation e.g. protocol checkers
  - Protocol being checked in every cycle of simulation
  - Quick to develop
  - Find bugs missed by property-based protocol checkers
Simple Example - FIFO

- Property checking would concentrate on properties like
  - Correct write pointer increment
  - Correct read pointer increment
  - No overflow, no underflow
  - (E.g. ‘Using PSL/Sugar with HDL for formal and dynamic verification’, Ben Cohen, Srinivasan Venkataramanan, Ajeetha Kumari)

- Operational approach will completely describe
  - Write operation: write of data into FIFO and increment of write pointer (including write stall if full), correct maintenance of all other valid FIFO entries
  - Read operation: read of data from FIFO and increment of read pointer (including read stall if empty), correct maintenance of all remaining valid FIFO entries
  - Simultaneous read and write, and no read or write
  - Data checked into, through and out of FIFO
  - In practice would probably be combined into single operation
Bus protocol description in specification

- Read
- Write
- Read-Write
Mondrian
Property Assertions - Gaps
Experience

- Complete formal verification targeted by RFE
  - Higher effort units with higher potential for killer bugs e.g. Fetch, Memory Interfaces, Cache Controllers
  - Lower effort units e.g. ECC, ALU
  - Have formally verified complete CPU with this methodology (‘Complete Formal Verification of TriCore2 and Other Processors’, DVCon 2007)

- Experience is very good with respect to outcome
  - Finds bugs but rarely misses bugs
  - Deltas can typically be verified quickly

- Requires a level of expertise to apply
  - Around 40% of TriCore verification team have applied methodology successfully

- Can be relatively heavy-weight approach to formal verification
  - Effort estimates typically match design effort
    - Reduces with experience
    - Exceptions e.g. ECC encoder/decoder pair < 1 day
ENERGY EFFICIENCY
MOBILITY
SECURITY

Innovative semiconductor solutions for energy efficiency, mobility and security.