CAD Tools and Algorithms of Processor-based Logic Emulators

Amir Ali Yazdanshenas

Supervisor: Dr. Mohammed Khalid

October 2005

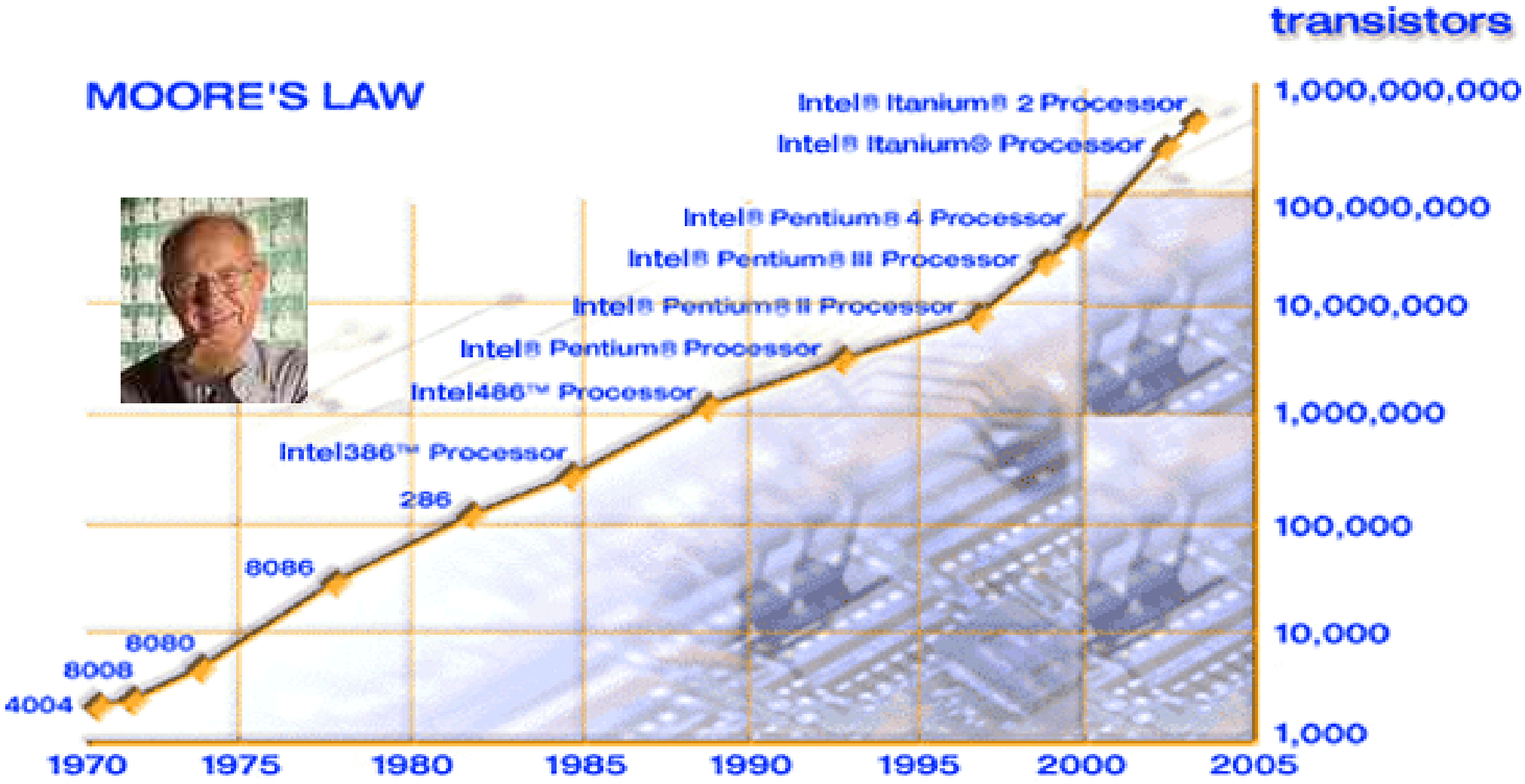
Department of Electrical and Computer Engineering
University of Windsor

OUTLINE

1- Introduction
   1-1- What is design verification and why we need it?
   1-2- Types of design verification
2- Logic Emulation Systems
   2-1- FPGA-based Logic Emulators
      - FBE Architectures and CAD Tools
      - Pros and Cons
   2-2- Processor-based Logic Emulators
      - PBE Architectures and CAD Tools
      - Pros and Cons
3- Comparison of PBEs and FBEs
4- PBE CAD Tools Flow By Example
5- Conclusion and Future Works
6- References

Introduction – Moore's Law

- Explicit: Number of Transistors x2 every 18 months (two years).[1]
- Implicit: Circuits become more *complicated* because more *functionality* and *performance* is needed for less *cost*.



Introduction – Design Process

Design: specifying design goals and characteristics

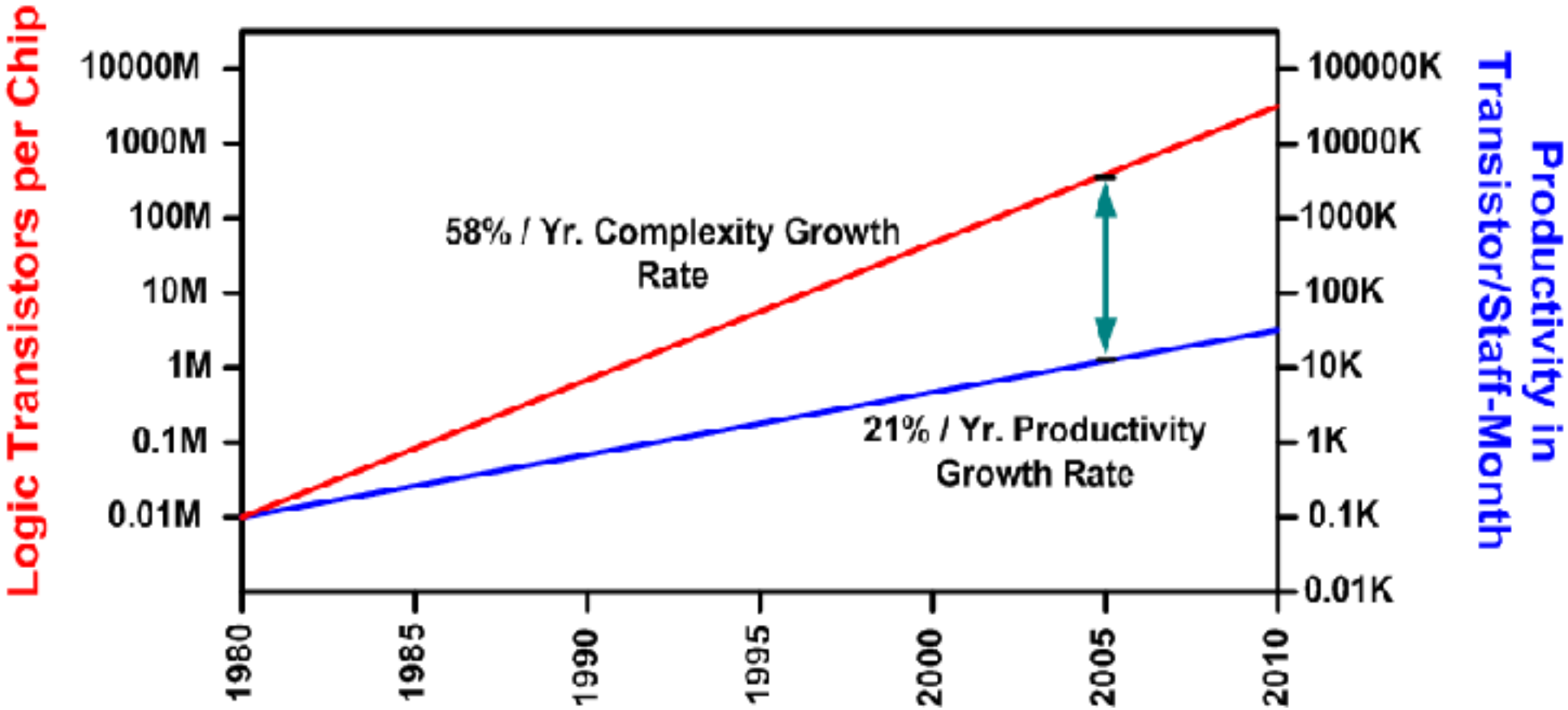
Implement: prototyping

Verification: verifying the correctness of design and implementation

Introduction – Verification Crisis[2]

Reasons:

- **Increased Gate/Flip-Flop (Circuit Elements) Count**
- **Increased Test-vector Count**



Introduction – Live Examples from Industry

- Intel: Processor design project [3]
  - "Billions of generated vectors"
  - "Our VHDL regression tests took 27 days to run."

- Sun: SPARC design project
  - "Test suites of 1500 tests, > 1 billion random simulation cycles."
  - "A server ranch of 1200 SPARC CPUs"

- Kodak: "Hundreds of 3-4 hour RTL functional simulation…."
- Xerox: "Simulation runtime occupies ~3 weeks of a design cycle."

CONCLUSION:

- **Verification methods have *failed* to keep up the pace with design trends.**
- **Verification is the *current* industry-wide issue.**
- **We need *faster* and *more efficient* verification methods!**

Introduction - What Kind of Verification?

Verification must be carried out at different stages of design process:

1- Design Verification

Is the design consistent with the original specification? Is this design what I wanted?

2- Implementation Verification

Is the implementation consistent with what I wanted?

3- Manufacture Verification (a.k.a. Test)

Did they build what I asked for?

Design Verification – Definition and Methods

Verification Goal:

Verify correctness of designs BEFORE fabrication to avoid functional errors and expensive/ time-consuming design re-spins.

Design Verification Methods: [4]

1- Logic Simulation

2- Hardware Accelerated Simulation

3- Logic Emulation

and few researchers add the followings too:

4- Formal Methods

5- Rapid Prototyping

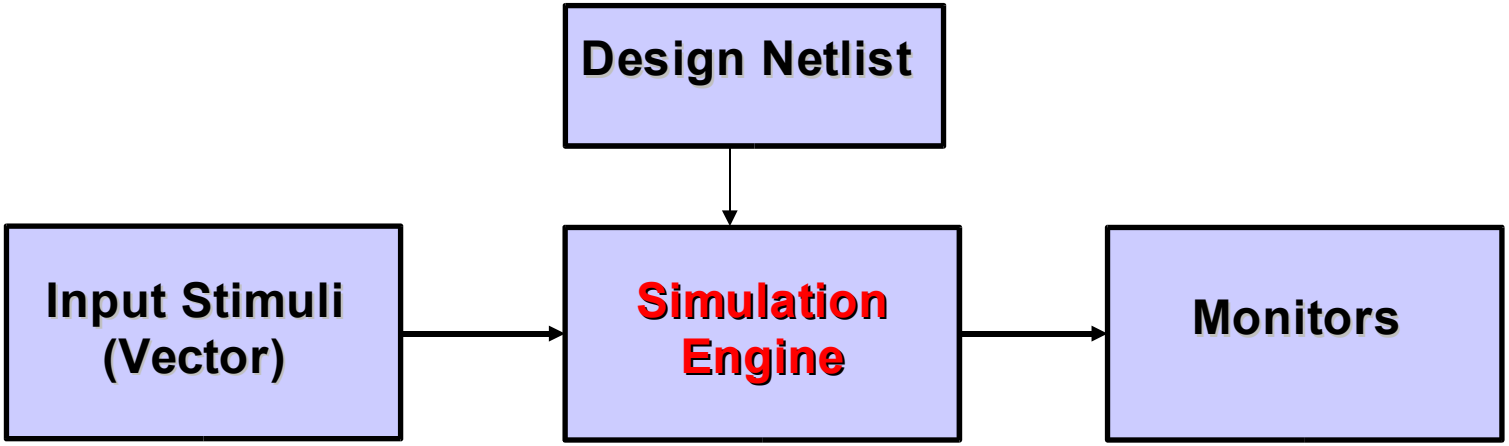
Design Verification - Simulation

[5][6]

1- Logic circuit is modeled and net-list is generated.

2- Inputs to the circuit (circuit's stimuli) are generated in the form of vector data files.

3- Software Engine (*Simulator*) takes the *model* and evaluates how the circuit behaves given those *vectors* as the input.



Finally:

- It is the designer's job to compare the outputs generated by the simulator with the expected correct outputs.

Examples: Circuit Simulator (e.g. Spice), Event-driven (e.g. Verilog-XL), Cycle-based (e.g. Cyclone VHDL), Hybrid (e.g. VSS)

Design Verification – Simulation Pros and Cons

Pros:

- Provide extensive capabilities in for modifying and debugging the design due to intrinsic flexibility in software.
- Easy to use
- Very cheap

Cons:

- Computation workload ≈ (Design Size)²

  => Not efficient and fast enough for designs more than 1M gates.
- Do not provide in-circuit-emulation (ICE)
- The accuracy of the simulation is very dependent on the accuracy of software models, input stimuli (i.e. depends on how well the designer knows his job!)

Result:

=> NOT comprehensive enough! some errors will remain undetected.

Design Verification – Hardware Accelerated Sim.

[7]

Definition:

Instead of using simulators on PC workstations, now designers can execute the simulation of their design models on a number of application specific parallel processors (multi-processor platforms).

Pros:

- Orders of magnitude faster than simulation $(\times 10^2 - 10^6)$
- Equipped with built-in test equipments such as signal generators…

Cons:

- Not usable in other applications.
- Could not perform ICE (In-circuit Emulation)
- Expensive

Design Verification – Other Methods

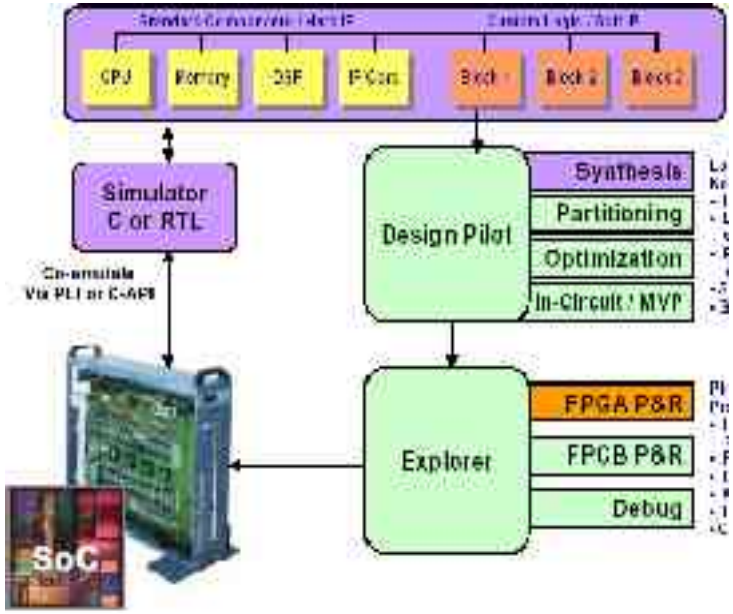
- Rapid Prototyping:

  Create a prototype of the actual design using available technology.

  (e.g. ASIC/IP cores + FPGAs)

  Pros: real hardware running @MHz & relatively cheap

  Cons: throwaway effort (ie. too ad-hoc) & limited debugging capabilities



Source: APTIX

- Formal Verification:

  1- Model checking: verify model's properties (e.g. enumerating all states in an FSM). Pros & Cons: comprehensive but can not handle large designs.

  2- Theorem Proving: Proving the theorems regarding properties of the model (e.g. verifying correctness of an floating-point multiplication algorithm).

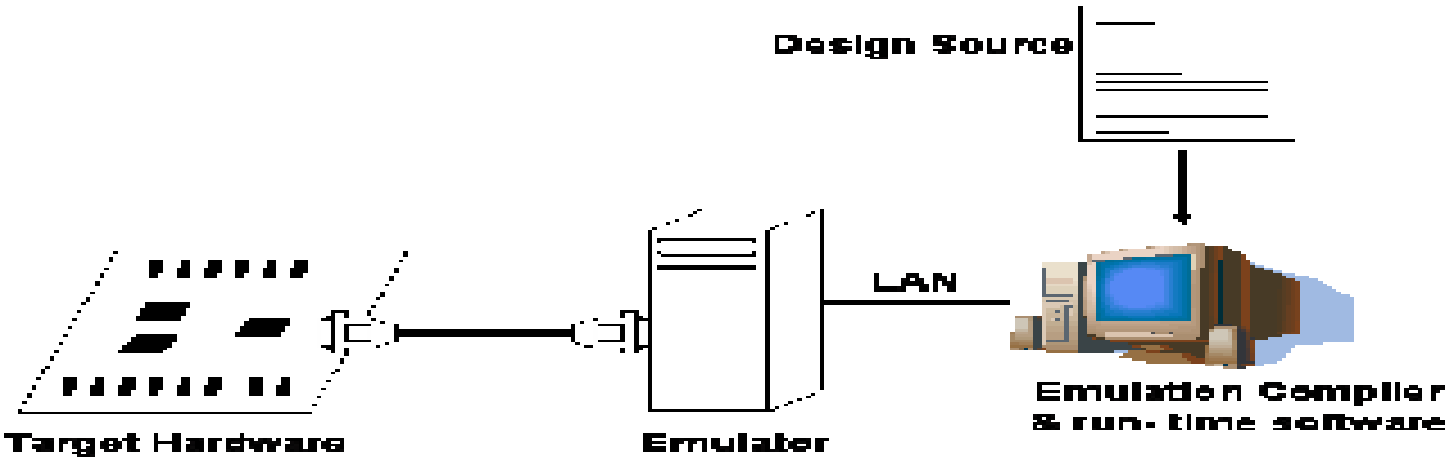
  Cons: Can not handle large designs due to computational complexity.

Design Verification – Logic Emulation [8][9][10] [11] [12]

A Logic Emulator (LE) is a *programmable* hardware system which can be programmed to emulate *large designs* at speed of multi *million cycles* per second (cps). LE functions just like the actual designed hardware without (before) fabricating the chip!

- The most efficient method for Design Verification
- Used by top semiconductor vendors such as Intel, nVidia, …

  - Emulator configured to as Pentium processor and connected to motherboard booted Microsoft Windows @500KHz.



Logic Emulators – General Architecture

Regardless of technology, LEs consists of following parts: [8]

1- Programmable hardware

programmable logic + programmable interconnection

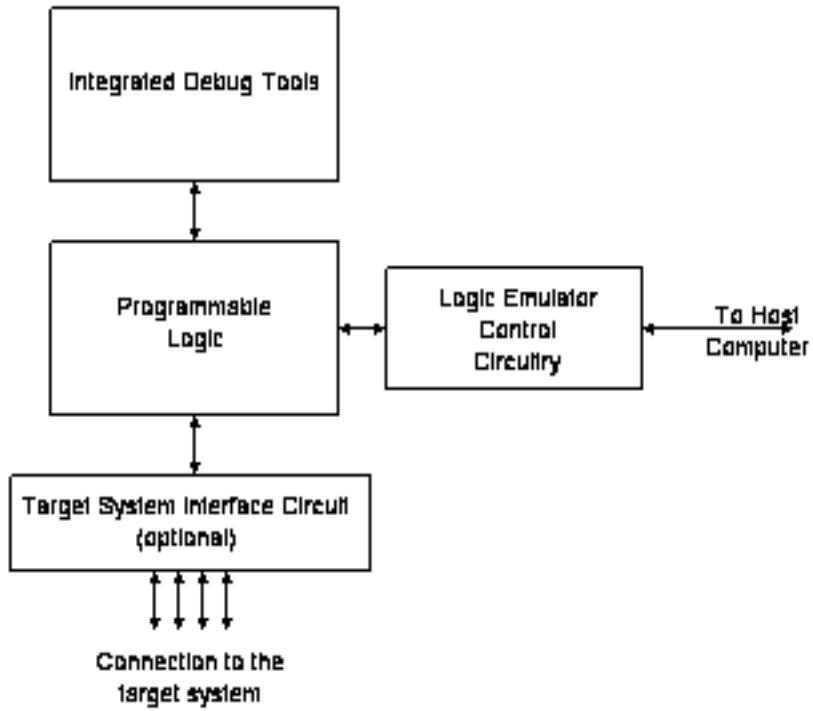
2- Compiler Software (i.e. EDA/CAD tools + GUI)

automatically translates designs into downloadable bit-stream into programmable hardware.

3- Integrated debug instruments (e.g. Logic Analyzers,…)

4- Integrated control hardware/software

5- Target hardware interface



Logic Emulators - Architectures[8]

Logic emulators are classified according to the technology used in their programmable hardware.

So far:

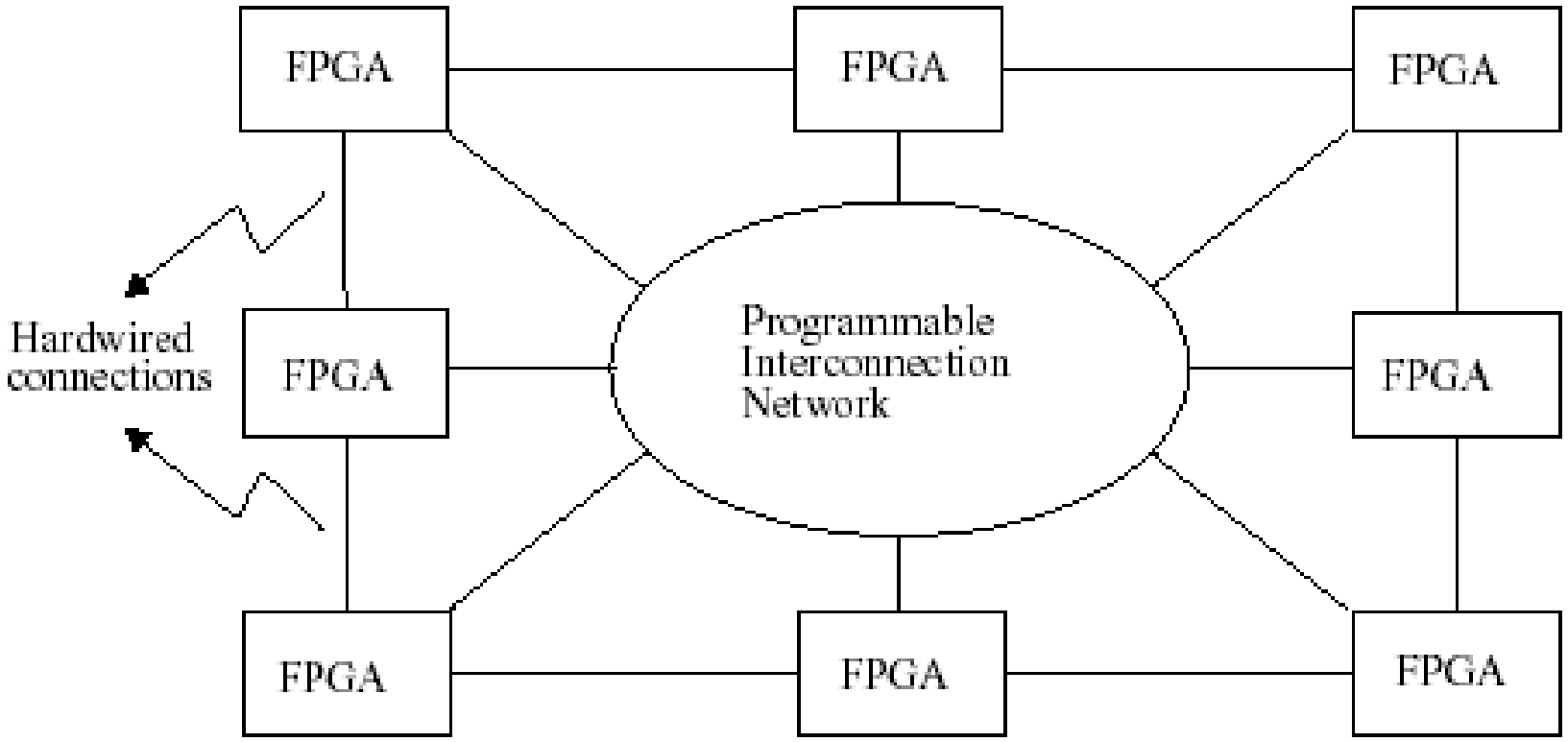
1- FPGA-based Logic Emulators (FBE)

2- Processor-based Logic Emulators (PBE)

3- Hybrid Logic Emulators (=1+2)

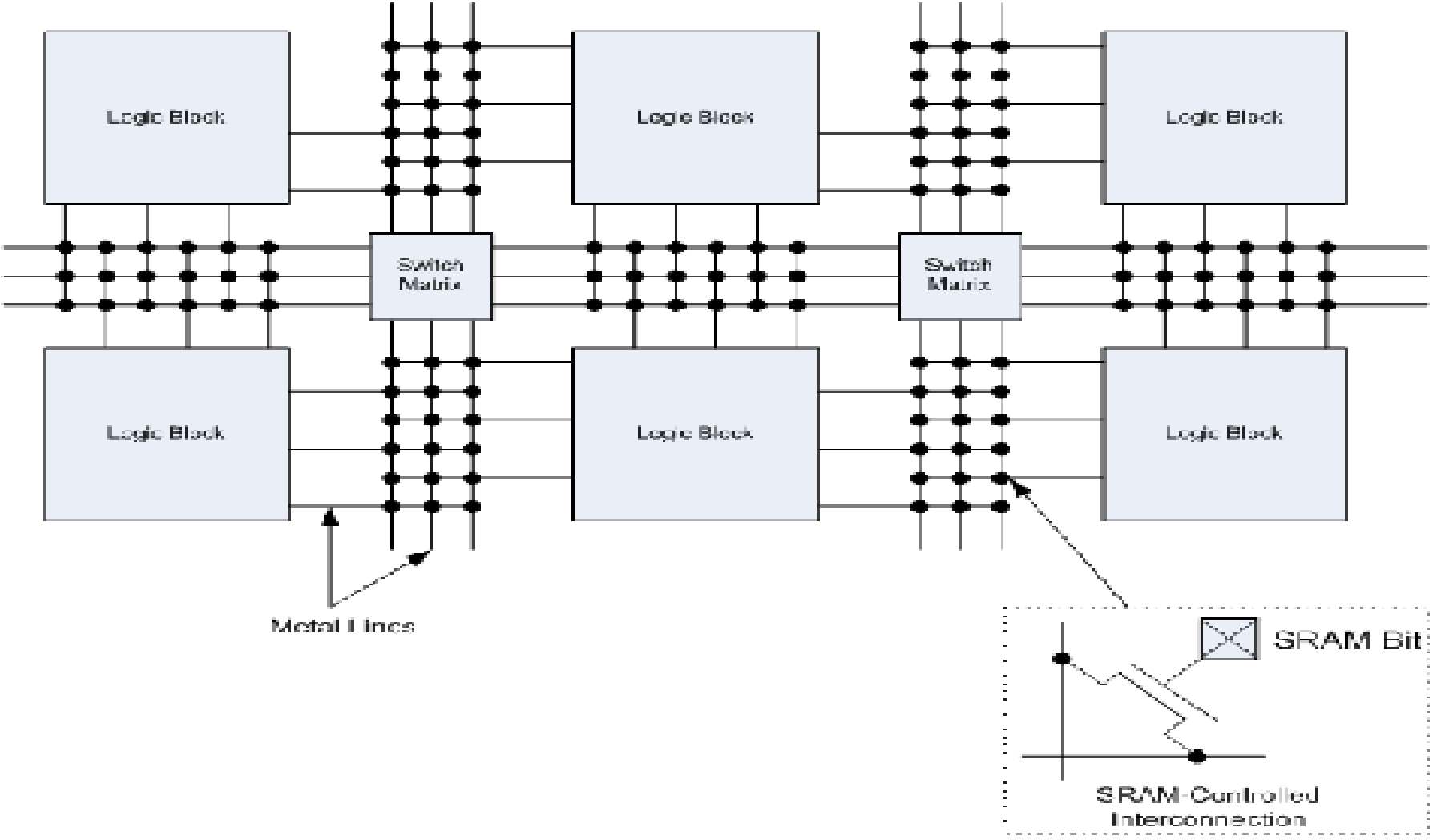
FPGA-Based Logic Emulators - Architecture[11]

- FBE consists of collection of multiple Field-Programmable-Gate-Arrays (FPGA) interconnected by hardwire and/or Programmable Interconnection Devices (PID).



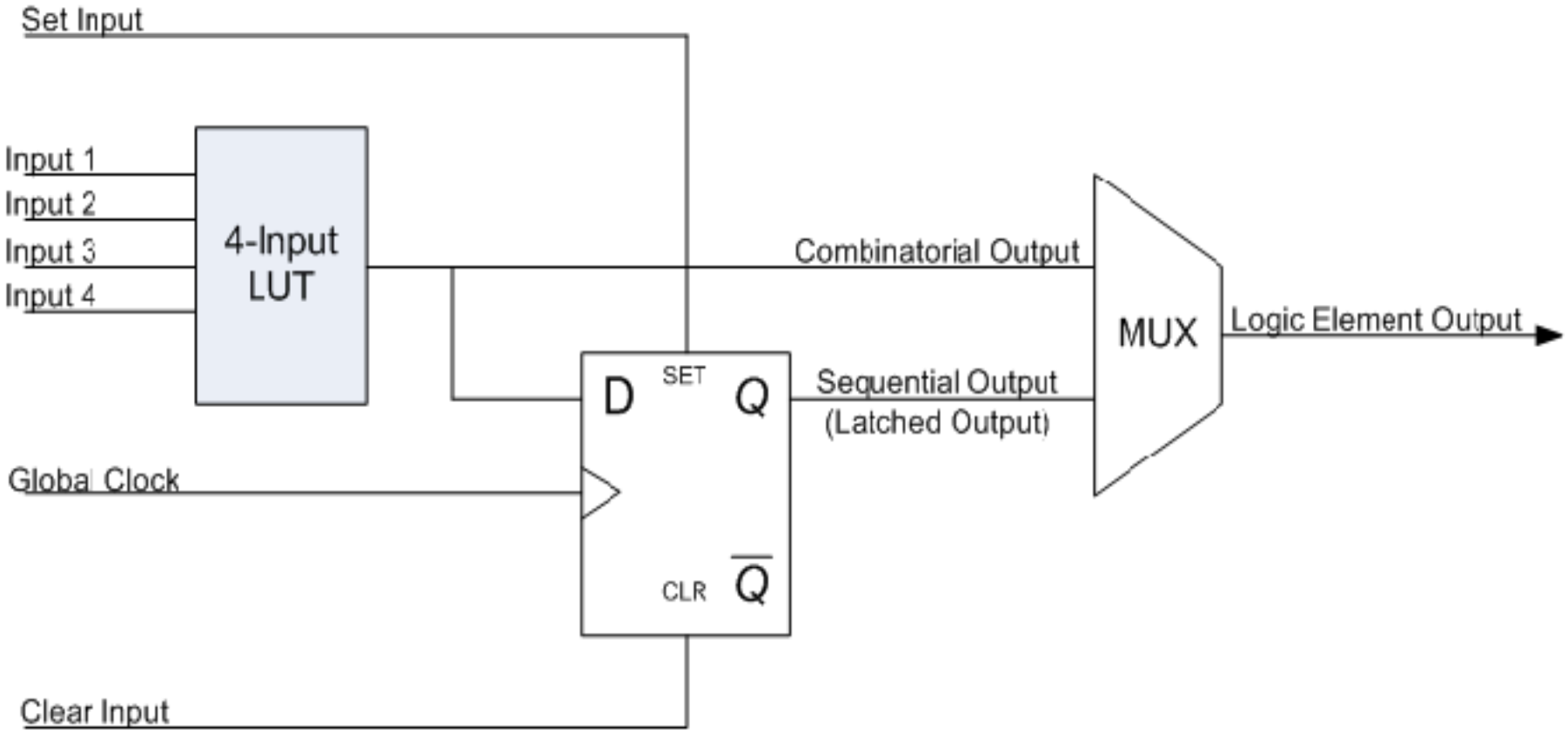
FBE - FPGA Architecture Review[12]

- FPGA is a programmable logic chip that consists of an array of multiple logic elements (a.k.a. logic cells) usually in form of RAM-controlled Look-up Tables (LUT) interconnected by metal lines or RAM-controlled interconnecting switches.



FBE - FPGA Architecture Review[12]

- Each *n-input* LUT is capable of implementing 2^{2^n} different combinatorial logic functions. Adding optional latch or Flip-Flop (FF) enables each cell to produce sequential/combinatorial output based on the user's choice.



FBE – Routing Architectures[11]

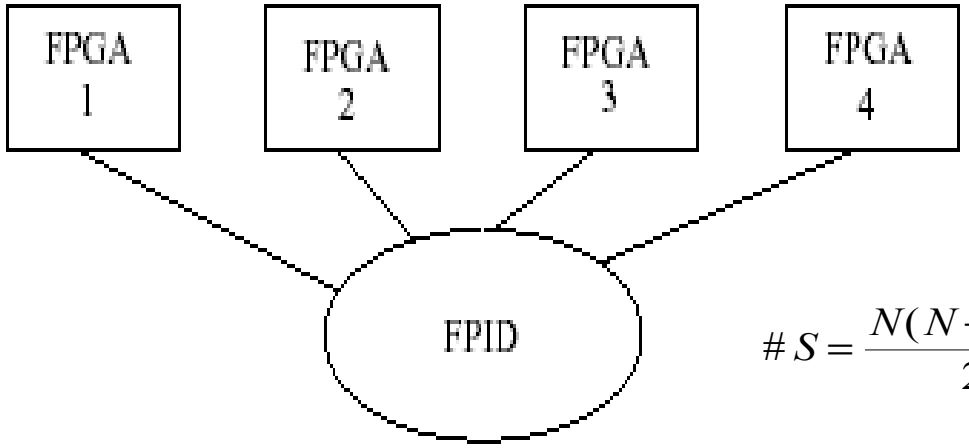
- The way FPGAs are connected to each other is called Routing Architecture.
- Since the design is directly mapped into FPGAs, the routing architecture has big effect on speed, cost and "routability" of emulated design.

inefficient routing architecture → excessive logic → { lower speed / higher cost

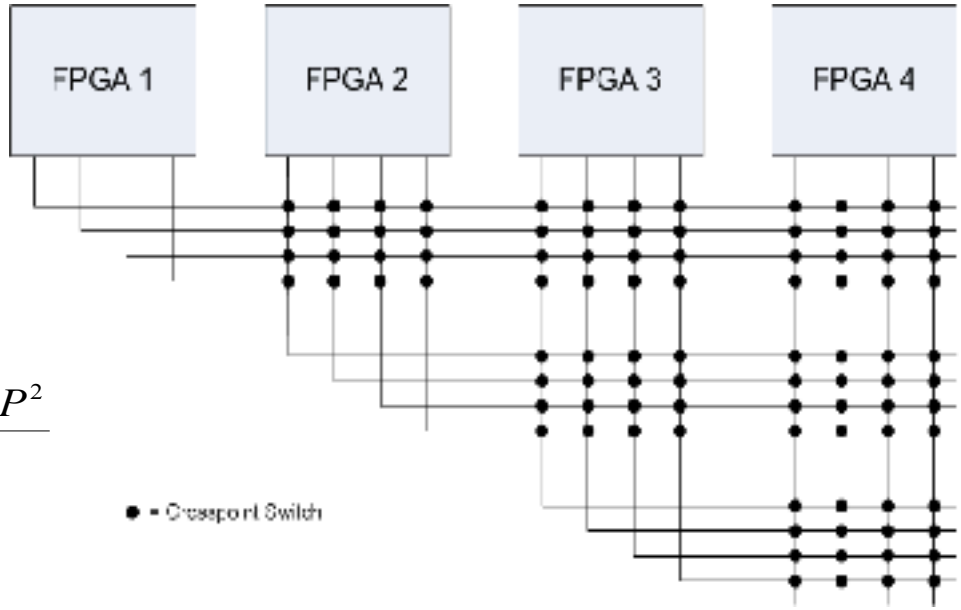
- Recent FBEs, use a number of Field Programmable Interconnect Device (FPID) to connect FPGA I/O pins to each other.

FBE – Routing Architectures

1- Full Crossbar

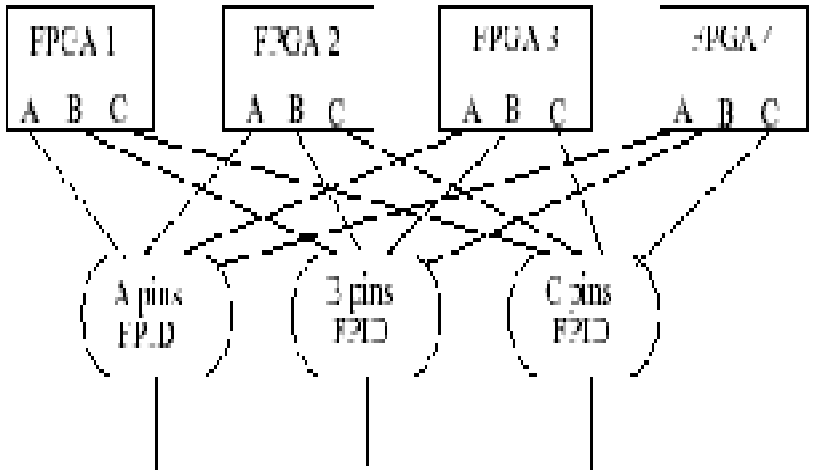


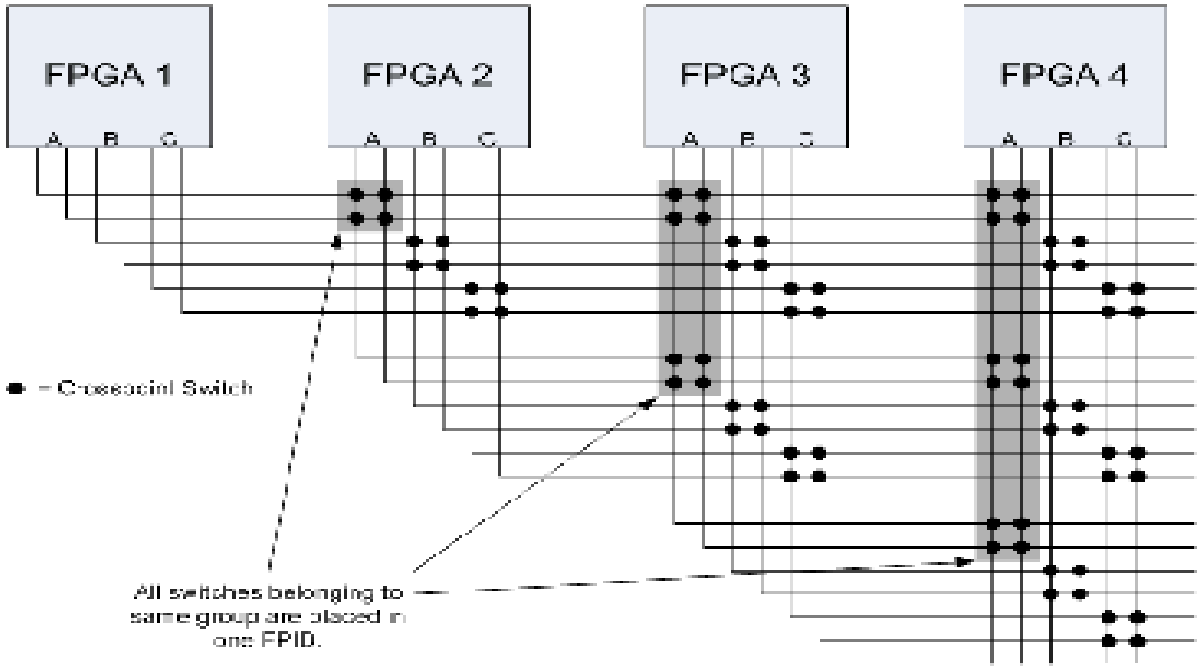
$$\#S = \frac{N(N-1)P^2}{2}$$



- Too many switches

2- Partial Crossbar

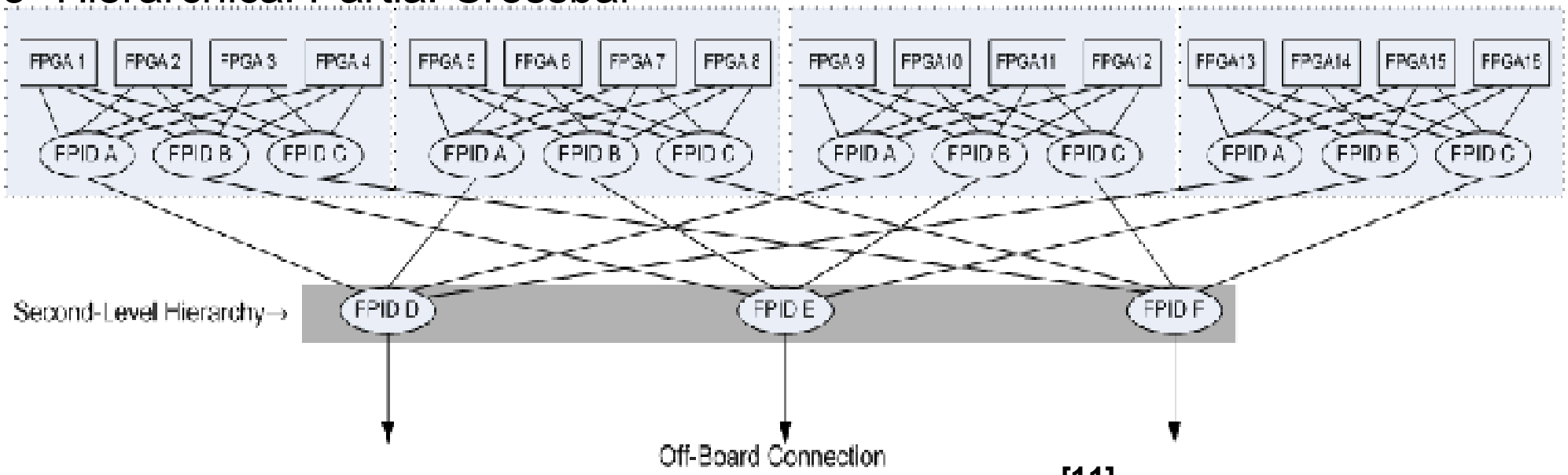




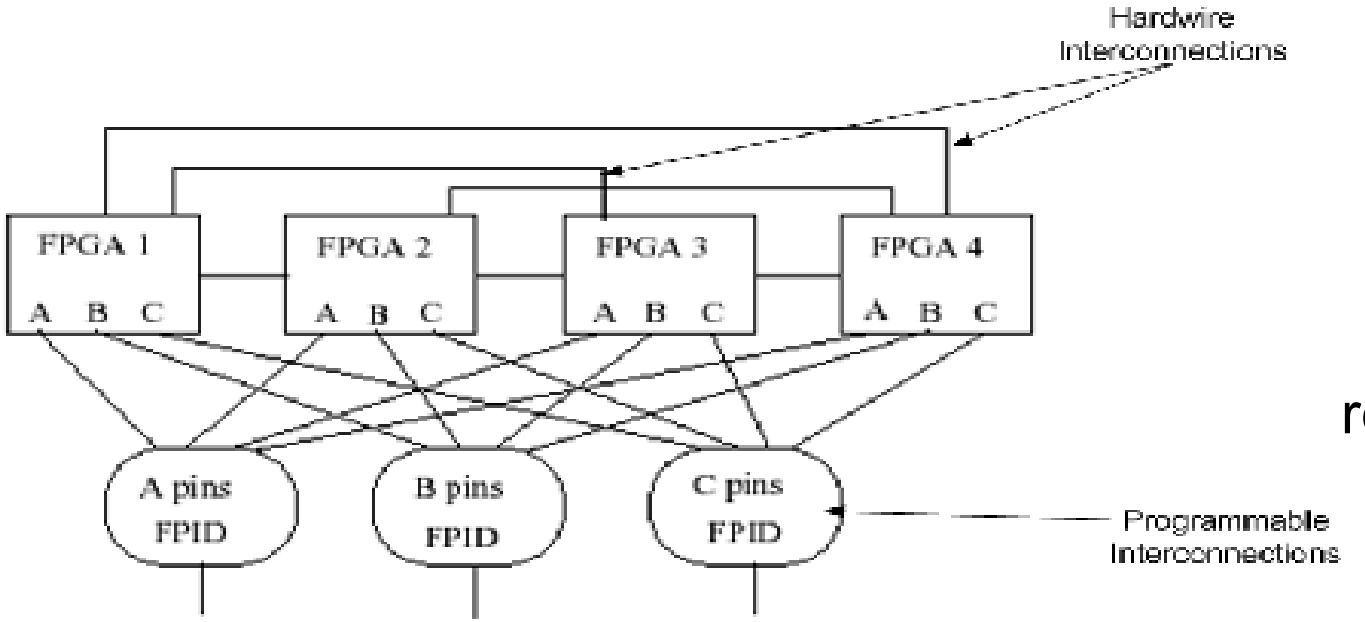
- More practical

FBE – Routing Architectures (cnt'd)

3- Hierarchical Partial Crossbar



4- Hybrid Complete Graph Partial Crossbar (HCGP) [11]



- Most recent

$$0.6 < P_p < 0.8$$

results in 100% routability

FBE – EDA/CAD Tools [8][10]

- Emulation CAD Tool = Design Compiler + Runtime support

- Design compiler : very complex software that efficiently maps huge structural representation (netlist) of a digital design into target emulator architecture.

- Runtime support : a collection of different front-end tools such as logic analyzer,…

- Emulation CAD tools are significantly different from one manufacturer to another or from one product to another.

FBE – EDA/CAD Tools (cnt'd)

1- Synthesis + Flattening to primitives

2- Technology mapping: Map primitives to LUTs
(objective: reduce size, delay, # of LUTs)

3- DRC: check for errors

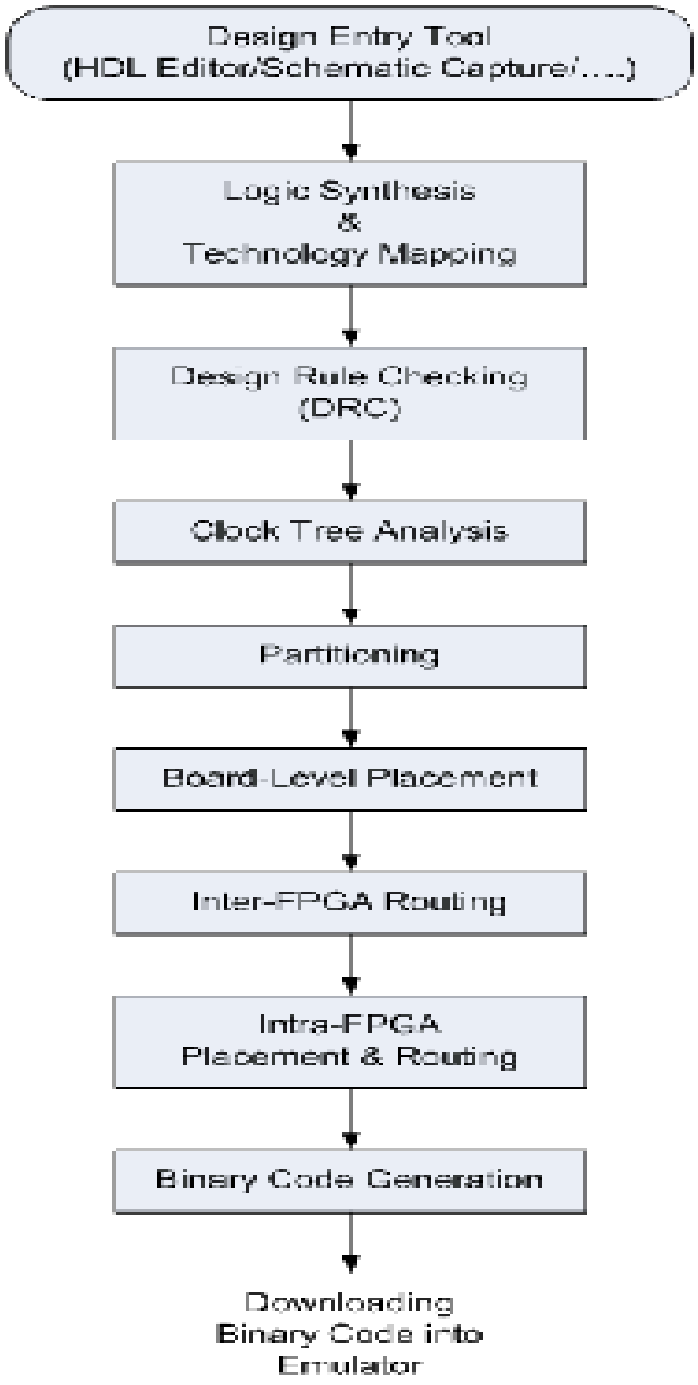
4- CTA: extract timing criteria

5- Partitioning: Divide netlist among FPGAs
based of FPGA's size, pin, speed & timing
(objective: reduce # of FPGAs, delay, size)

6- Board-level placement: (objective: reduce delay)

7- Inter-FPGA routing

8- Intra-FPGA placement & routing



FBE – Pros and Cons[13]

Quickturn™ MercuryPlus

- Pros:

  1- High programmability ≡ High Flexibility

  2- Very fast emulation speed (e.g. 1.5MHz)

  3- Multi-million gates logic capacity (>5 Million gates)

  4- Can emulate asynchronous/multi-clock designs

  5- Can use off-the-shelf FPGAs/FPIDs

- Cons:

  1- CAD Crisis: technology mapping, partitioning, placement and routing algorithm are all NP-hard/NP-complete problems that require heuristic algorithms which results in very long/ unpredictable compilation time.

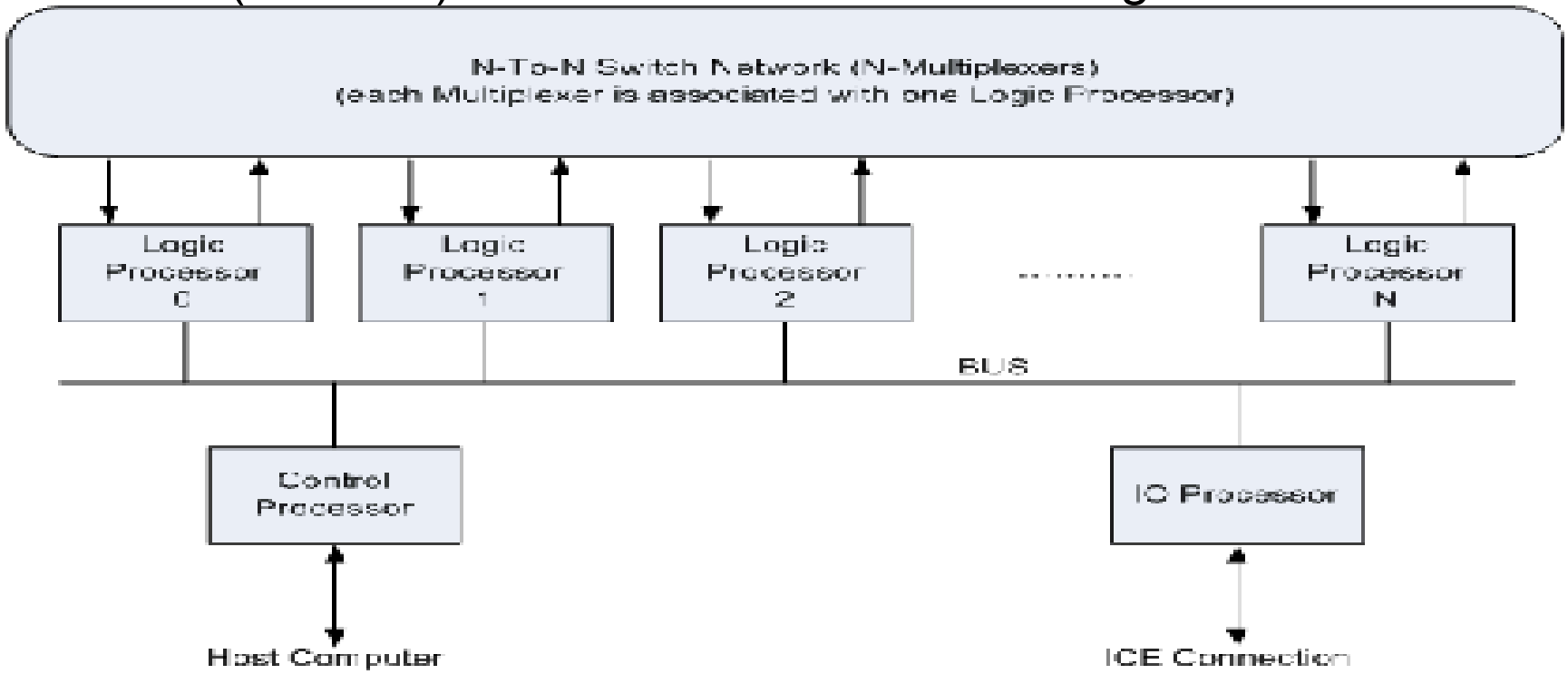
  2- Significant logic capacity inside FPGA is wasted (Rent's rule).

  3- Debug visibility within FPGA is poor.

  4- They are not easily scalable in terms of capacity and performance.

Processor-based Logic Emulation - Architecture

- A number of highly parallel hardware processors (x10 ~ x100) are used to simulate (emulate) functional behavior of a design.

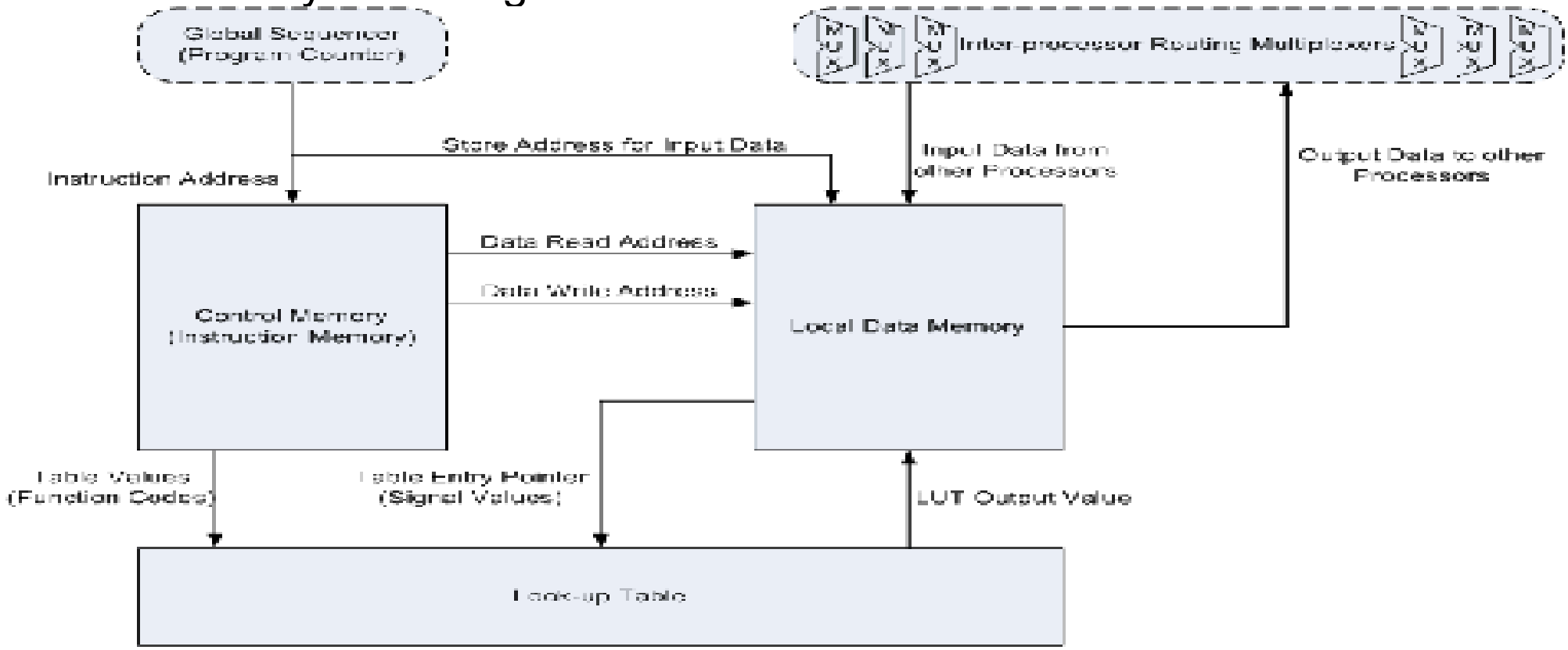


[9][14]

- Each processor simulates the functionality of an *n-input* logic gate at each instance of simulation cycle.
- A single processor is reused to simulate multiple logic gates at different time slices.
- Recent architectures use LUT-based approach inside each processor.
- In most conventional architectures *n = 2, 3 or 4.*

PBE – Architecture (cnt'd)

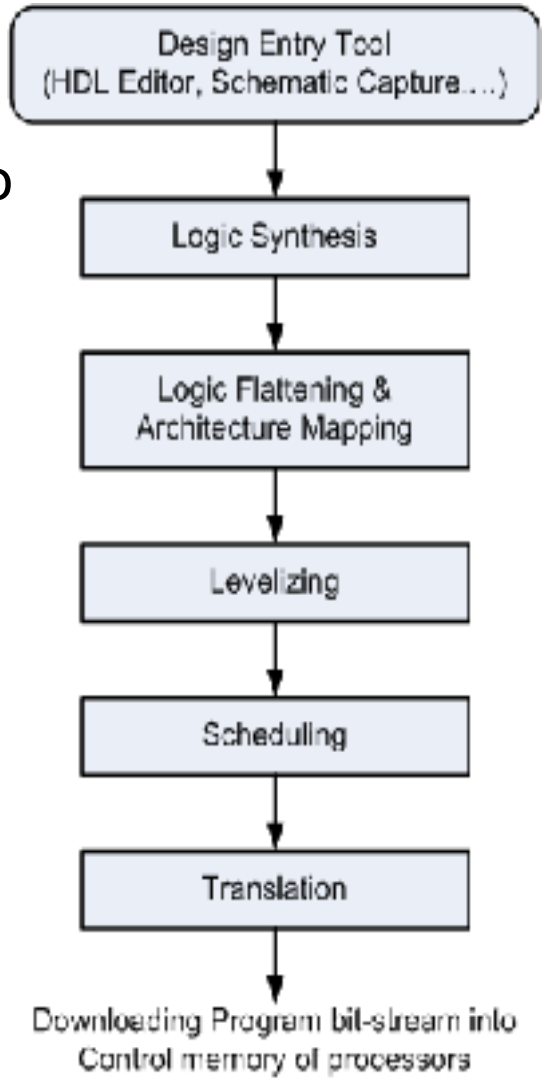
- A sequencer synchronously cycles each processor through many step cycles.
- Each processor has associated Data/Control memory that store emulation data/program.
- At each emulation clock cycle an instruction is fetched from control memory that configures the LUT.
- Data memory stores signal values.



PBE – EDA/CAD Tools[9]

- Emulation compiler translates a logic design into a sequence of control instruction words that can be loaded into processors' control memory.
- Unfortunately, the compilation process may vary from one vendor to another.

  - Architecture mapping: maps flattened netlist into the target processor architecture.
  - Levelizing: assigning logic blocks to processors.
  - Scheduling: Balancing and Maximizing processors' workload.
  - Translation: transforming each processor's control program into sequence of op-codes.

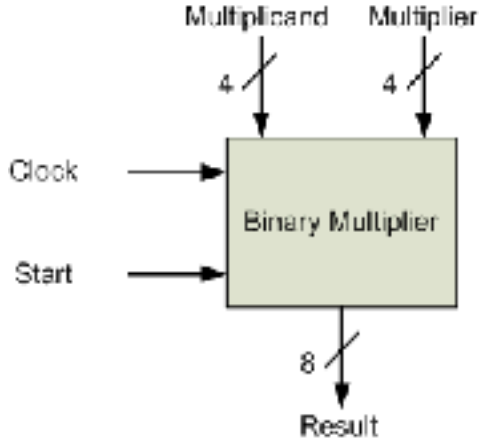


PBE – CAD Flow Example

- Design Example:

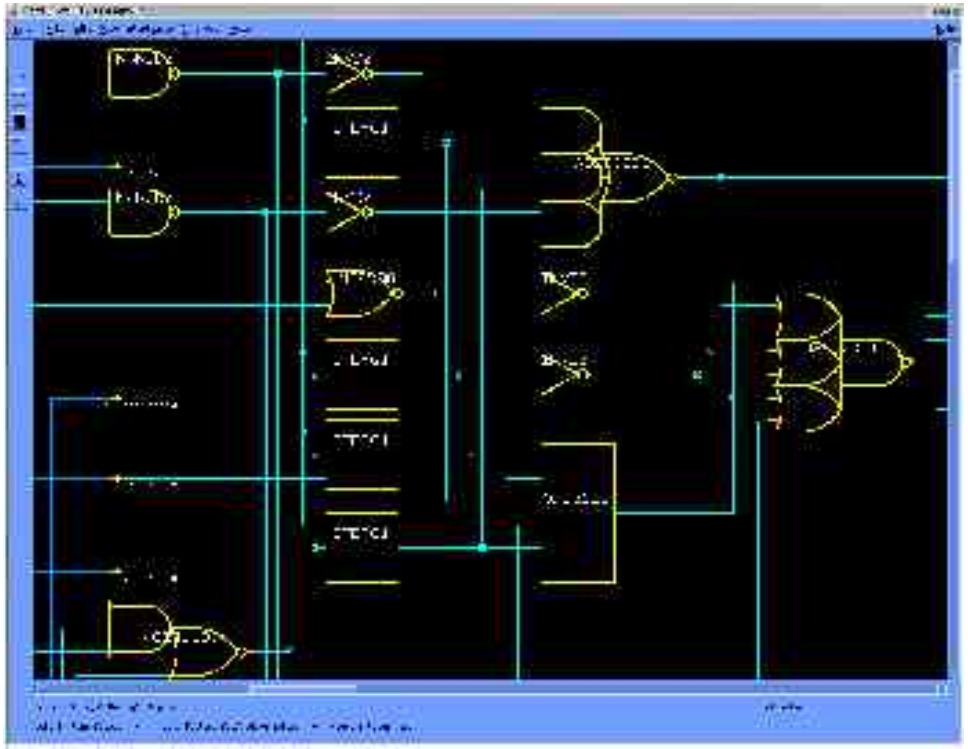
  Using VHDL, designed a 4x4 bit synchronous sequential binary multiplier.

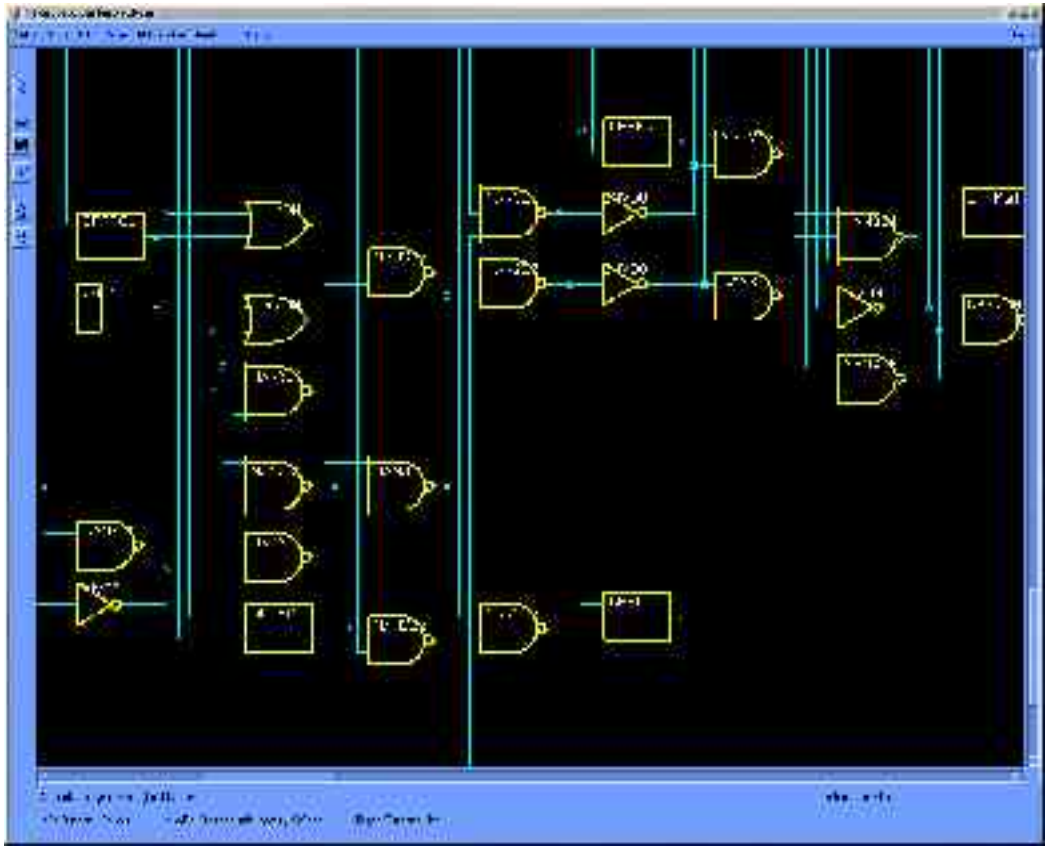
  - After using Synopsys™ *Design Compiler* for synthesis:



PBE – CAD Flow Example (cnt'd)

- Sysnpsys uses different technology libraries for synthesis that do not contain standard logic elements. Therefore, we have to flatten the netlist.



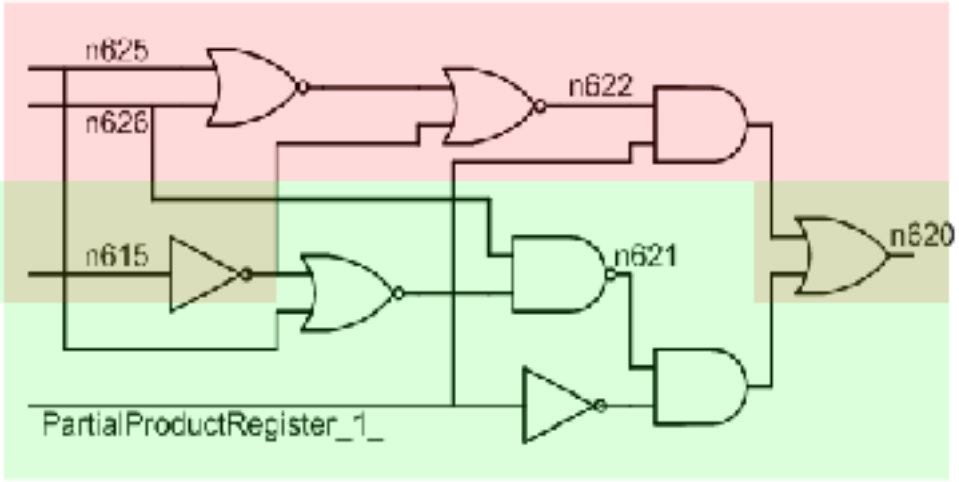


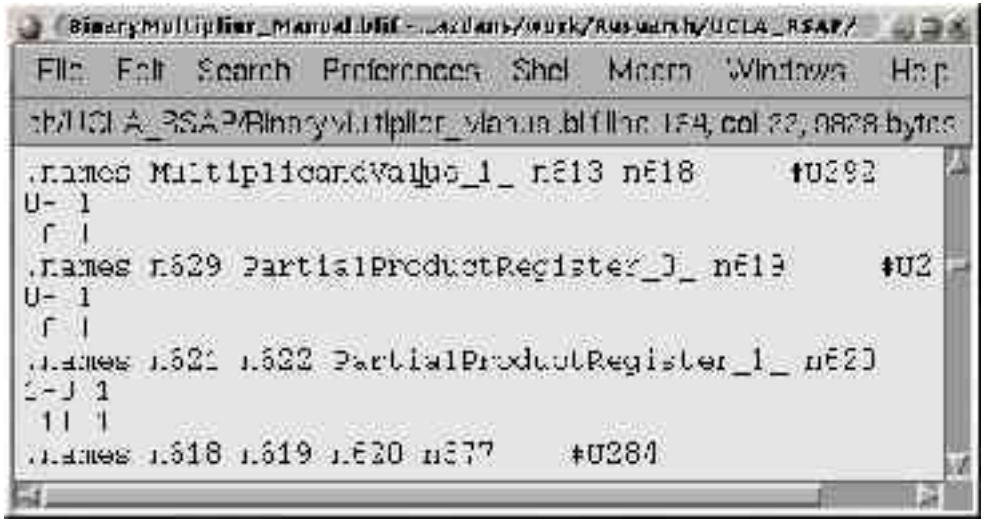
PBE – CAD Flow Example (cnt'd)

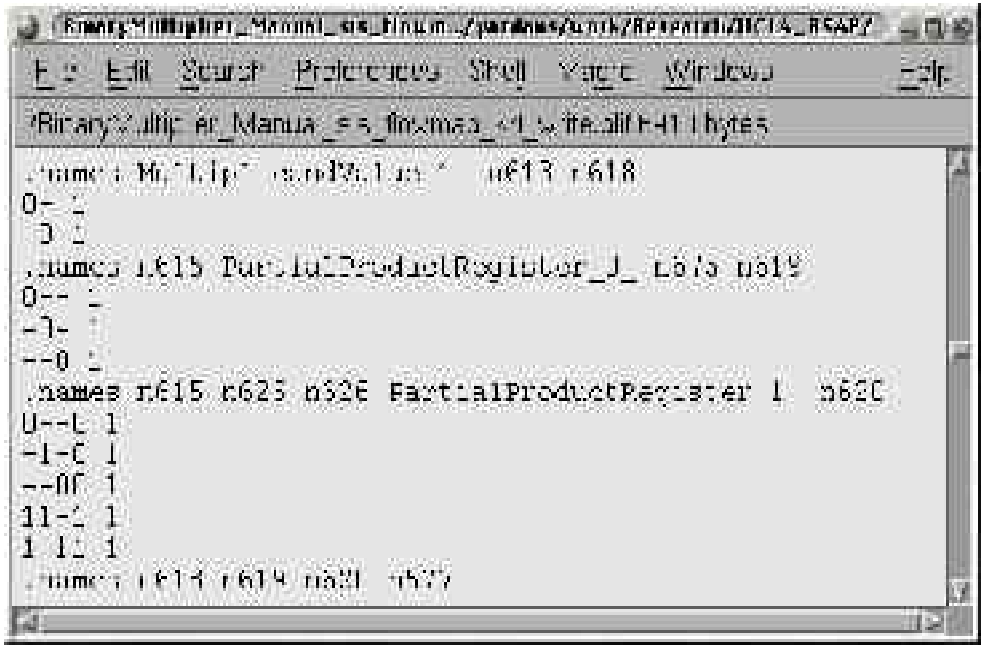
- Architecture Mapping

  1- converted the flattened netlist to ".blif" format.

  2- We *assumed* that each processor contains a reprogrammable 4-input LUT. Mapped the flattened netlist into a 4-bounded netlist.

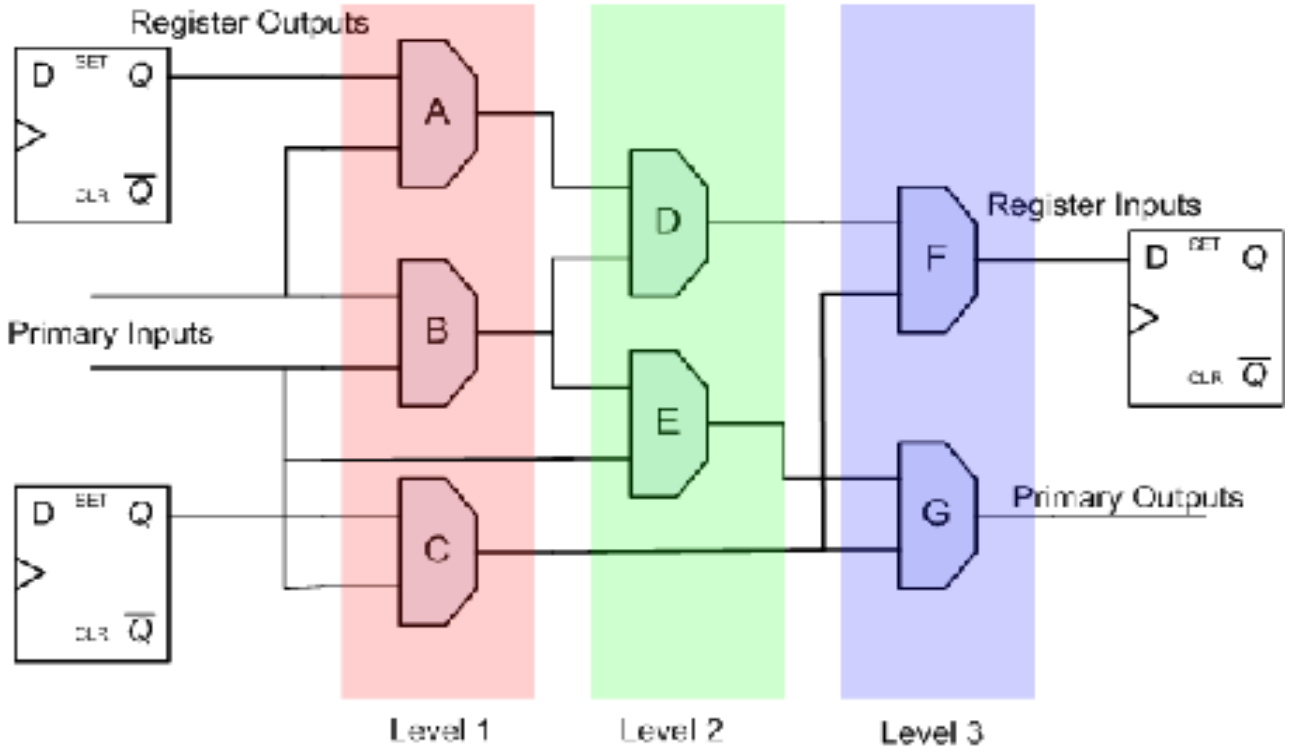






PBE – CAD Flow Example (cnt'd)

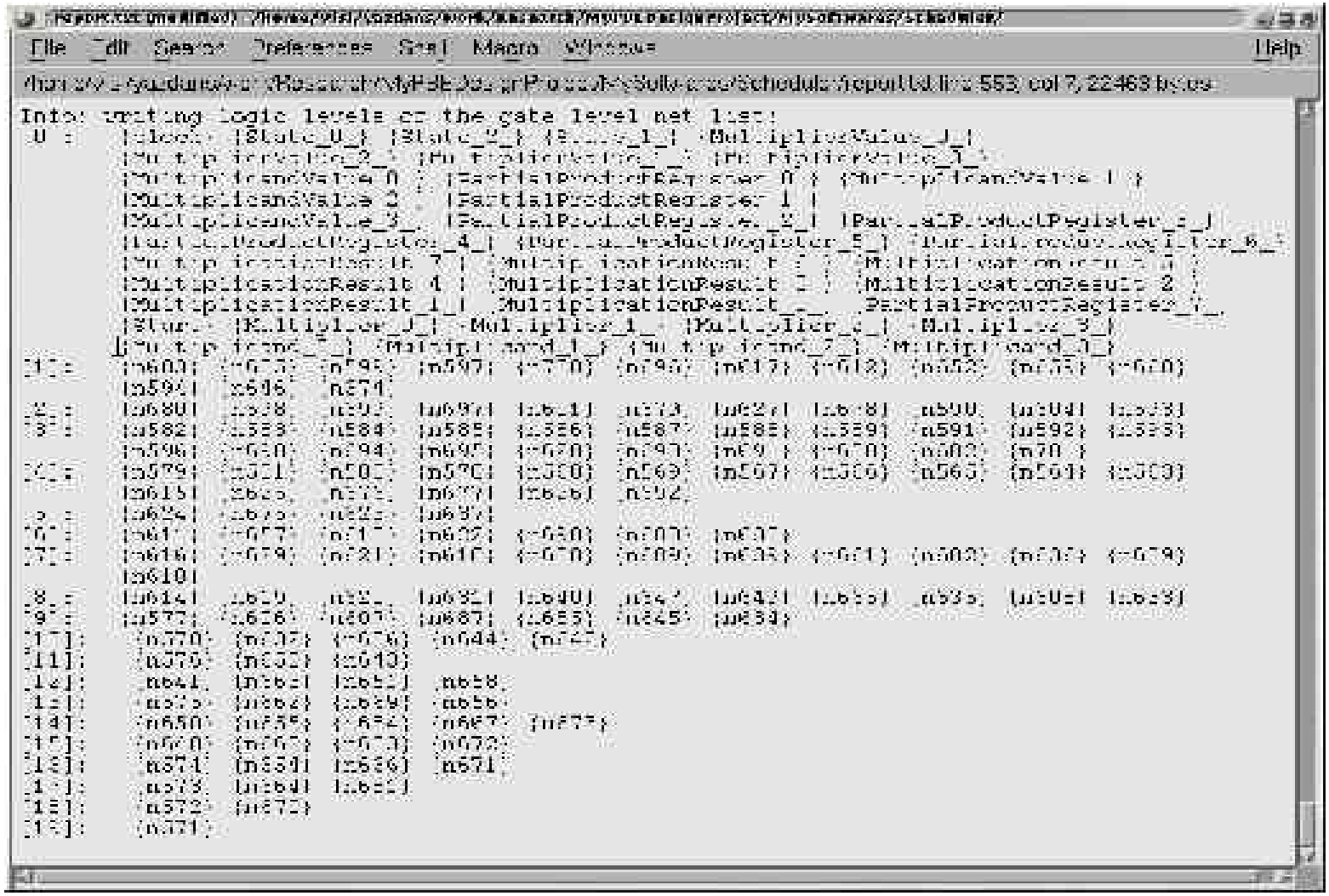
- Software levelizes the netlist (i.e. assigns logic units to processors based on their precedence in mapped netlist)



- Blocks "A" and "B" must be evaluated first.
- Blocks "F" and "G" must be evaluated last.
- Block "C" can be either evaluated at level "1" or "2".
- What is the minimum number of levels? How many processors are needed at each level?

PBE – CAD Flow Example (cnt'd)

- 19 level are required in this example.



- If 21 4-input processors are connected in a Complete Graph topology then the emulation of one design cycle takes only 19 emulation cycles!

PBE – Pros and Cons[13]

Cadence™ Palladium

- Pros:

  1- Higher logic capacity (>120 Million gates)

  2- Fast and predictable compile time (eg. 24MGates @ 3Hours)

  3- High debug visibility (static/dynamic debug during runtime)

- Cons:

  1- Lower emulation speed (x100KHz ~ < 1MHz)

  2- Require to modify the netlist if combinatorial feedback loops/multiple asynchronous clock signals exist in the design.

A Comparison Between PBE and FBE

FBEs are better because:

- FBEs have faster emulation speed (≈x10)
- FBEs are more programmable than PBEs
- FBEs are more contemporary (= lots of ongoing researches)

PBEs are better because:

- PBEs have greater logic capacity (≈x10)
- FBE's compiler takes days to compile a design for which a PBE's compiler takes only few hours. Therefore it is much easier to make "what if" modifications while using PBE.
- PBE have larger visibility and debugging facilities.

Conclusion and Future Work

- Recently, PBEs have come in to re-consideration and lots of academic researches are pointing that way.
- The main questions are:

1- Given a PBE architecture, which Levelization and Scheduling algorithms will result in fastest emulation?

2- Are those the only solutions?

3- What is the effect of PBE architecture on PBE CAD tool?

- Effect of processor granularity on CAD tool
- Effect of routing and data path architecture on CAD tool

References

[1] Intel Corp., www.intel.com, 2005.

[2] K. Wakabayashi, T. Okamoto, “C-based SoC design flow and EDA tools: an ASIC and system vendor perspective”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Dec. 2000.

[3] Synopsys Inc., www.synopsys.com.

[4] D. MacMillen, R. Camposano, D. Hill, T.W. Williams, “An industrial view of electronic design automation”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Dec. 2000.

[5] Z. Barzilai, J.L. Carter, B.K. Rosen, J.D.Rutledge, “HSS--A High-Speed Simulator”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Jul. 1987.

[6] M. Abramovici,Y.H. Levendel, P.R. Menon, “A Logic Simulation Machine”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, April 1983.

[7] IKOS Systems Inc., www.mentor.com

[8] A. Yazdanshenas, M.A.S. Khalid, “A Tutorial on Logic Emulation Systems”, paper currently under revision, 2005.

[9] W.F. Beausoliel et al. “Multiprocessor for Hardware Emulation.” U.S. Patent 5551013, Aug. 1996.

[10] J. Varghese, M. Butts, J. Batcheller, “An efficient logic emulation system”, IEEE Trans. on VLSI Systems, June 1993.

References (cnt'd)

[11] M.A.S. Khalid, J. Rose, "A Novel and Efficient Routing Architecture for Multi-FPGA Systems," IEEE Trans. on VLSI Systems, Feb. 2000.

[12] Xilinx Inc., "The Programmable Gate Array Book", www.xilinx.com.

[13] Cadence Design System Inc., Cadence Quickturn Division, www.cadence.com.

[14] M. M. Denneau, "The Yorktown Simulation Engine", ACM Proc. of the 19th Conference of Design Automation, Jan. 1982.

Thank You!