2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Automatic Partitioning for Multi-core HDL Simulation

Gaurav Kumar, Sandeep Pagey, Mohit Sinha, Manu Chopra, Cadence Design Systems

{gauravk,pagey,mohits,mchopra}@cadence.com

cadence®

Agenda

- Multi-core HDL simulation
- Partitioning and load balancing
- Automatic static partitioning
- Results
- Conclusions

1

Introduction

- Advances in processor and memory architectures attracting soft-ware applications towards multi-threaded implementations.
- HDL event simulation combines parallelism inherent in an HDL specification and multi-core CPU architecture.
- Commercial simulators currently mimic HDL concurrency through sequential execution only.
- Partitioning (Static or Dynamic) plays important role in multi-core HDL simulation.

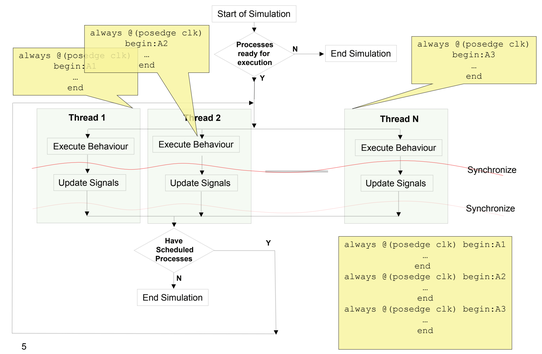
2

Multi-Core Simulation

- Parallelism inherent in HDL specification needs to be exploited
- "Simulate" parallel HDL specifications on multiple cores parallely
- Ensure that HDL semantics are honored
- Divide work to be done across parallel threads
- Synchronize regularly to maintain HDL semantics
- Many different algorithms/techniques

3

Multi-Core Simulation Cycle



5

4

Multi-Core Simulation

- Colored boxes indicate useful work between synchronization points
- Grey boxes indicate waiting by thread
- Reduce grey areas to maximize performance

5

Multi-Core Partitioning

- Partitioning defines the way work load is distributed across threads
- Dynamic partitioning refers to the fact that work load is distributed during execution
- Static partitioning refers to the fact that decisions about work load distribution are made before execution and do not change during execution
- This paper describes static partitioning
- Goal is to make sure that work-load is balanced to get max perfor-mance gain

6

Static Partitioning

- Annotate instance hierarchy with static information
  - No of sequential always block, UDP's
  - No of concurrent always block, UDP's
  - No of gates
  - Wire count, Reg Count
- Determine self and cumulative score of each node on tree, based on these parameters and type of design
  - Serial scan, Parallel Scan, Gate level
- Perform bin packing on nodes as close to root as possible
  - Entire sub-tree rooted at a node goes to one partition

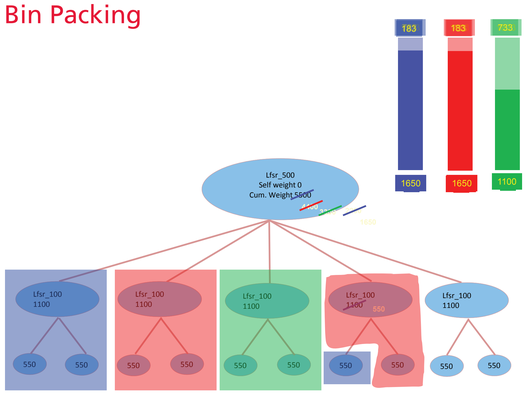
7

Bin Packing

- Unlike classic bin packing, element/object sizes are not fixed
- If a node cannot be packed in any bin, try to pack the child nodes
- When a child node gets packed into a bin, the parent node's size is reduced by the size of the child node
- When a child node gets packed into a bin, the parent node might get packed
- If a node cannot be fit into any bin, it is forced fit into a bin with minimum overflow

8

Bin Packing



9

Initial Results

Test	Type of design and Simulation	Number of Partitions	Performance Ratio using Multicore Simulation
Test 1	Gate Level	4	1.6X
Test 2	Gate Level	2	1.36X
Test 3	Gate Level	2	1.13X
Test 4	Gate Level	4	1.39X
Test 5	Gate Level	2	1.6X
Test 6	Gate Level	4	2.1X
Test 7	RTL	2	0.9X
Test 8	ATPG	2	0.5X

10

Conclusion

- Presented partitioning algorithm to balance work load
- Algorithm uses design parameters and knowledge of use-case to achieve performance gains.
- The algorithm has shown good results on a number of designs, but degradations on some designs.
- More design parameters need to be used and the weightings used for use-cases might have to be tuned.

11

cadence®

12