Verification I

Sungho Kang

Yonsei University

Outline

- Introduction
- Definition
- Hardware Acceleration
- Emulation

Complexity Trends

- **Rapidly Growing Design Size**
  - **Doubling of million-gate designs**
  - **50% reduction in designs under 500K gates**
  - **3X reduction in designs under 100K gates**
- **Shrinking Process Geometries**
  - **Nearly 10X reduction in .5 micron designs**
  - **Most designs going to .35 micron or below**
- **Architectural Complexity**
  - **Before*:* simple instruction pipelines, single functional units, simple stalls/holds, simple caches/TLBs, protocols at the pins**
  - **Now*:* deep instruction pipelines, super-scalar design, multiple (pipelined) functional units, instruction re-circulate, speculative execution, complex stalls/holds, complex protocols on chip and at the pins, integration of external IP**

Verification is the Biggest Problem

- Verification groups growing faster than design groups
- Verification tools are largest budget item for many groups
- Expenditures are growing

Informal Verification

- Simulation
  - Compare against an executable version of the specification, also know as THE GOLDEN MODEL
  - Simulate in software
  - Simulate in hardware
- Test cases
  - Hard-written by the designers
  - Randomly generated test vectors

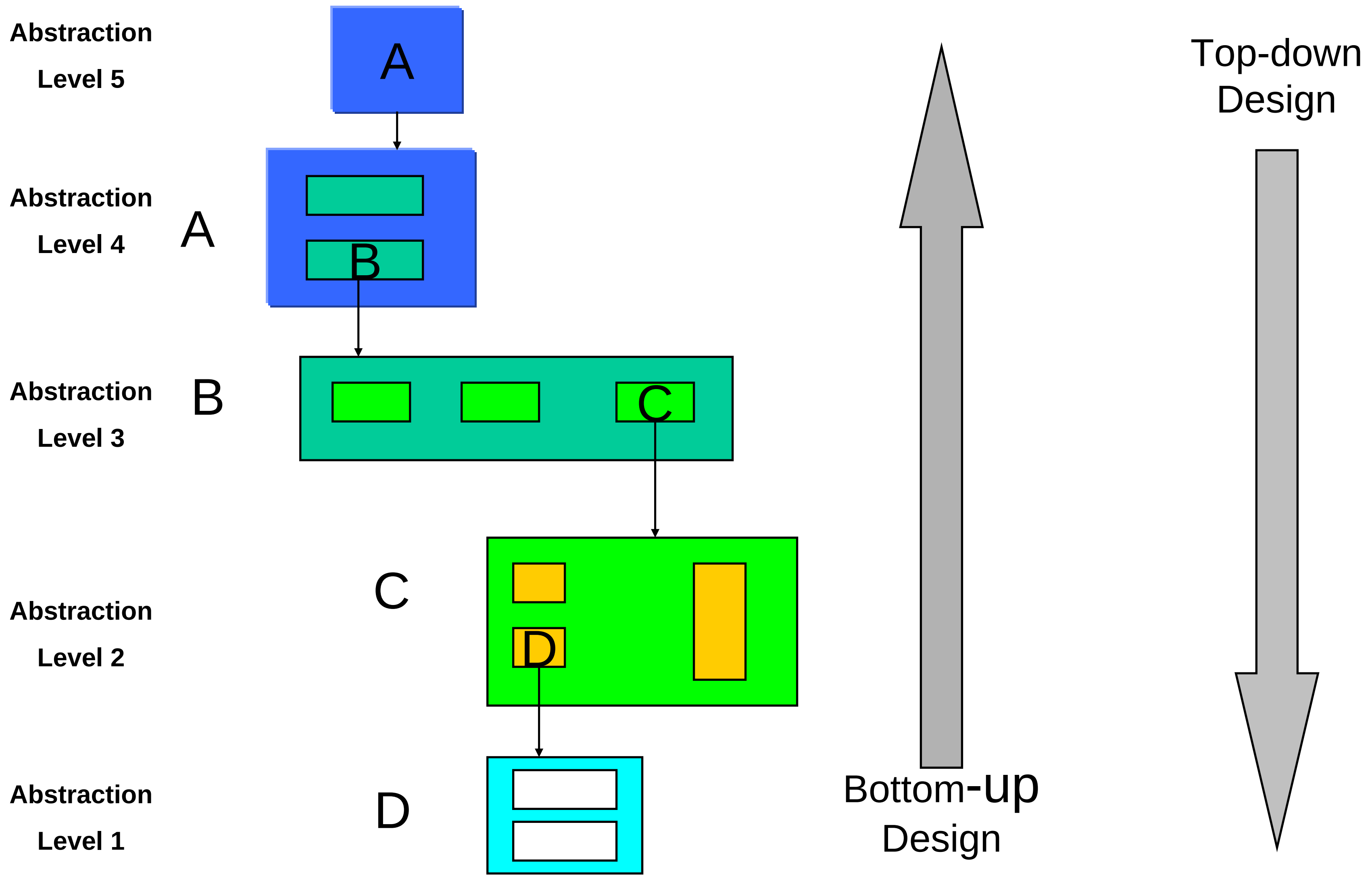
Trends : Verification Problem

- Number of test vectors proportional to design complex
  - Simulation cannot guarantee correctness
  - Confidence based on proportion of design space explored
- Hardware, software systems are becoming increasingly complex
  - Number of basic components growing exponentially
  - Designs are increasingly aggressive

Why Specify

- Additional documentation of system’s interface
- More abstract description of system design
- To perform some formal analysis of the system

Why Specify


Abstraction Level 5
A
Abstraction Level 4
A
B
Abstraction Level 3
B
C
Abstraction Level 2
C
D
Abstraction Level 1
D
Bottom-up Design
Top-down Design

Verification

- Hardware Acceleration ✓
- Emulation ✓
- Co-Verification
- Formal Verification

Hardware Acceleration

Sungho Kang

Yonsei University

Outline

- Introduction
- Boeing
- TEGAS
- Yorktown Simulation Engine
- Logic Simulation Machine
- HAL
- ZYCAD
- AAP-1
- Reconfigurable

Why Simulation Engine

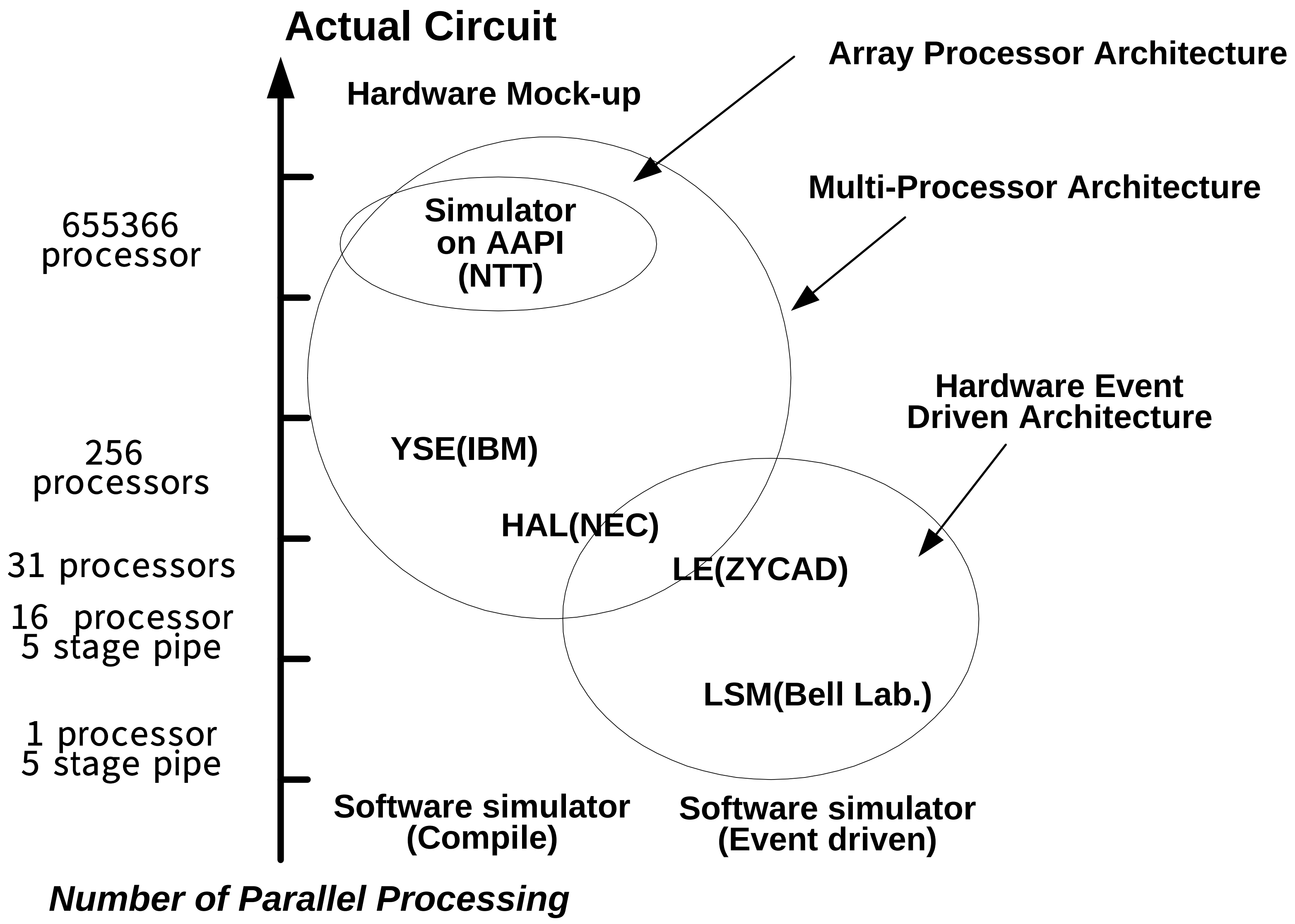
- **Speed up difficulty in software simulation**
- **Parallel Processing**
  - **Multi-processing**
  - **Pipelining**
  - **Array Processing**
- **Hardware Implementation**

- **Simulation Engine / Hardware Accelerator**
  - **Compiled**
  - **Event Driven**
  - **Multi-Processor**
  - **Array**

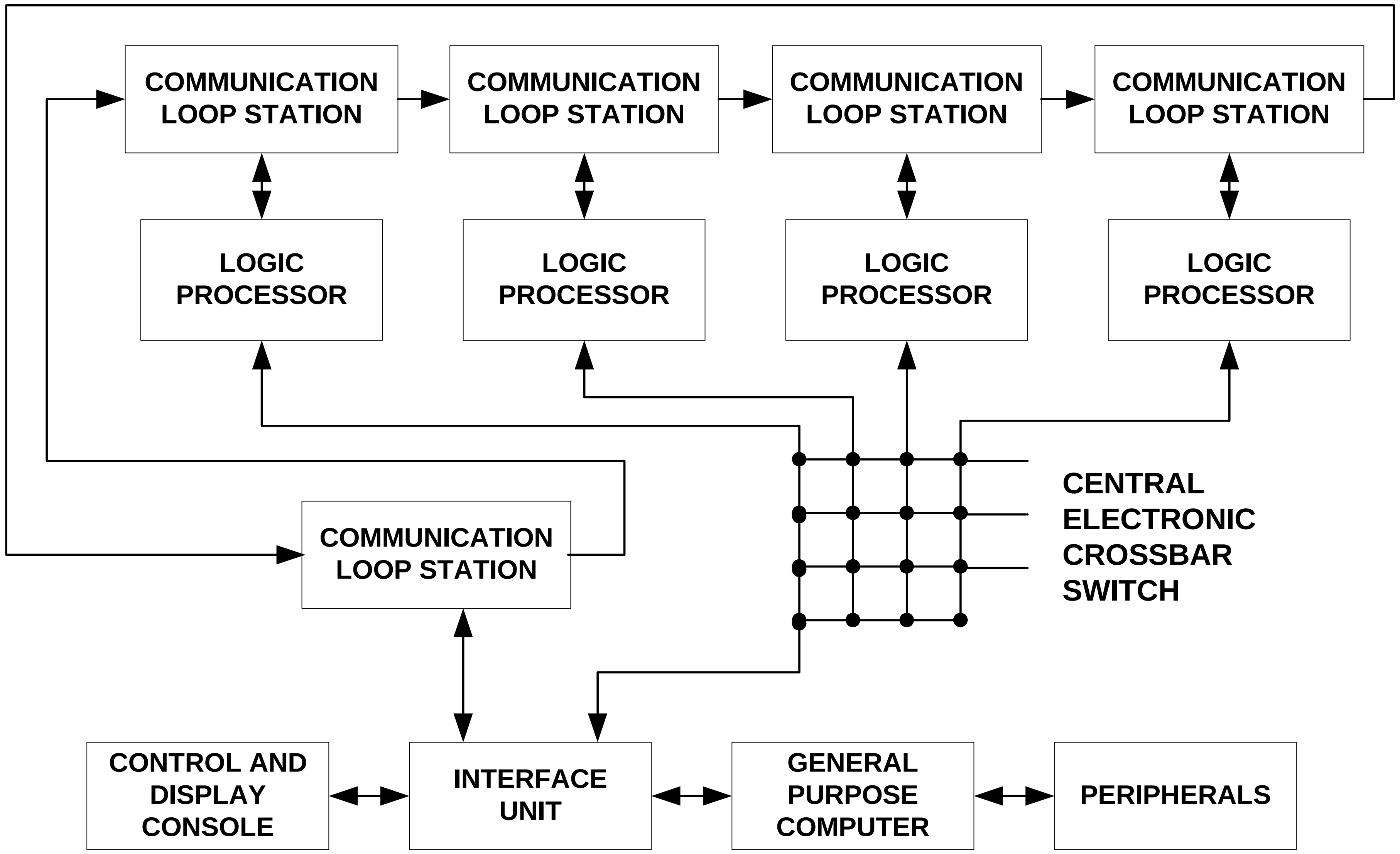
Simulation Engine Performance

	Architecture	Maximum Evalution Units	Simulation Algorithm	Maximum Gates	Announced Speed
YSE (IBM)	Multi-Processor Pipelining	256	Compile	1M (4 input / 1 output gate)	2000M (gates / sec)
HAL (NEC)	Multi-Processor Pipelining	31	Level Controlled Event Driven	1.5M	300M (gates / sec)
LE (ZYCAD)	Multi-Processor Pipelining	16	Event Driven	1.6M (2 input / 1 output gate)	16M (gates / sec)

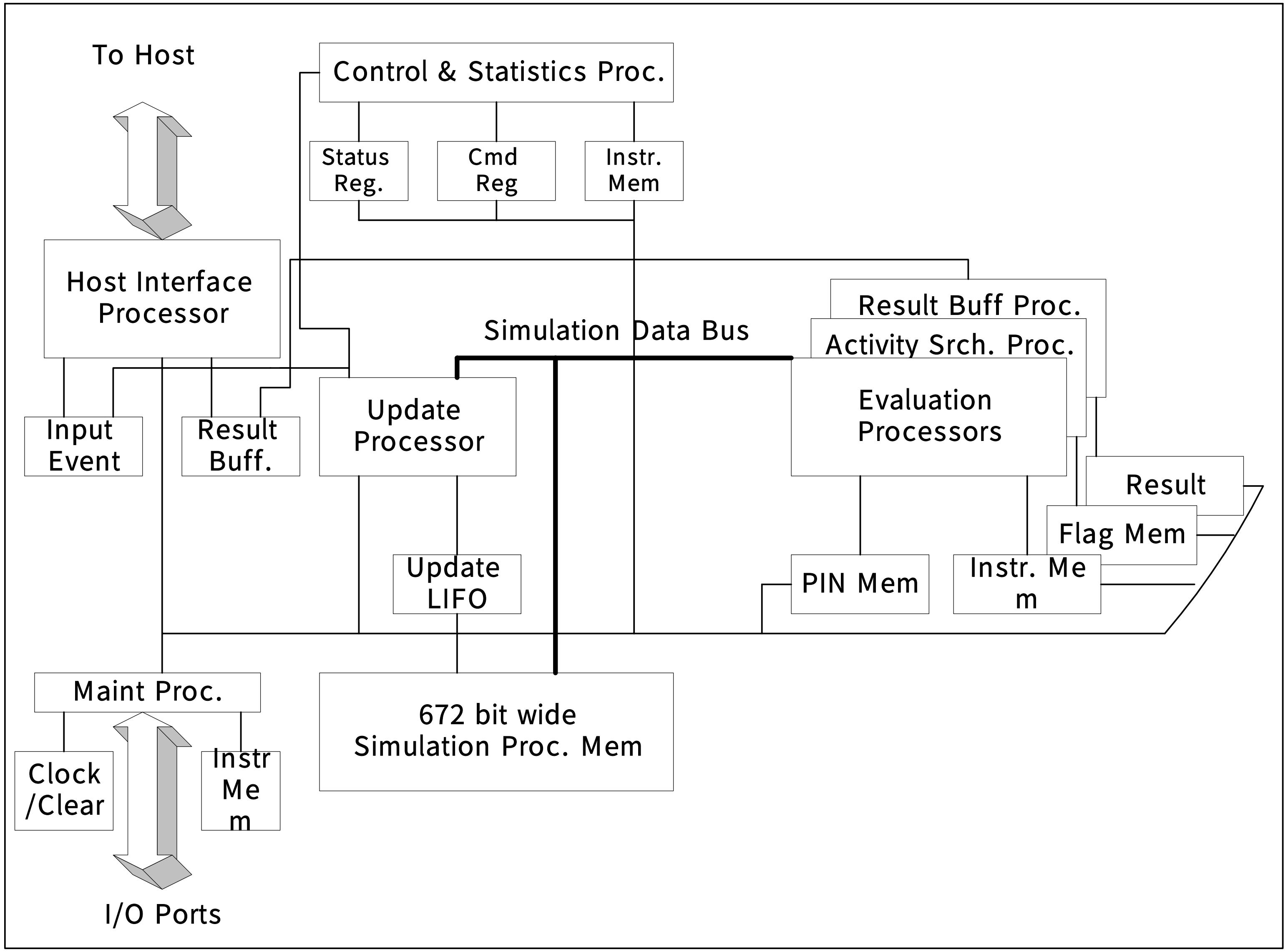
Simulation Engine Classification


Actual Circuit
Hardware Mock-up
Array Processor Architecture
Multi-Processor Architecture
Hardware Event Driven Architecture
655366 processor
256 processors
31 processors
16 processor 5 stage pipe
1 processor 5 stage pipe
Simulator on AAPI (NTT)
YSE(IBM)
HAL(NEC)
LE(ZYCAD)
LSM(Bell Lab.)
Software simulator (Compile)
Software simulator (Event driven)
Number of Parallel Processing

Boeing Computer Simulator

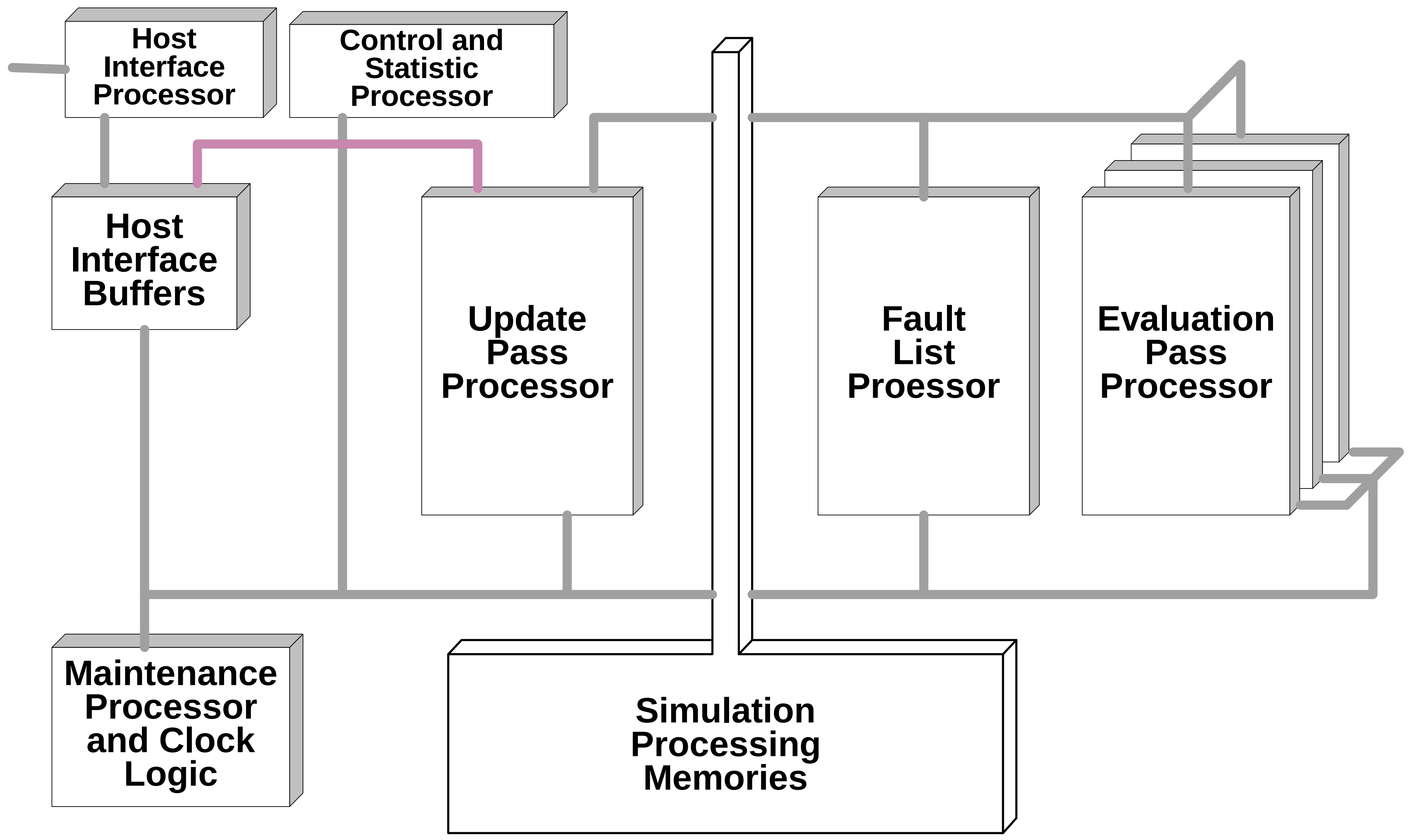

COMMUNICATION LOOP STATION
COMMUNICATION LOOP STATION
COMMUNICATION LOOP STATION
COMMUNICATION LOOP STATION
LOGIC PROCESSOR
LOGIC PROCESSOR
LOGIC PROCESSOR
LOGIC PROCESSOR
CENTRAL ELECTRONIC CROSSBAR SWITCH
COMMUNICATION LOOP STATION
CONTROL AND DISPLAY CONSOLE
INTERFACE UNIT
GENERAL PURPOSE COMPUTER
PERIPHERALS

TEGAS Accelerator



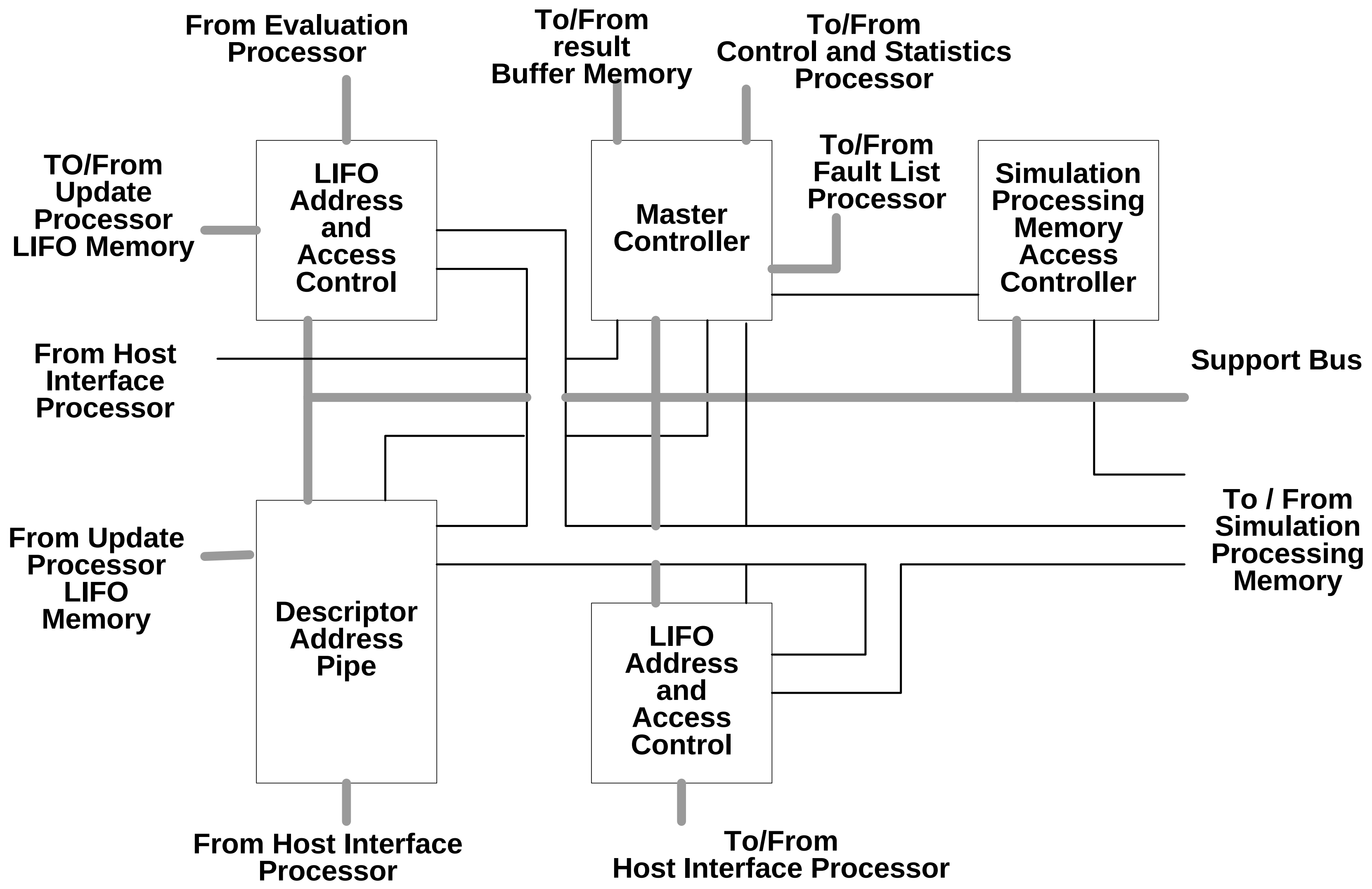
TEGAS Accelerator

- **Functional Level Block Diagram**


Host Interface Processor
Control and Statistic Processor
Host Interface Buffers
Update Pass Processor
Fault List Proessor
Evaluation Pass Processor
Maintenance Processor and Clock Logic
Simulation Processing Memories

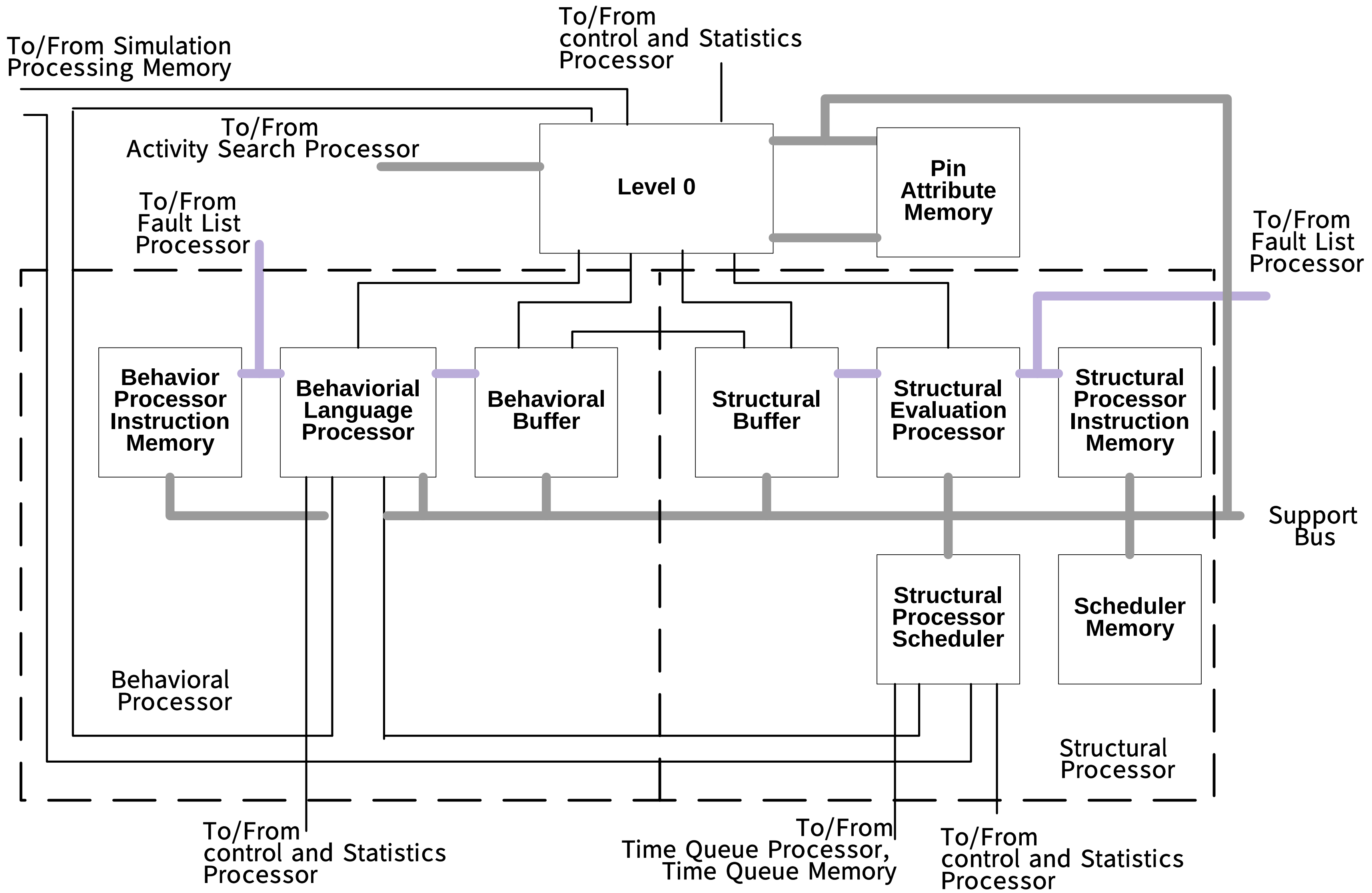
TEGAS Accelerator

- Accelerator Update Processor


From Evaluation Processor
To/From result Buffer Memory
To/From Control and Statistics Processor
TO/From Update Processor LIFO Memory
LIFO Address and Access Control
Master Controller
To/From Fault List Processor
Simulation Processing Memory Access Controller
From Host Interface Processor
Support Bus
To / From Simulation Processing Memory
From Update Processor LIFO Memory
Descriptor Address Pipe
LIFO Address and Access Control
From Host Interface Processor
To/From Host Interface Processor

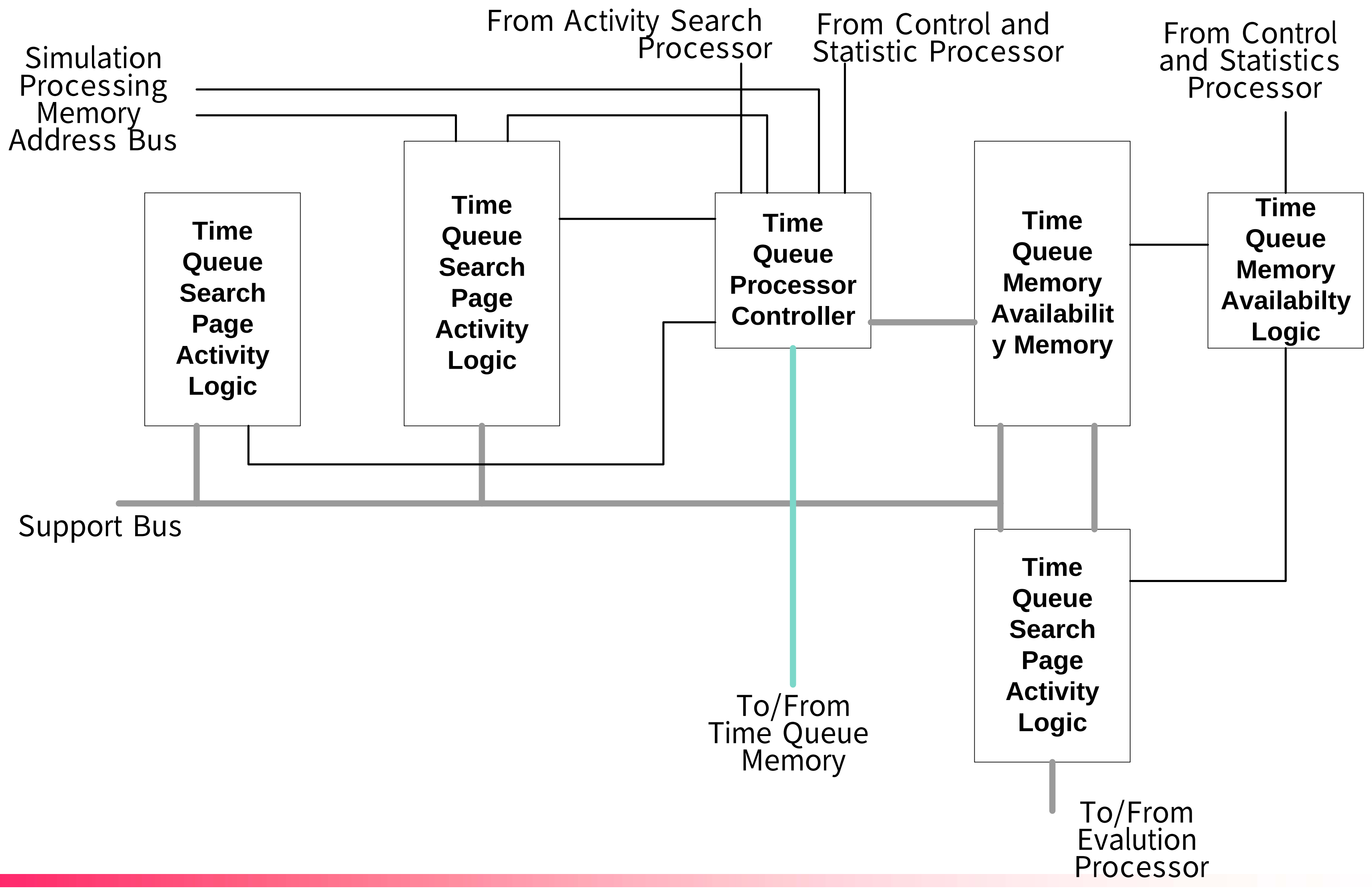
TEGAS Accelerator

- Accelerator Evaluation Processor


To/From control and Statistics Processor
To/From Simulation Processing Memory
To/From Activity Search Processor
Level 0
Pin Attribute Memory
To/From Fault List Processor
To/From Fault List Processor
Behavior Processor Instruction Memory
Behaviorial Language Processor
Behavioral Buffer
Structural Buffer
Structural Evaluation Processor
Structural Processor Instruction Memory
Support Bus
Structural Processor Scheduler
Scheduler Memory
Behavioral Processor
Structural Processor
To/From control and Statistics Processor
To/From Time Queue Processor, Time Queue Memory
To/From control and Statistics Processor

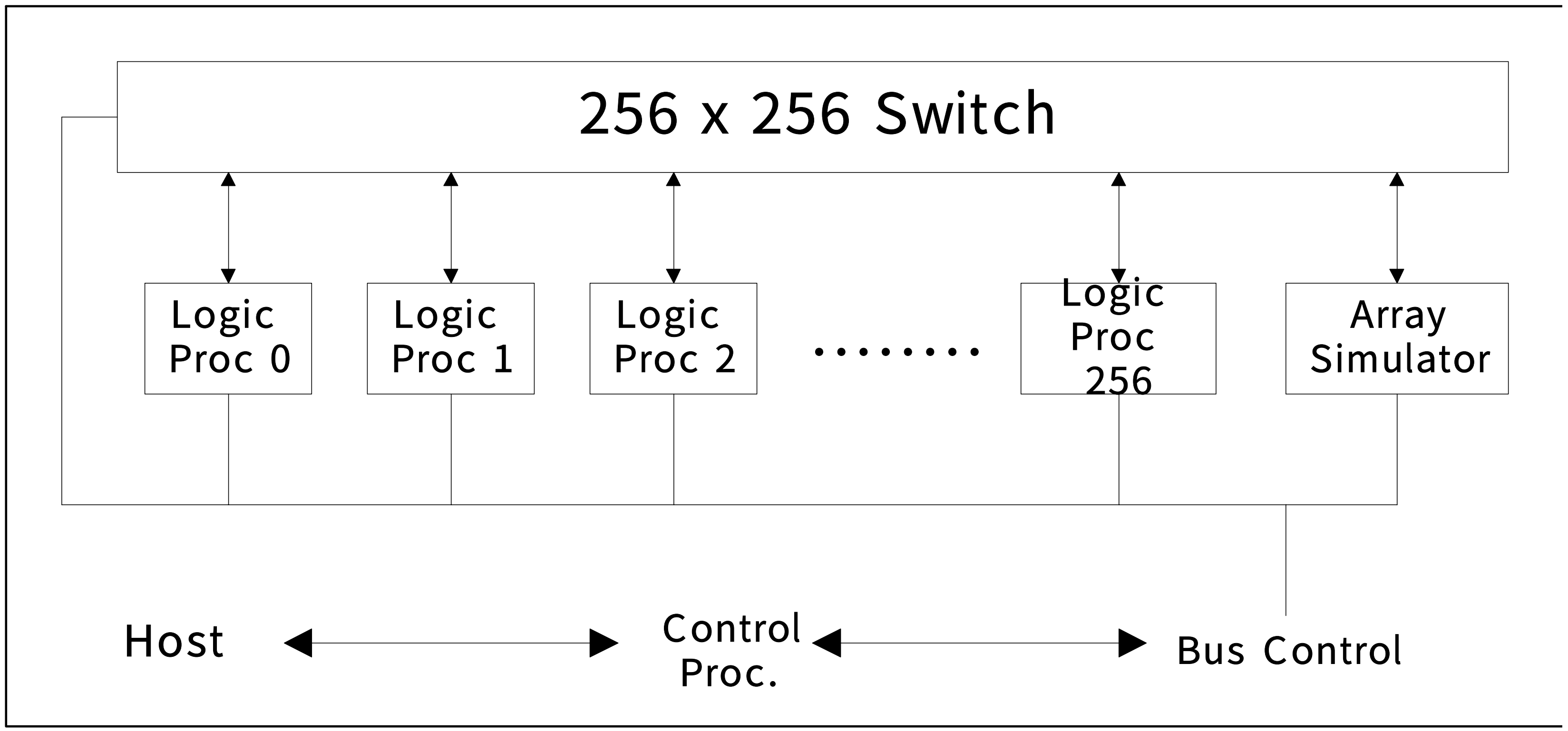
TEGAS Accelerator

- **Accelerator Time Queue Processor**


From Activity Search Processor
From Control and Statistic Processor
From Control and Statistics Processor
Simulation Processing Memory Address Bus
Time Queue Search Page Activity Logic
Time Queue Search Page Activity Logic
Time Queue Processor Controller
Time Queue Memory Availability Memory
Time Queue Memory Availabilty Logic
Support Bus
Time Queue Search Page Activity Logic
To/From Time Queue Memory
To/From Evalution Processor

Yorktown Simulation Engine

- Compiled

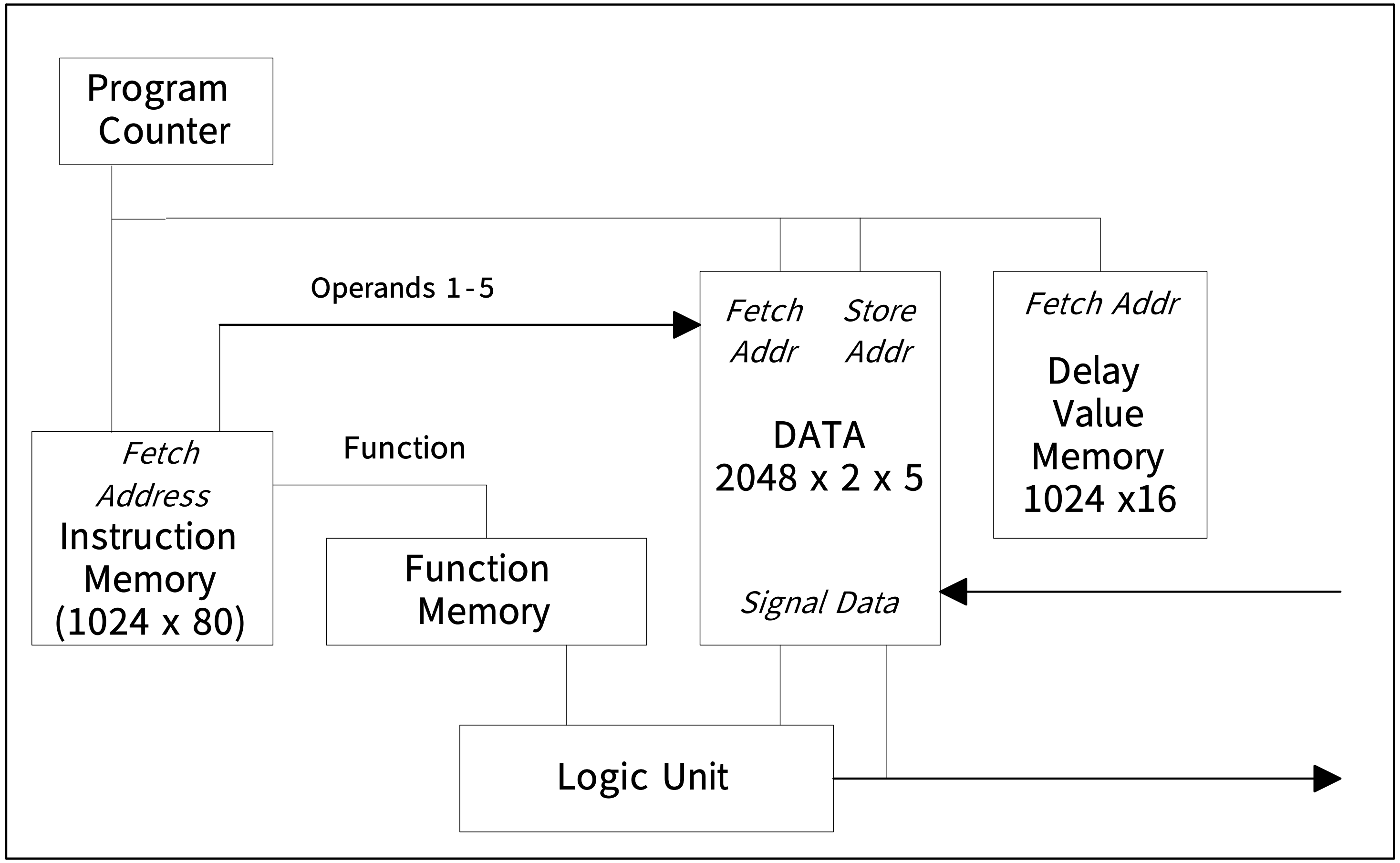


Yorktown Simulation Engine

- Partitioning of 256 PUs
  - Each PU simulates a subcircuit consisting of up to 4k gates
  - Specialized PU for RAMs and ROMs
- All PUs are synchronized by a common clocks
- PU can evaluate a gate during every clock cycle
- Partitioning
  - Minimize the waiting time
- Control processor : host to YSE

Yorktown Simulation Engine

- **Logic Processor**



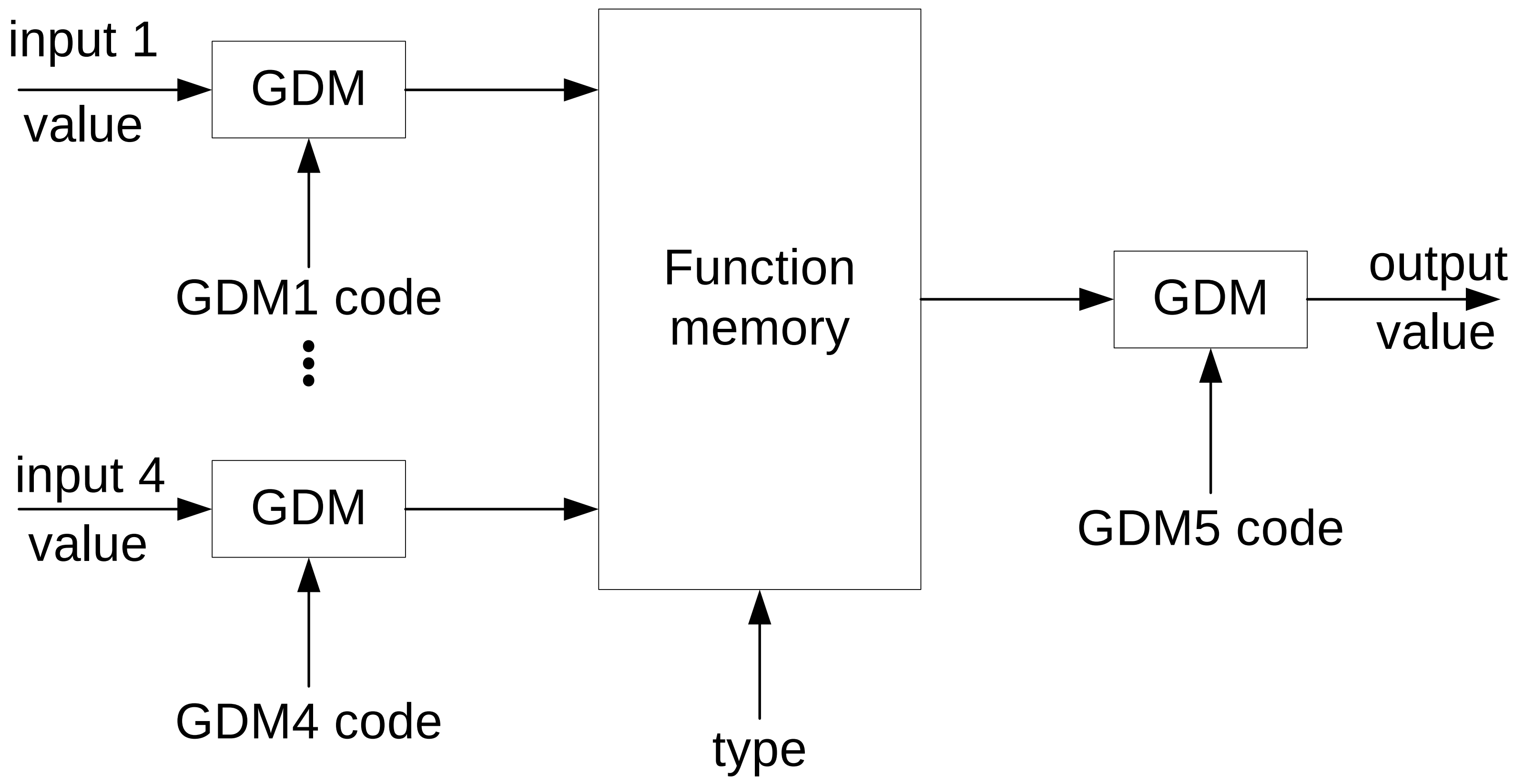
Yorktown Simulation Engine

- Logic Processor
  - PC provide the index to the next gate to be evaluated (compiled)
  - Signal values(0,1,X,Z) are stored in data memory
  - Up to 4 inputs for each gate

  - Generalized DeMorgan Code(GDM)
  - 16 functions of 4 valued variables
  - Evaluation is done in zoom table

Yorktown Simulation Engine

- **Function Unit**
  - **Concatenating gate type with input values by GDM code**

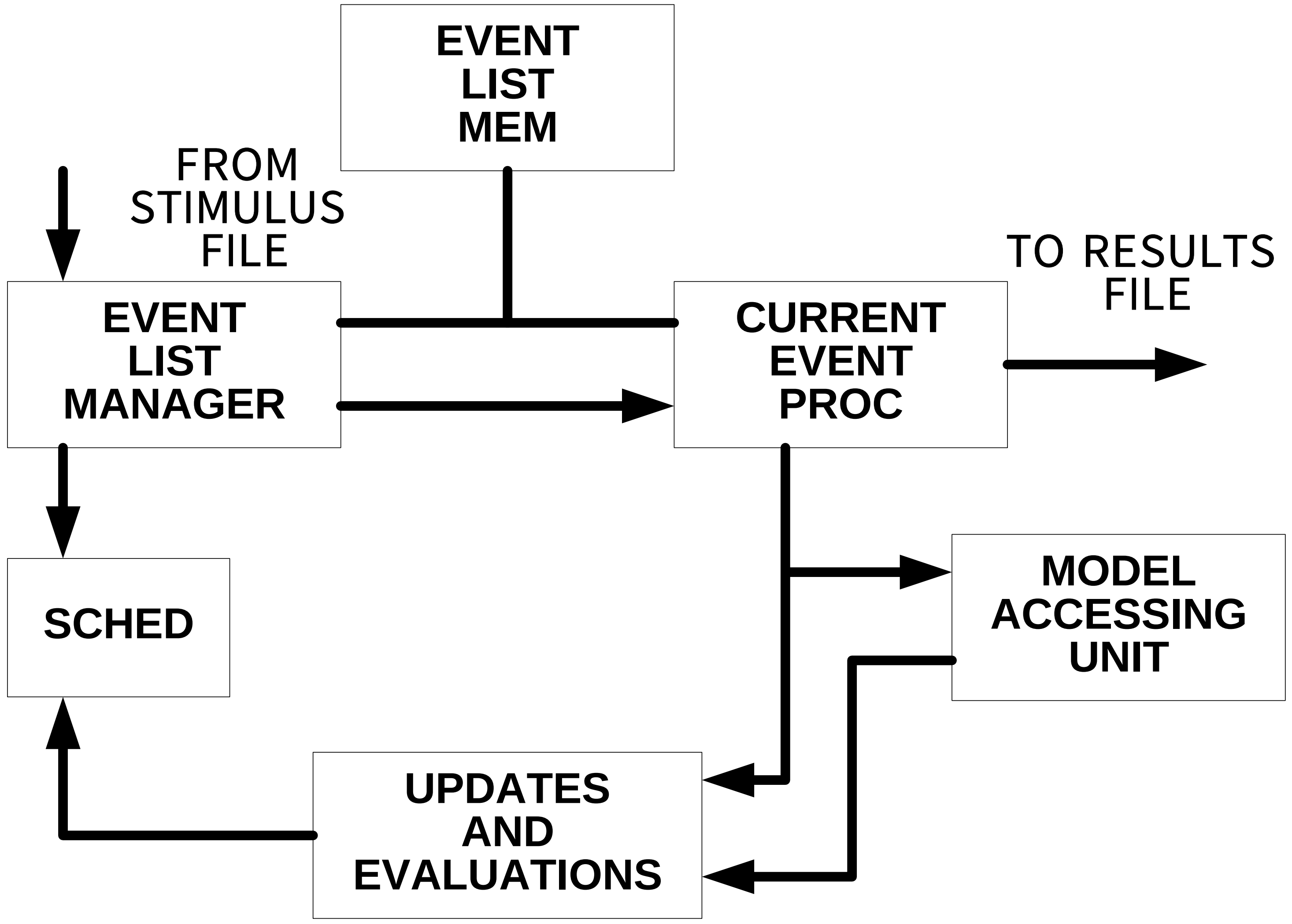


Logic Simulation Machine

- Event Driven Logic Simulation Tasks

Processor	Task
CURRENT EVENT PROC	1) retrieve and distribute event 2) record event 3) oscillation check
MODEL ACCESSING UNIT	1) determine first fanout 2) determine next fanout
SIMPLE CONF PROC	1) update source configuration 2) update and transmit fanout configuration
EVAL	1) evaluate 2) check for repeated evaluations
SCHED	1) schedule 2) timing analysis
EVENT LIST MANAGER	insert in event list

Logic Simulation Engine

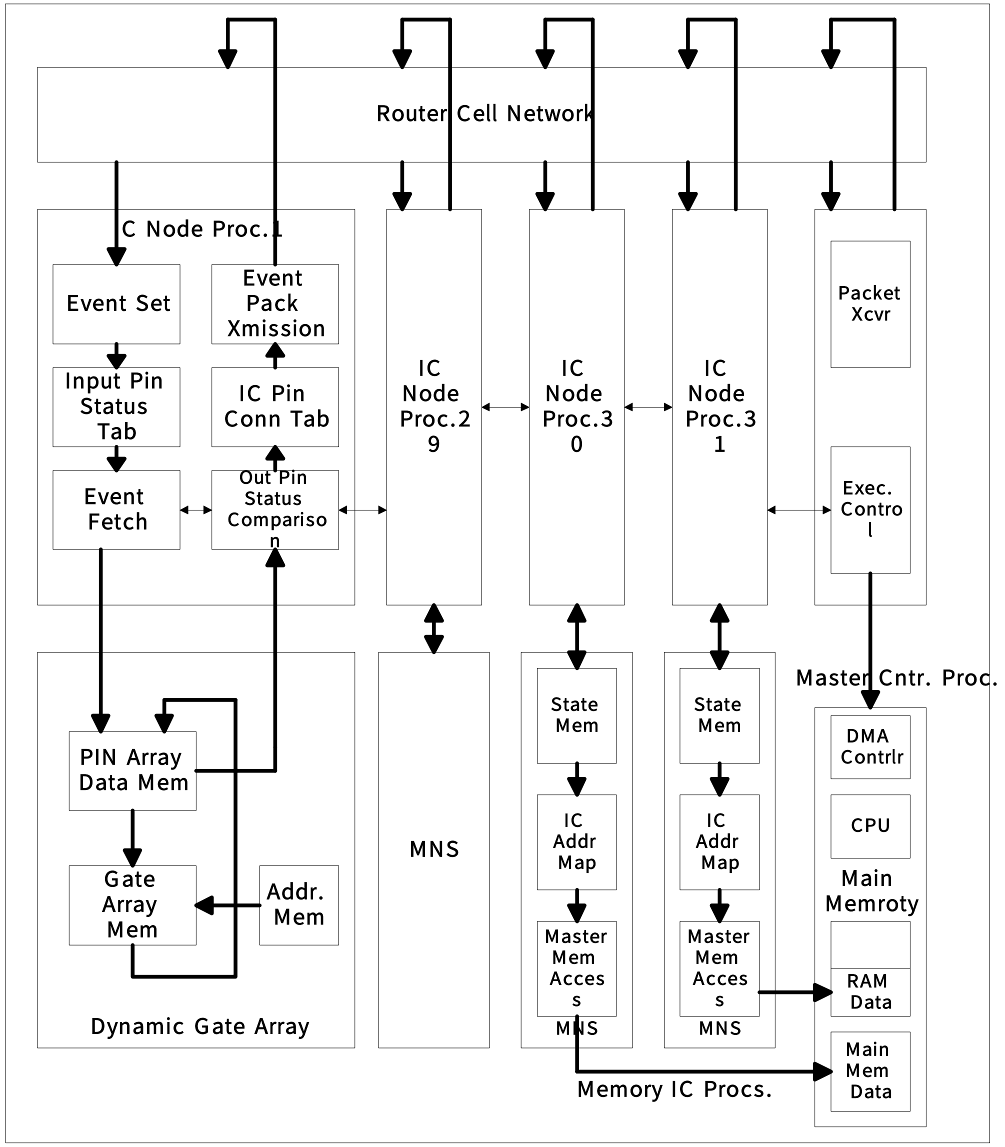


Logic Simulation Engine

- Pipeline - uniformly distribute the work load
- How to solve bottleneck of special processors or FIFO buffers
- Element evaluation is indicated as soon as one of its inputs change
- Most time consuming
  - Fanout determination
  - Gate evaluation
- Function evaluations are executed concurrently by parallel processors

HAL

- **Block Level Hardware Logic Simulator**



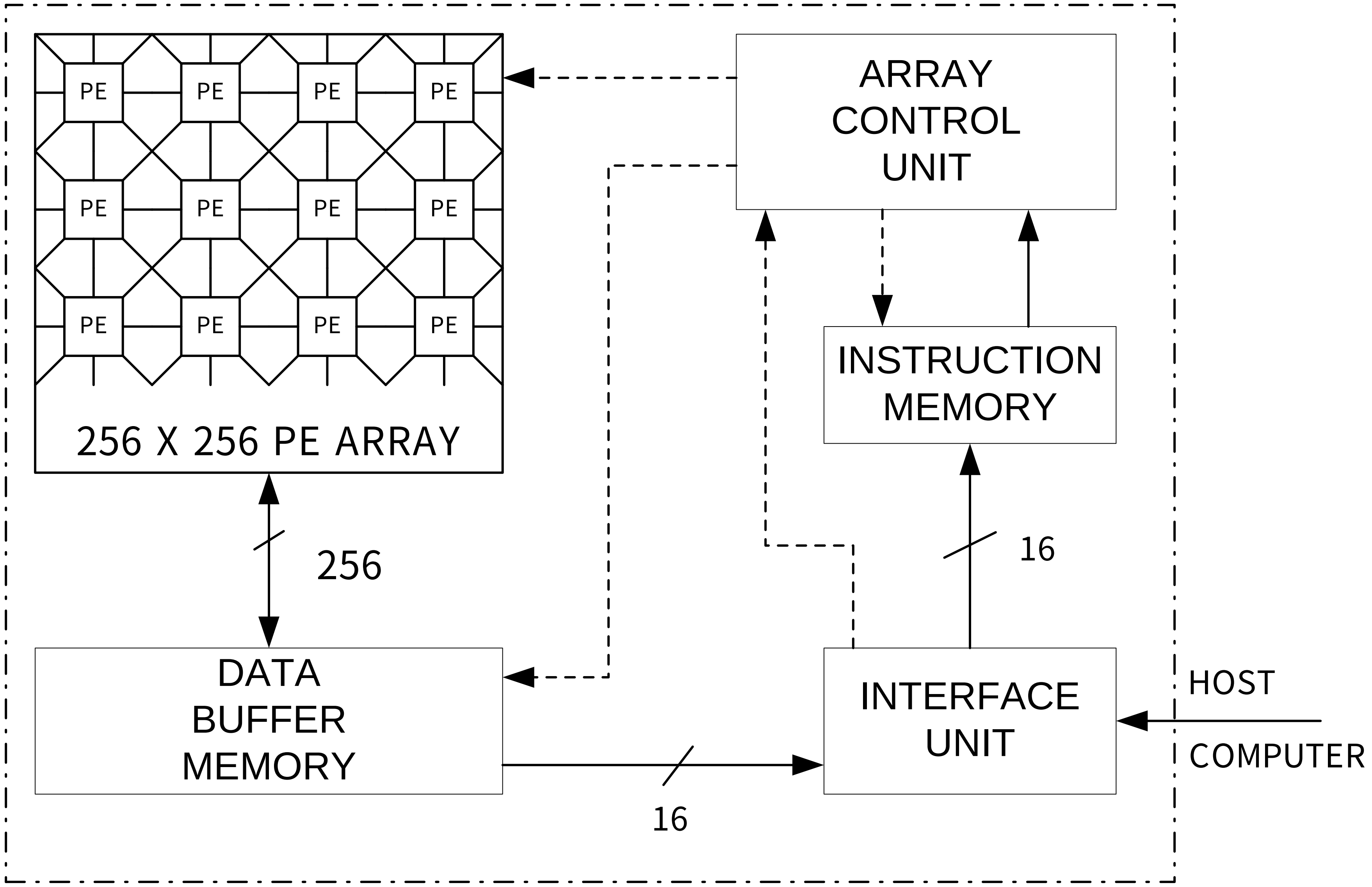
HAL

- Event search
- Signal propagation
- System clock change
- Executes a level controlled event driven mechanism
- Simulation model consists of logic blocks
- 24 bit message packets solve conflicts when more than one access

ZYCAD Logic Evaluator

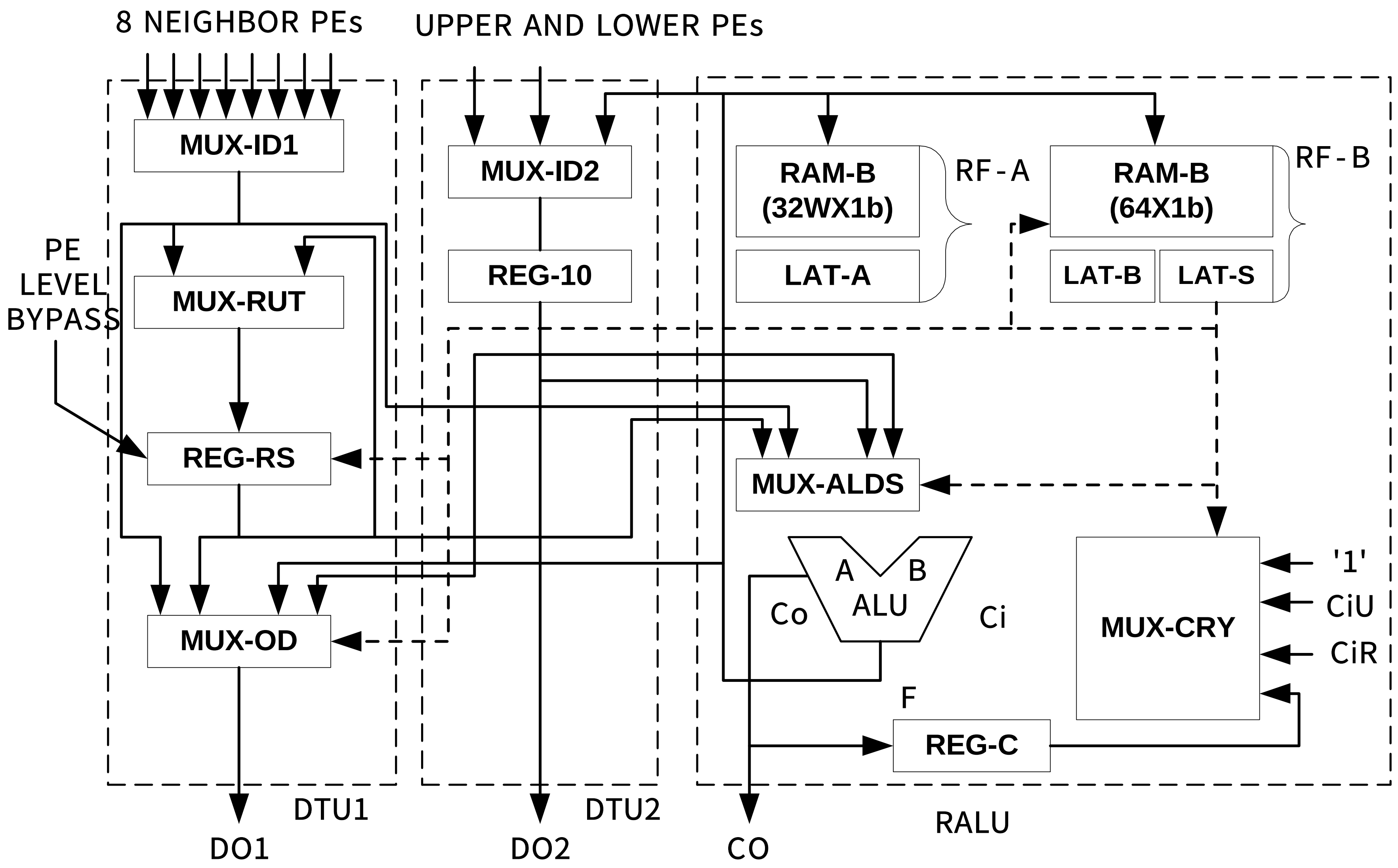
- Event driven
- Up to 16 processors
- Event processor - 5 stage pipeline
  - Get fanout list and determine delay
  - Read fanout and update output states
  - Update input states and determine gate type
  - Evaluate gate model
  - Schedule new event
- 3 input 1 output gate only
- Event stack - reduce overhead
- Future event scheduler - timing wheel

AAP-1


PE
256 X 256 PE ARRAY
ARRAY CONTROL UNIT
INSTRUCTION MEMORY
256
16
DATA BUFFER MEMORY
INTERFACE UNIT
16
HOST COMPUTER

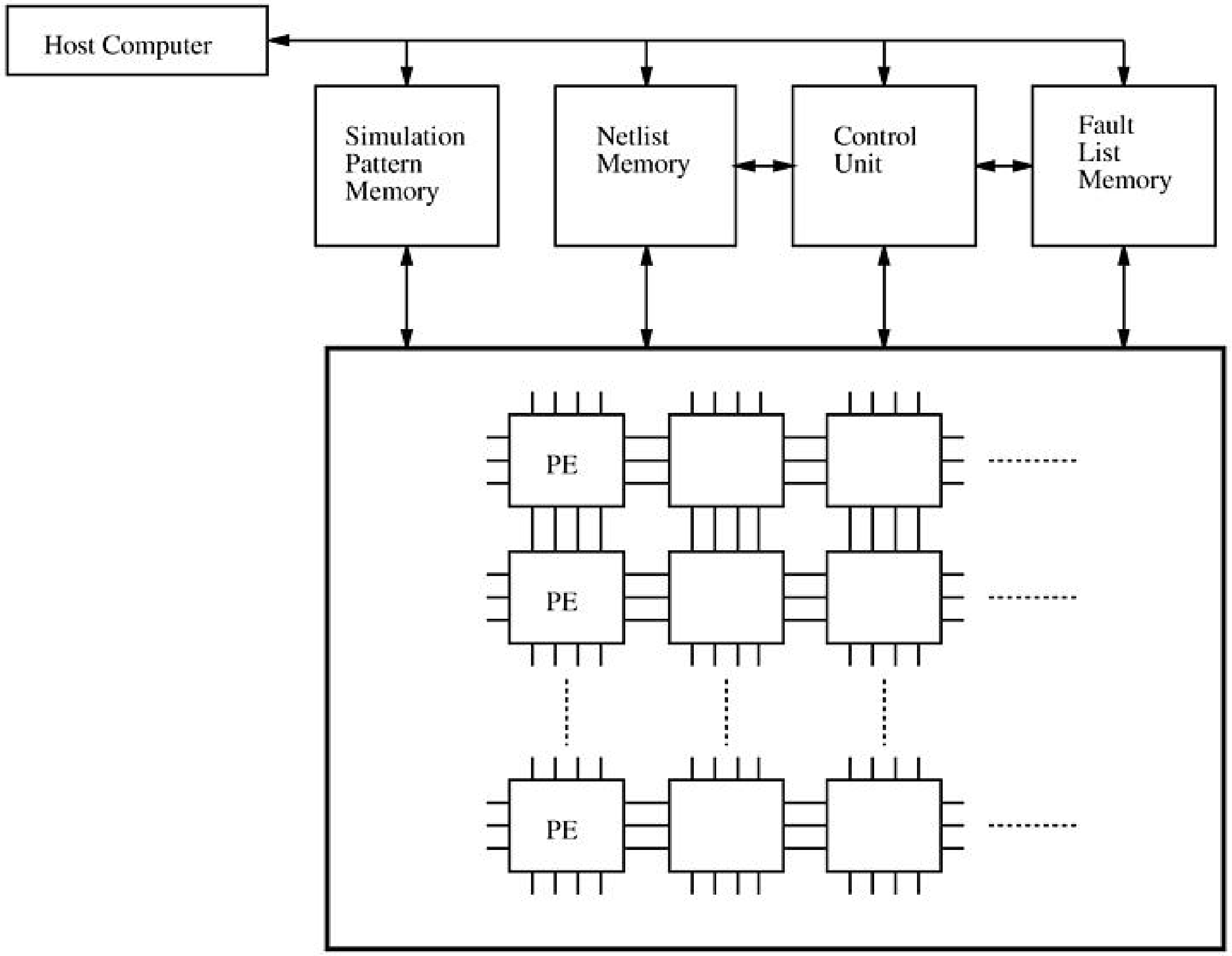
AAP-1

- **Processing Element**



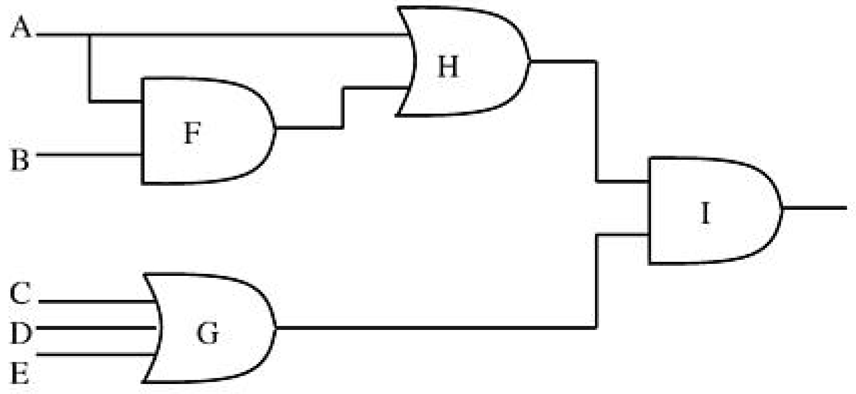
Reconfigurable Array Architecture

- **Configuration**

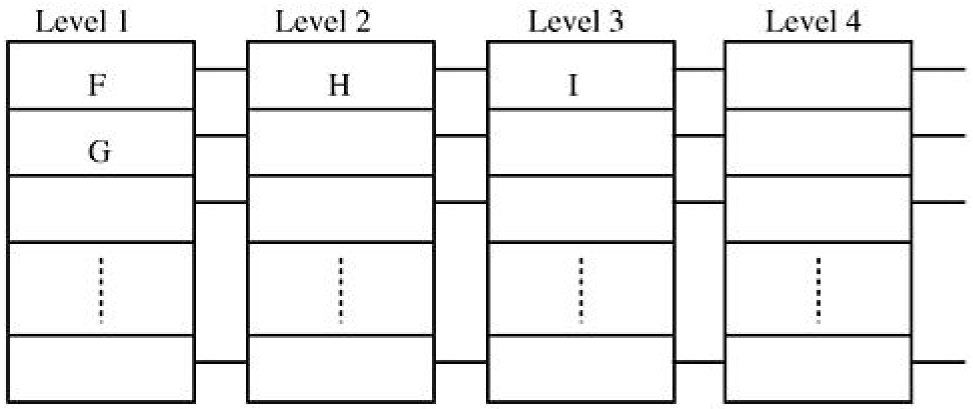


Mapping

- **Mapping onto PE Array**



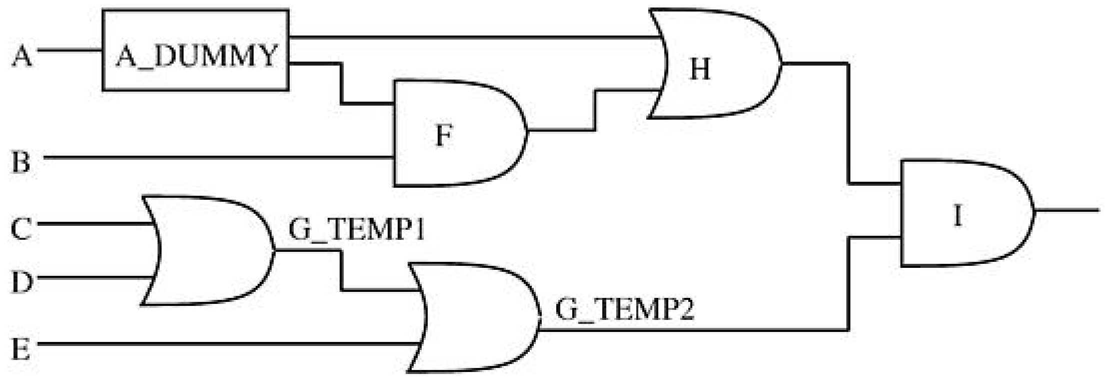
(a) Example Circuit



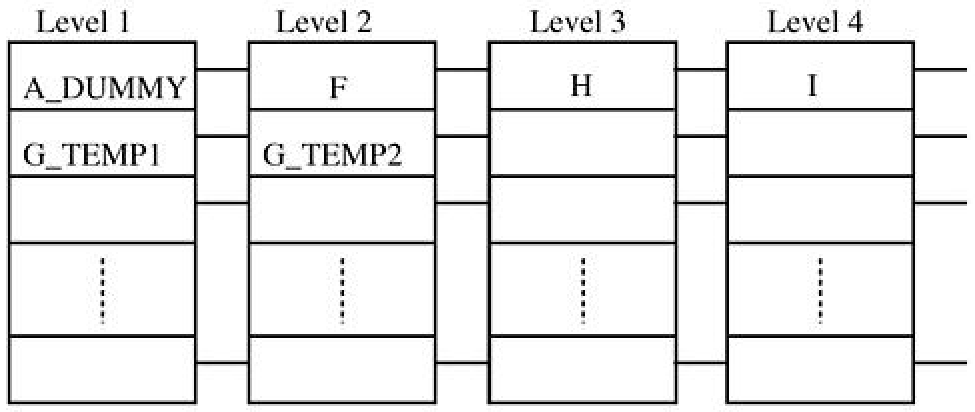
(b) Mapping onto PE Array

Mapping

- **Expanded Circuit Mapping**

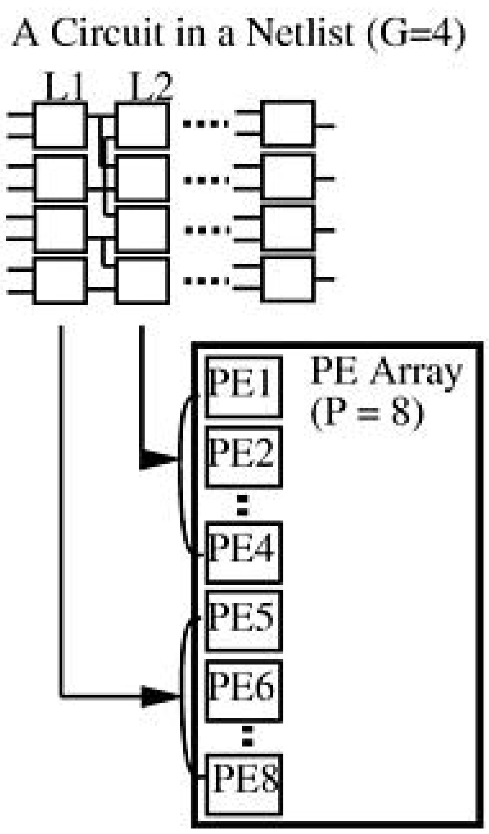


(a) Expanded Example Circuit

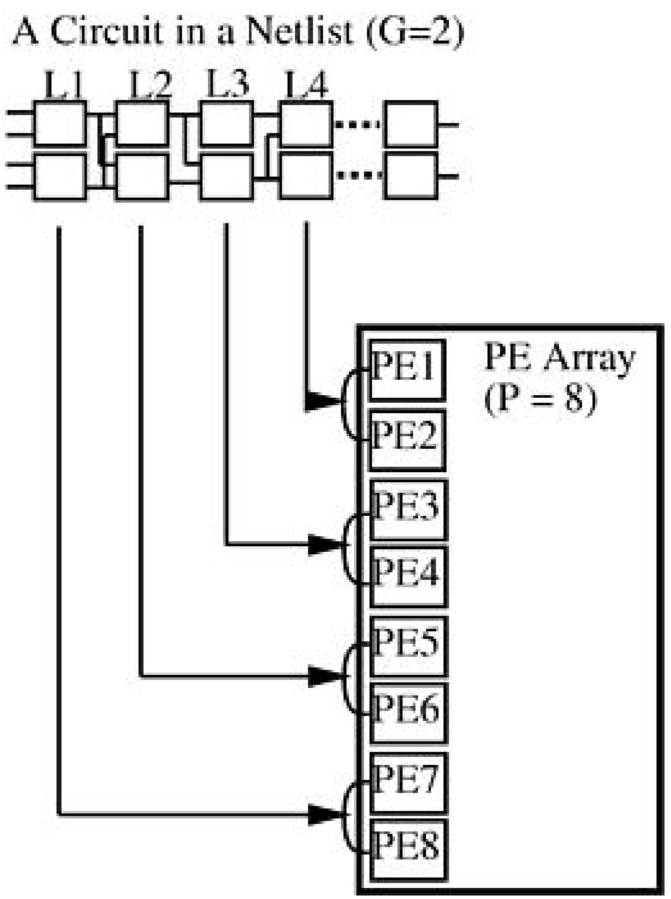


(b) Mapping onto PE Array

Array Reconfiguration



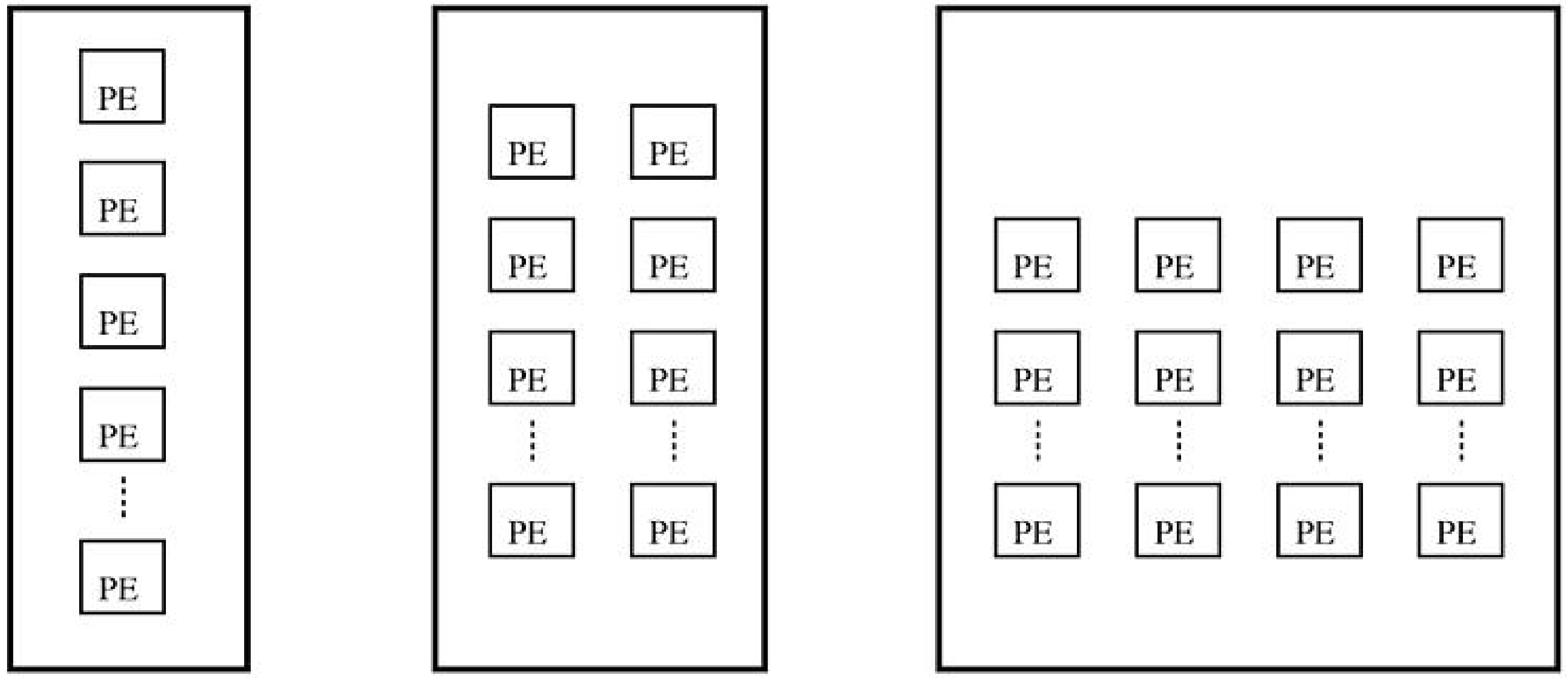
(a) Two Levels are mapped onto thePE Array (P >= 2G)



(b) Four Levels are mapped onto the PE Array (P >= 4G)

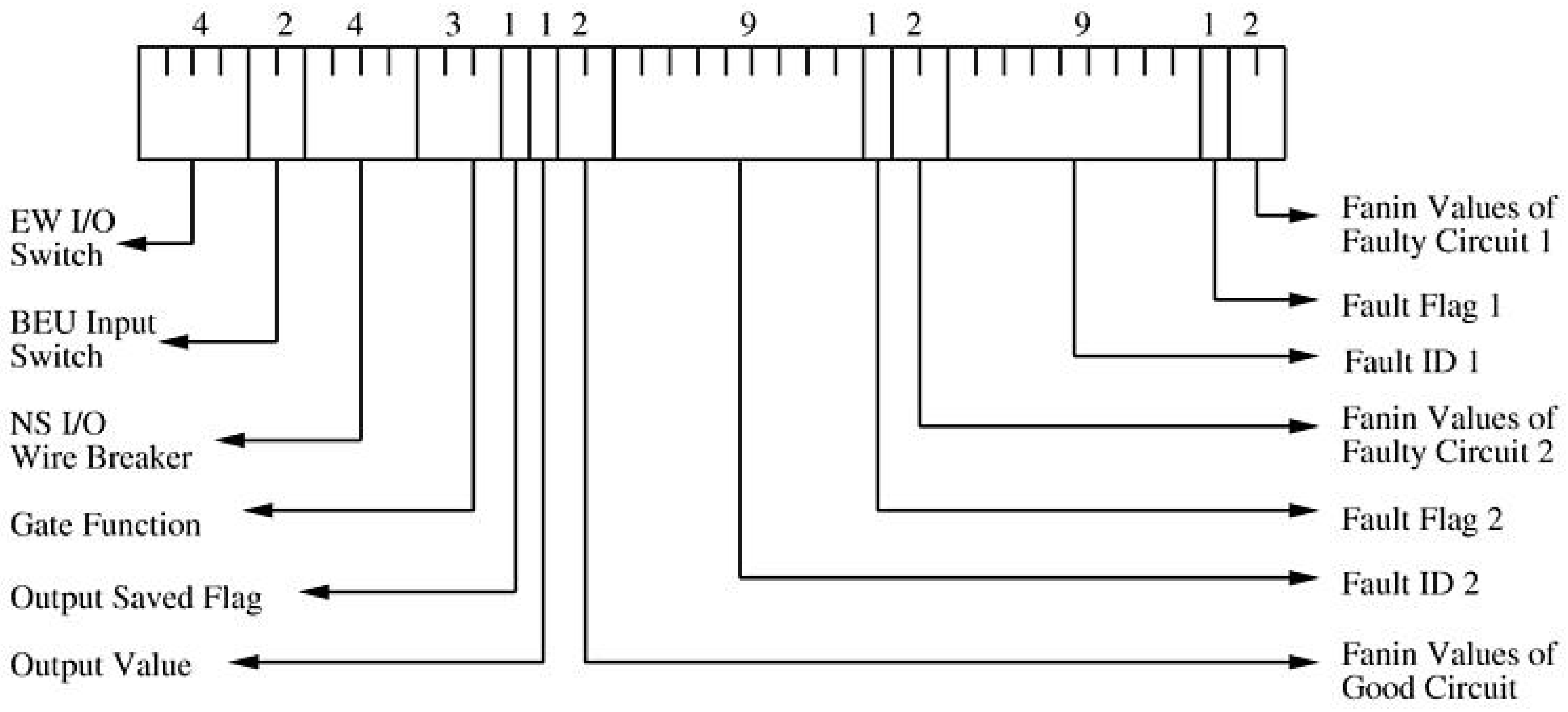
PE Array Reconfiguration

- **Folding**

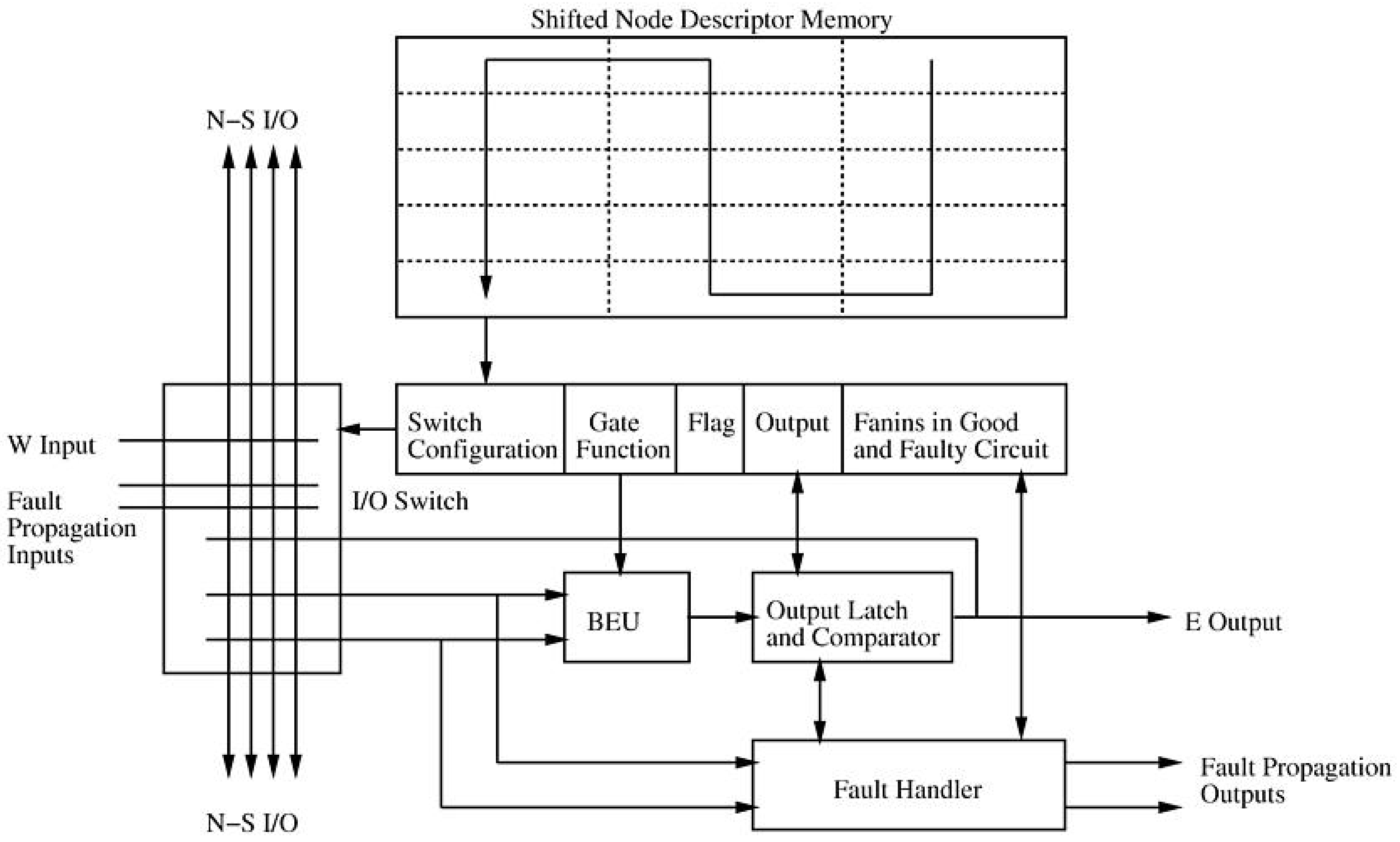


(a) P [PEs/level] with 1 level (b) P/2 [PEs/level] with 2 levels (c) P/4 [PEs/level] and 4 levels

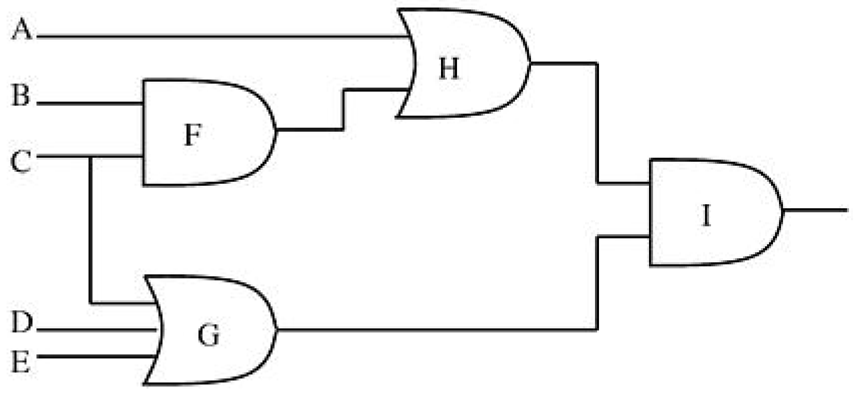
Node Descriptor Memory

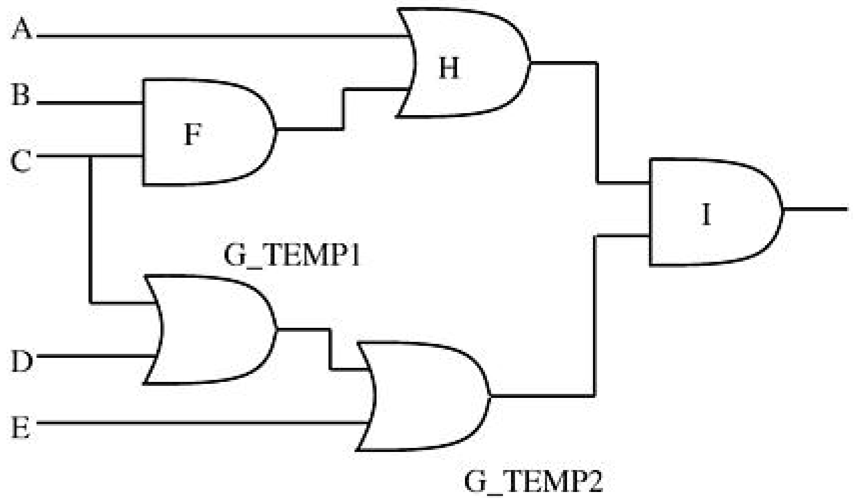


PE Cell Block

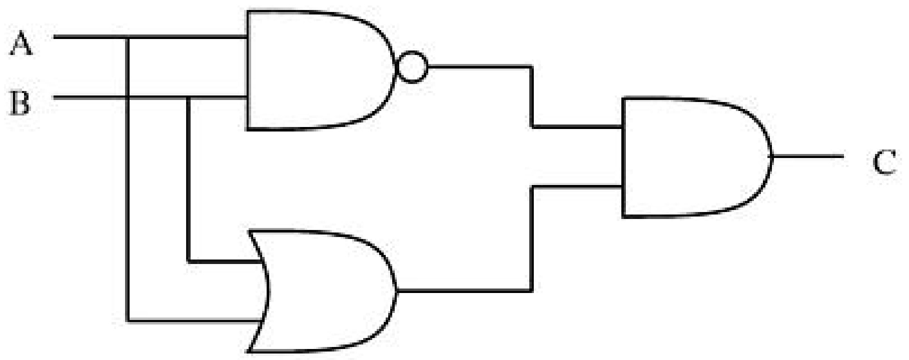


First Expansion

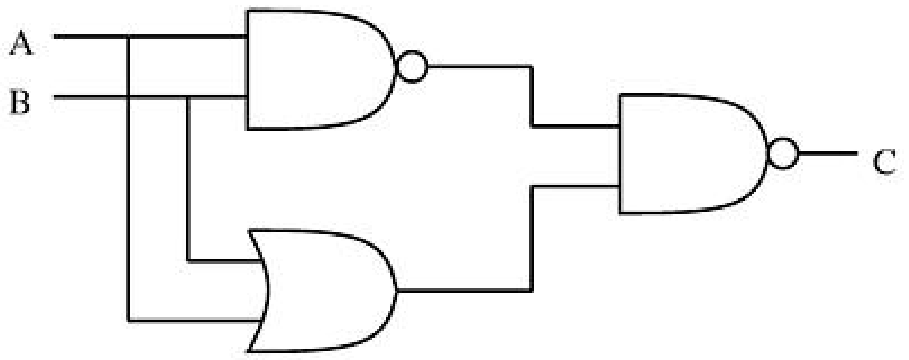




Expansion of XOR and XNOR

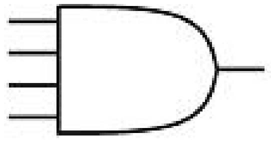


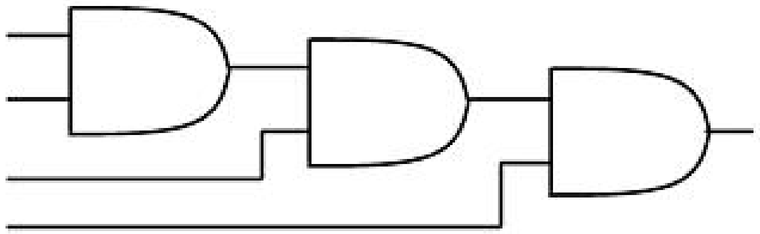
(a) XOR Gate Expansion

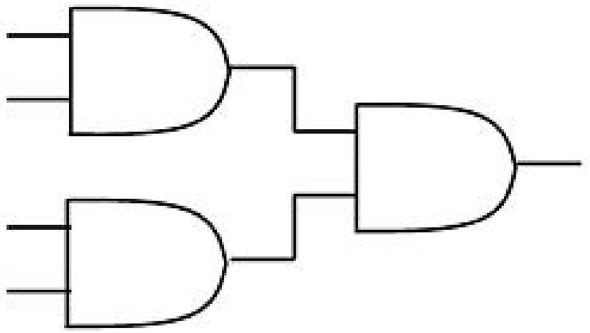


(b) XNOR Gate Expansion

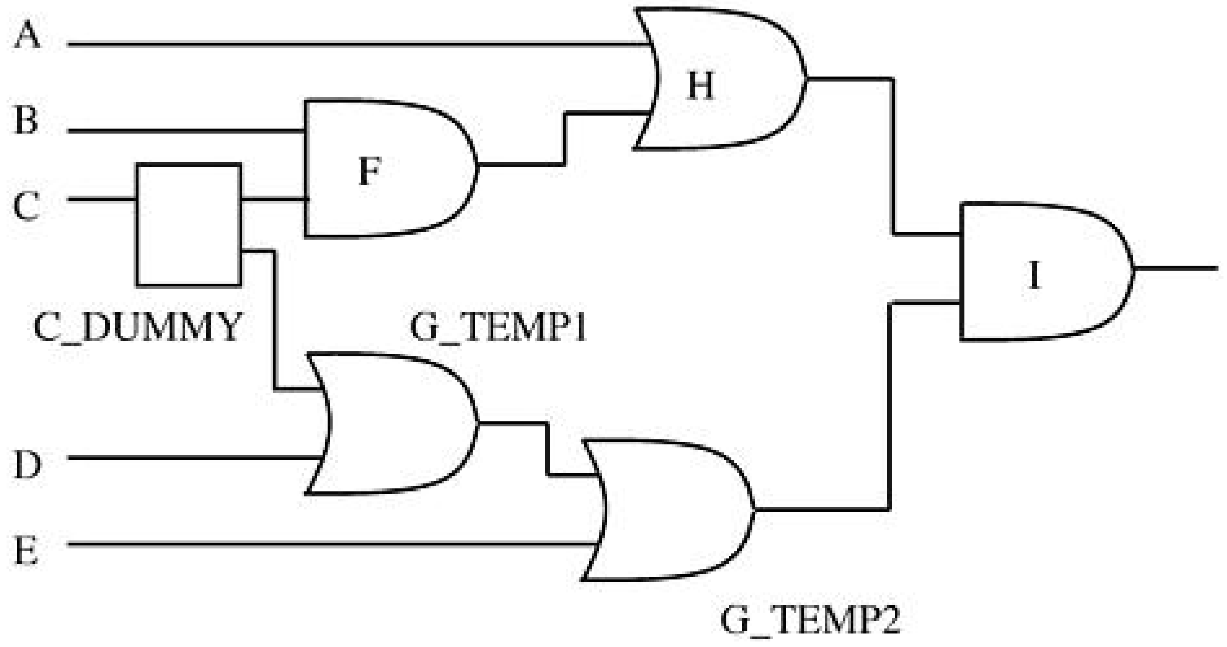
Possible Expansion of 4 Input AND


(a)


(b)


(c)

Second Expansion



Fault Simulation Algorithm

```
make a fault list attached to each element
for all patterns
    perform good simulation
    store the results of good simulation
    for all levels
        insert the first fault  in the fault list
        simulate the fault
        compare the value
        if the value is different from that of good simulation
            propagate the fault to the next level
        else
            drop the fault
        insert the first fault  in the fault list
        simulate the fault
        compare the value
        if the value is different from that of good simulation
            propagate the fault to the next level
        else
            drop the fault
    end
end
```

Logic Emulation

Sungho Kang

Yonsei University

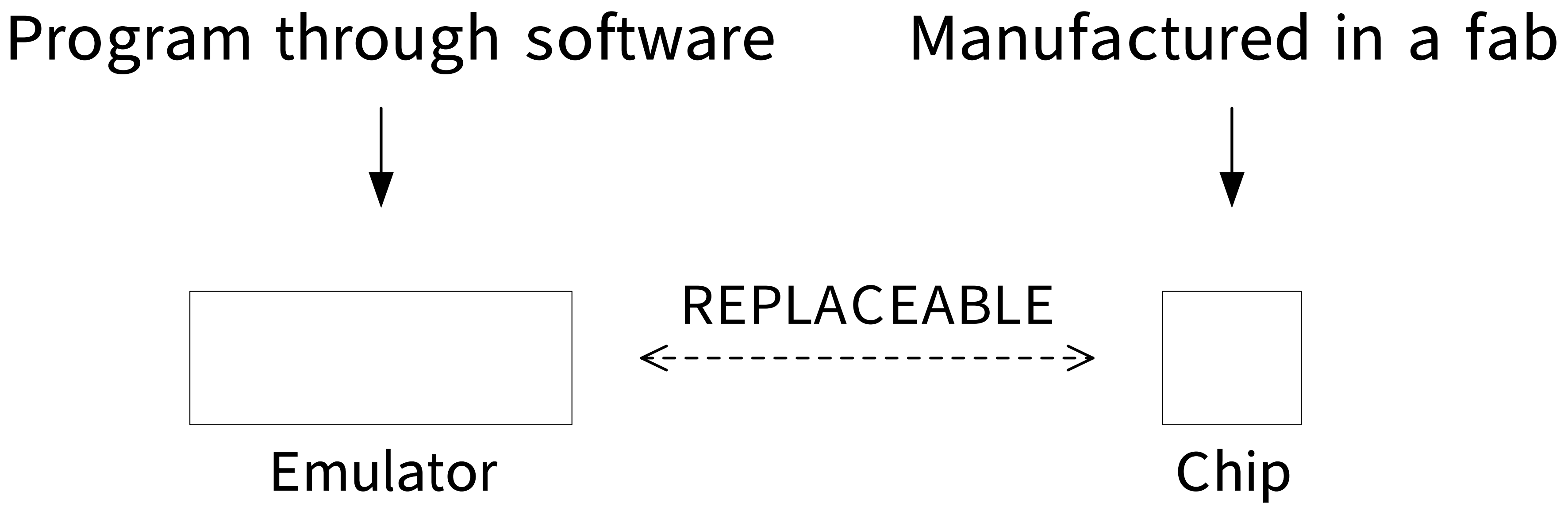
Outline

- Introduction
- Anatomy
- FPGA Architecture
- Emulation System
- Case Study
- Conclusion

Challenges for CAD

- **EDA industry**
  - 0.35% of the electronics industry
  - It is a vital enabling technology
- **CAD in service**
  - Not a contest in inventing new algorithms
  - Co-think with designers
- **Technology Trend**
  - Short life time (time-to-market pressure)
  - Submicron processing
  - High degree of system complexity
  - Merging of CPU, DSP, communicatoins, consumer electronics
  - Embedded software : complicated even to specify
  - Reuse of hardware
  - Increased field programmability

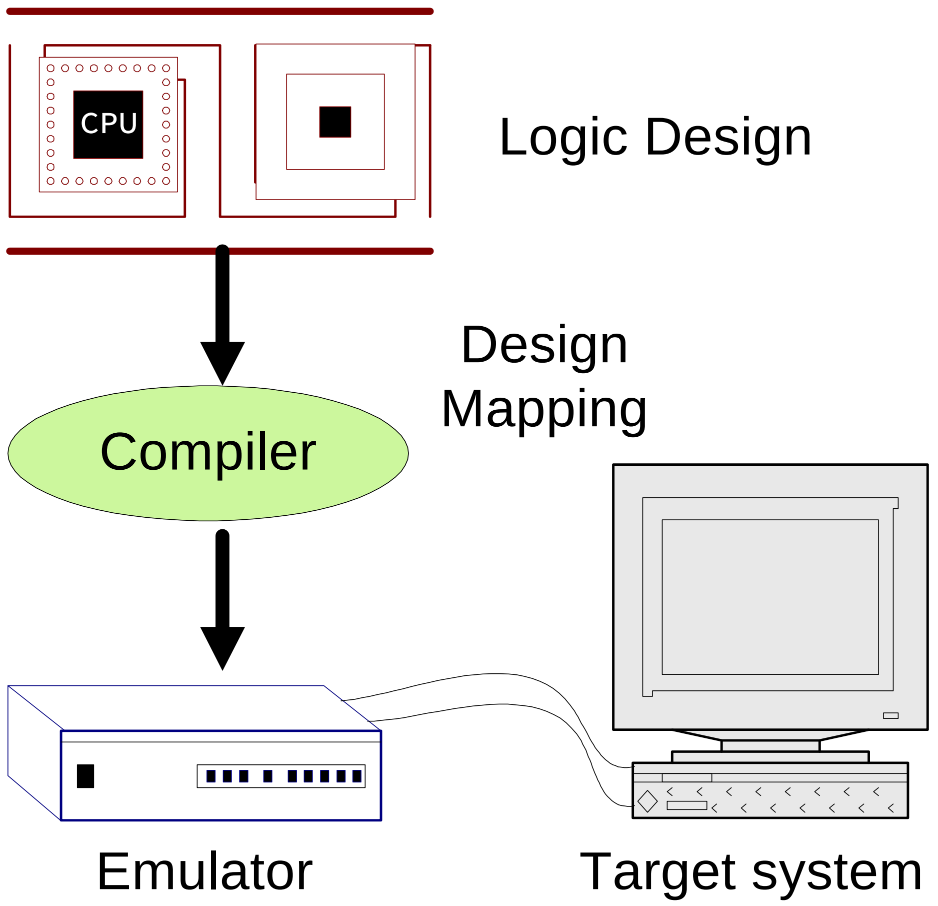
Logic Emulator



What is Emulation?

Turnkey rapid prototyping systems

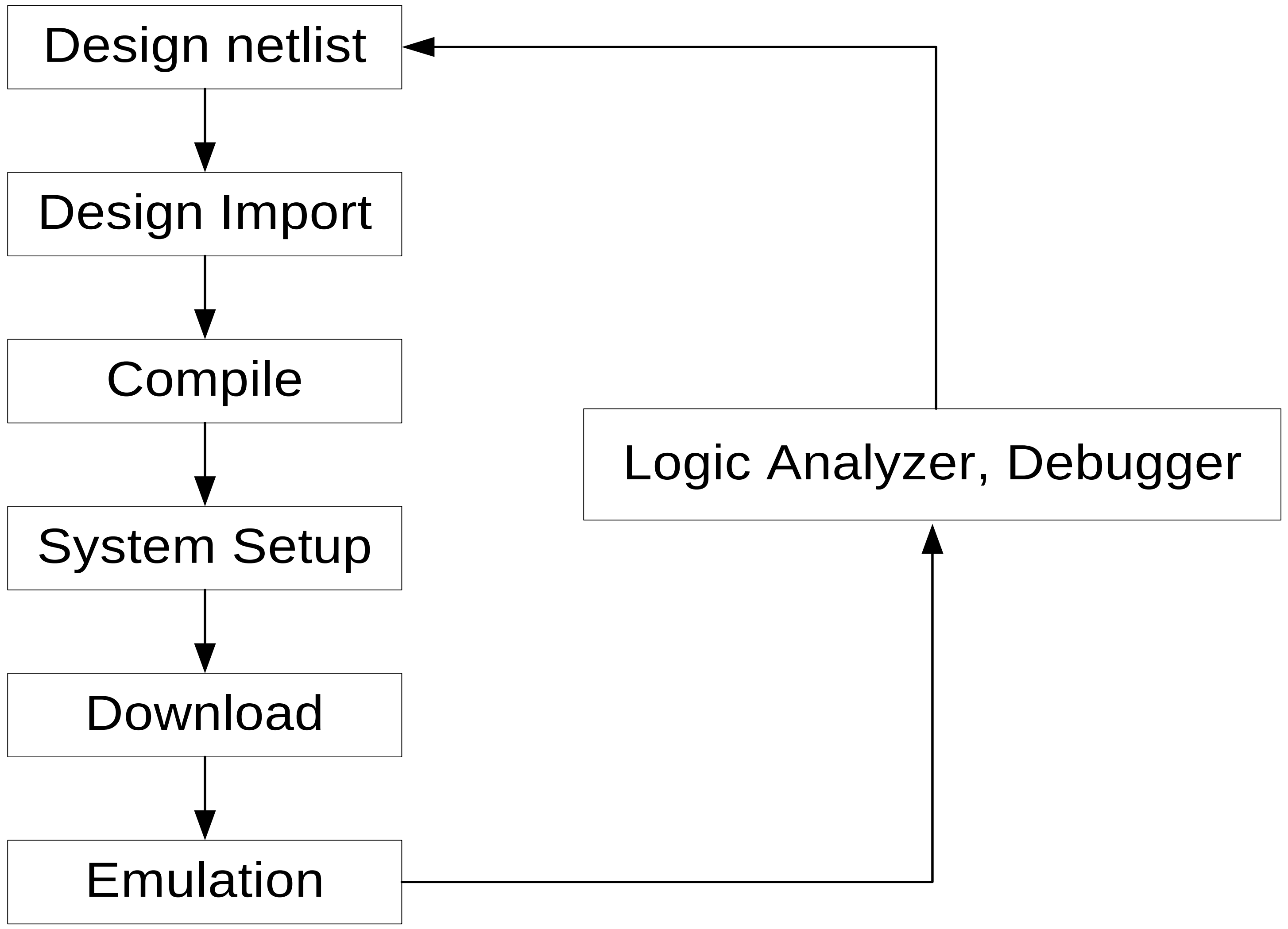
- **Read users design and automatically partition & map to array of FPGAs**
- **Enable user to run at system level and verify with application software**
- **Full internal visibility to debug - thousands of probes**
- **Modify design in minutes**



Bugs Found with Emulation:

- Functional ASIC bugs
- Board/system-level bugs
- Software, firmware bugs
- Synthesis bugs
- Bugs that require rich, real-world stimulus or high throughput to find
- Bugs caused by spec. misinterpretation

Design Flow for Emulation System



Comparison with Co-simulation

Performance potential of simulation accelerator is not achievable with current testbench strategies

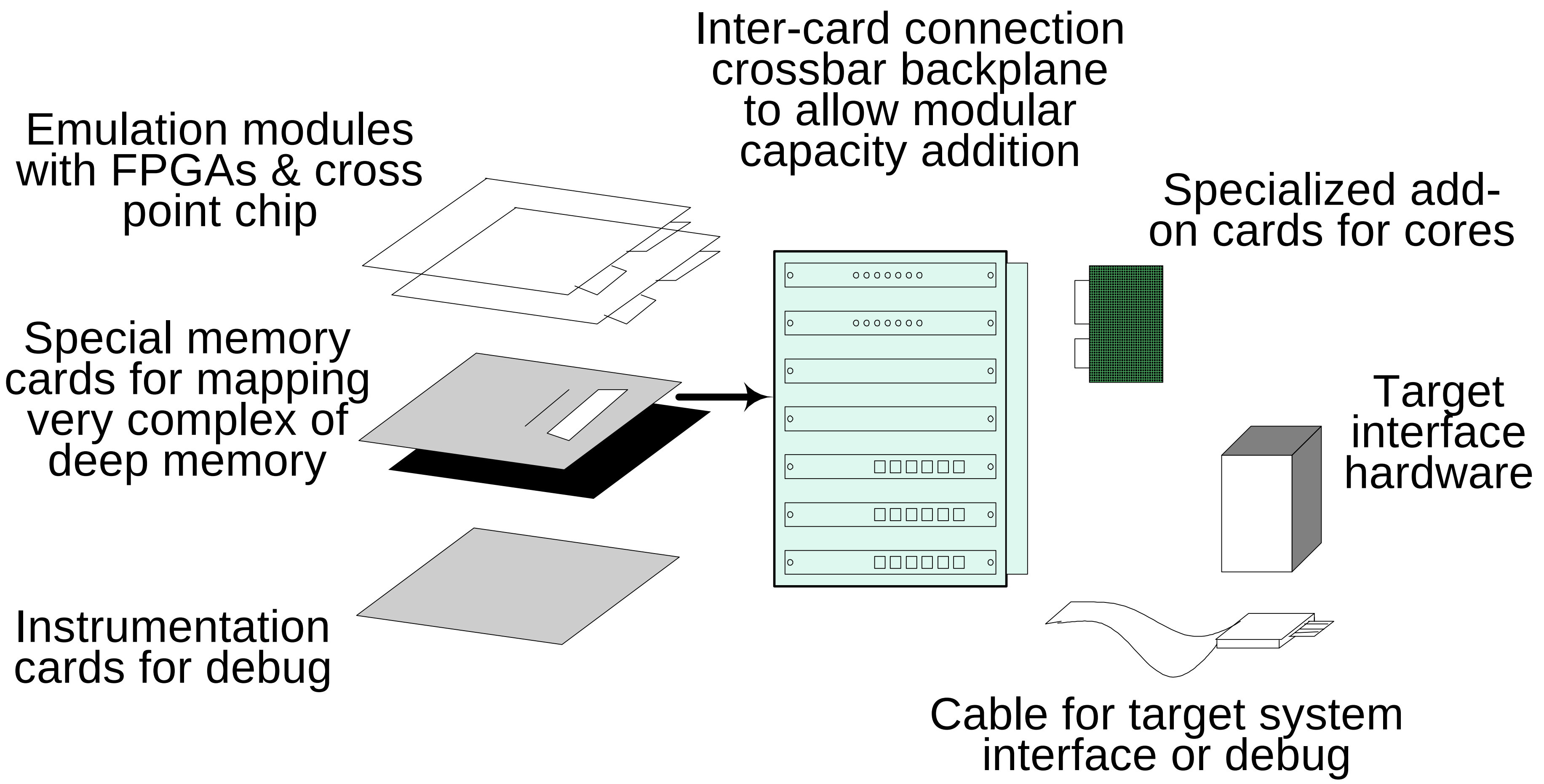
- **Speed of testbench (workstation)**
- **Channel latency & bandwidth**
- **Frequency of communication**
- **Design under test execution speed**

Comparison with FPGA

- **FPGA**
  - **High chip capacity**
  - **Slow compilation**
  - **Low I/O to gate ratio**
- **Emulation**
  - **Fast compile speed**
  - **Productive debugging**
  - **High I/O to gate ratio**
  - **On-board logic analyzer**

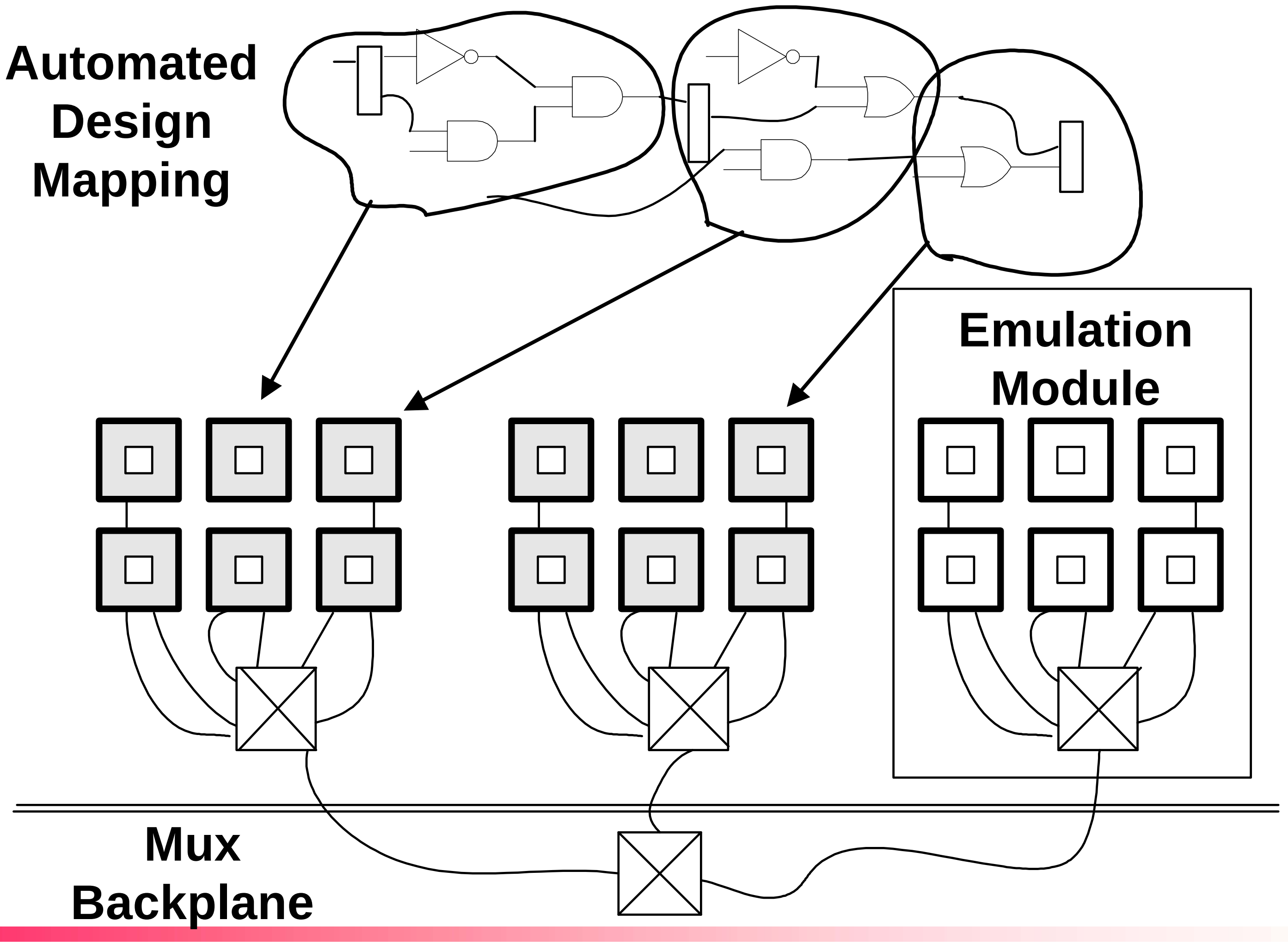
- **Generic FPGAs used for emulation**
  - **Unpredictable capacity and highly variable routing delays with poor debuggability**

Anatomy of an Emulator


Inter-card connection crossbar backplane to allow modular capacity addition
Emulation modules with FPGAs & cross point chip
Specialized add-on cards for cores
Special memory cards for mapping very complex of deep memory
Target interface hardware
Instrumentation cards for debug
Cable for target system interface or debug

Emulator Architecture

- **Hierarchical Multiplexed Architecture Simplifies Design Mapping Process**



Definition

- A logic emulator is a system of
  - Programmable hardware with capacity much greater than one FPGA
  - Software which automatically programs the hardware according to a gate level design representation
  - Software and hardware to support operation and analysis of the emulated design as a component in real hardware

System Overview : SW Components

- Design compiler
  - Netlist reader and parser :
    - Reads and parses gate-level design netlists
  - Technology mapper :
    - Maps design components into optimal emulator equivalents
  - System-level Partitioner and Placer :
    - Partitions mapped design into boxes, boards, ultimately into FPGA netlists.
  - System-level Interconnect router :
    - Determines the programming of interconnect hardware to complete nets cut by the partitioner
  - FPGA compiler :
    - Reads each FPGA netlist, maps, partitions, places and routes FPGA.
  - Timing Analysis(optional) :
    - Analyzes compiled design on emulation hardware for speed, hold violations.
- Runtime download and analysis controller.
- Graphical User Interface Hardware diagnostics

System Overview : HW Components

- Logic emulation boards : FPGAs and interconnect chips
- Memory emulation boards : RAMs, FPGAs and interconnect.
- System interconnect board : chips which interconnect emulation boards.
- I/O Connectors and Pods : connects to in-circuit interfaces, external components.
- Instrumentation : stimulus generator, logic analyzer, vector interface.
- Controller : downloads configurations, operates instruments.
- Interface : to host computer.

Why Emulate?

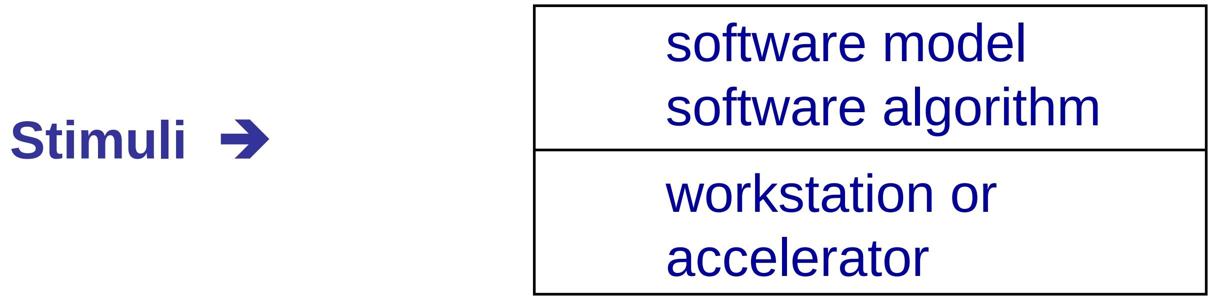
- Concurrent design verification - faster time to market
- Higher predictability of schedule - reduced project risk
- Fewer design changes in final phases - improve quality
- Lower cost for fewer silicon iterations - lower cost
- For mission critical designs - high quality
- Simulation : only methodologies are limited by processing power in verifying complex designs - better verification
- Emulation is the only verification methodology which is keeping up with system complexity

Need for System Level Emulation

- Emulation combines the flexibility of simulation and the realism of a prototype
  - Simulation : limited by the availability of software models
  - Custom prototyping : increased time to build and debug
- Leverages synthesis technology to optimize design differentiation and uniqueness
- Enable fast, incremental design changes that shorten design iteration cycles and improve quality
- Avoid costly respins of silicon and saves months of redesign
- Increased confidence in your entire project schedule and your ability to meet requirements

Design Verification Methodologies

- **Simulation : allows observation of a fraction of real world interactions (sequential)**



- **Emulation : enables the designer to explore alternatives (parallel)**
  - **Verification takes place in a hardware environment**
  - **The design is retargeted to a programmable hardware environment**
- **Rapid prototyping : allows operating speeds such that all interfaces to target applications can operate in real time.**
  - **If the system runs at real time, the quality of the algorithm can be evaluated on the fly and DSP design time can be greatly reduced.**

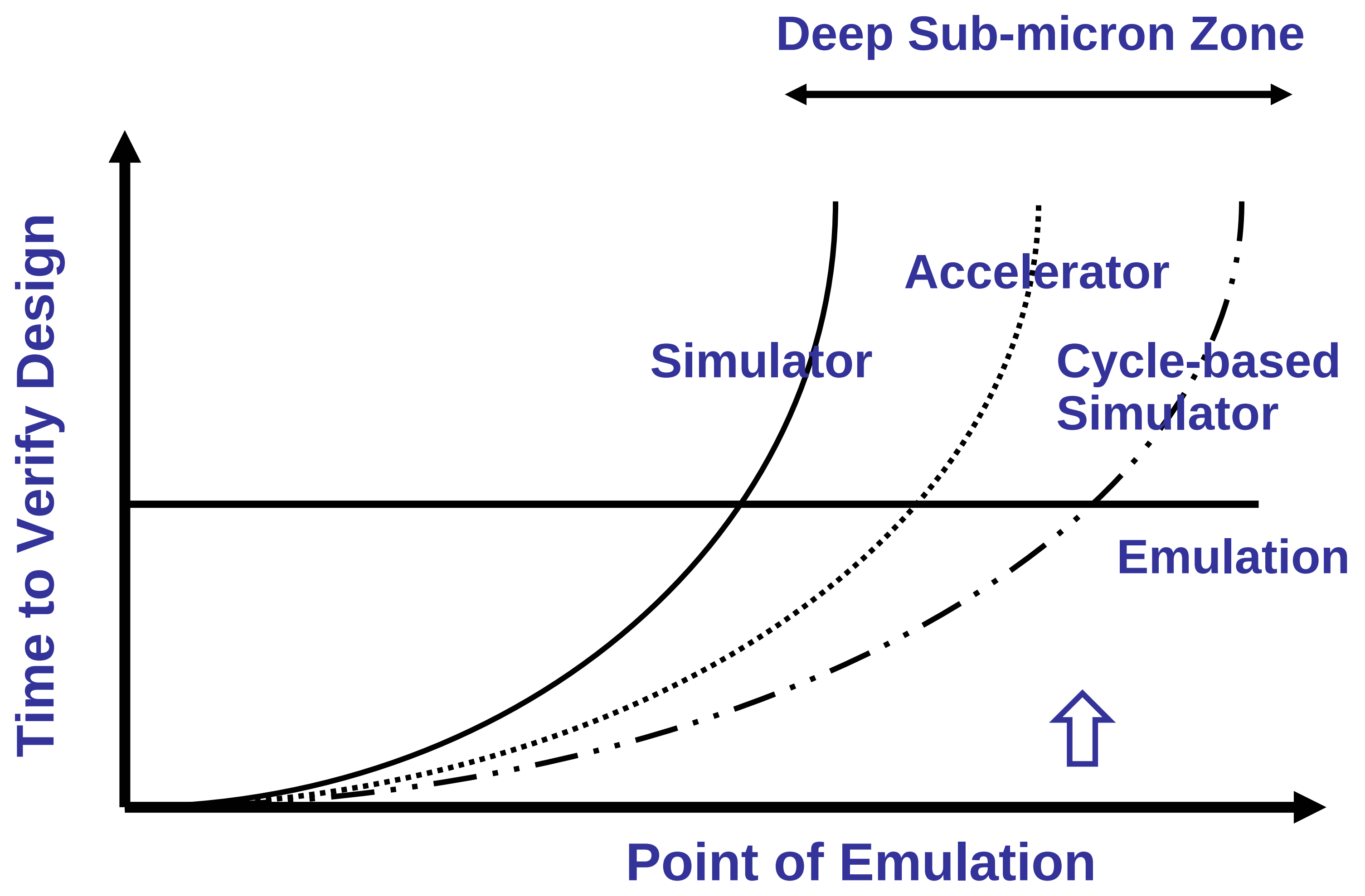
Advantages of Emulation

- Emulation is the only verification methodology which is keeping up with system complexity

Time		Speed up factor		
1 second	---------	10^7 ---------	←←	Actual Hardware
10 second	---------	10^6 ---------	←←	Logic Emulation
2 minutes	---------	10^5 ---------		
16 minutes	---------	10^4 ---------		
3 hours	---------	10^3 ---------		
1 day	---------	10^2 ---------		
12 days	---------	10^1 ---------		
3 months	---------	1 ---------	←←	Software Simulations

Advantages of Emulation

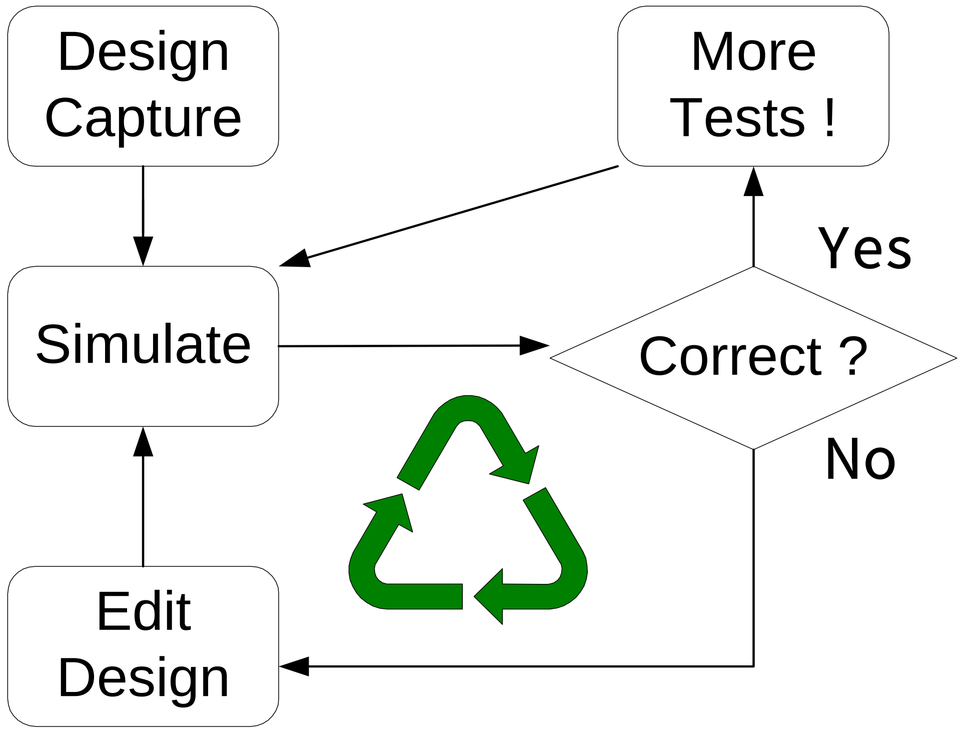
- Emulation performance is not a function of design size



Motivation : Verification Cycles

- **Simulation is the most compute-intensive problem in EDA**

It's Inside the Design Cycle



Design Sizes are Growing

- **2X Design Size → 2X work / vector.**
- **2X Design Size → 2X vectors.**

⇨ **2X Design Size → 4X workload**

Motivation : Verification Realism

- **Often the chip meets its spec, but does not work in the system :**
  - **Spec errors, misunderstandings:**
    - **" I thought your chip was handling that…"**
  - **Real system puts designs into unanticipated situations:**
    - **Interaction between components across time and function: Combinatorial Explosion**
      - **i.e. the Ethernet driver interrupts a page fault which is servicing a floating point exception.**
    - **Other parts of system don't adhere to their specs, or their specs aren't known:**
      - **Undocumented behavior in other devices, such as CPU,**
      - **Peripherals from other projects or other vendors.**

Motivation : Verification Realism (cont.)

- Some applications need real-time operation for verification :
  - Display are far easier to verify by actual observation
  - Closed-loop operation with analog hardware
  - Electro-mechanical controllers
  - Human perception : audio, video compression, processing

- Simulation generally requires test vector development:
  - Costly and difficult, critical path in schedule,
  - Verification depends of test vector correctness,
  - Test vectors may have to be based on assumptions,
  - Test vectors are intrinsically open-loop.

Motivation : Verification Realism (cont.)

- **Only when the real design is running its real application in its real environment is correctness assured.**

- **Emulated design connected to actual hardware can run:**
  - **actual diagnostic code, compatibility tests,**
  - **actual operating systems,**
  - **actual applications,**
  - **receiving real data from storage, sensors, devices,**
  - **sending real data to storage, devices, displays.**

Motivation : Visibility

- Once a chip is fabricated, placed in a system, and fails
  - Internal probing is impossible
  - It may be difficult or impossible to put the simulation into the failing state for analysis
- Emulated design can have internal probes programmed in, for direct connection to instrumentation
- Emulated design may be used to generate test vectors for fabrication

Rapid Prototyping

- Once emulated design is debugged, it is available for immediate use by software developers.
  - This can directly reduce the projects;'s critical path and time-to-market.
- Emulated design is available for demonstration to customers, users, management.
  - Proof of concept, proof of progress.
  - Find out early whether result will be satisfactory.
- Architectural workbench :
  - Drive emulation with RTL-level synthesis.
  - Experiment with architectural features on real code and data:
    - structures, sizes, algorithms of caches, busses, buffers,
    - quantity and design of functional units,
    - novel architectures, representations, algorithms. etc., etc.,

Disadvantage of Logic Emulation

- Hardware emulation system is required
- Speed is 5-10 X slower than real design speed
  - System emulation speeds of 1 to 4 MHz are common today
  - Target system must be slowed down for emulation
- Delays do not match those of real design
  - Timing-induced errors are possible, that is, hold-time violation
  - Delay independent functionality may not operate correctly

Logic Emulation

- **FPGA-based Hardware Emulation**
  - **Contain a large pool of general purpose logic block**
  - **Design preparation time and compilation time are costly**

- **Processor-based Hardware Emulation**
  - **An array of basic CPUs or simple Boolean processors that perform basic logic operation on a time sharing basis**
  - **Design under verification is converted into a simulation data structure, similar to that of a software simulator**
  - **Slower than FPGA-based**

FPGA Architectures for Emulators

- **Partitioning, placement and routing optimized for the emulator performance**
- **FPGA efficient interconnection technology**

  (currently use ~25% of FPGA logic gates)
- **Interconnection of logic blocks, of multiple FPGAs, of multiple emulator modules**
- **Incremental design change**
- **Observability and controllability of debug process**
- **Memory resource**

  (separate memory or FPFA RAM)
- **Clock lines**

  (low skew, no setup and hold time violations)
- **Lower cost (than silicon)**

Interconnect Problem

- It is critical to maximize gate capacity and speed by packing as much logic into each FPGA as possible.

- Interconnect hardware architecture must:
  - provide successful connectivity in all cases.
  - Permit maximum logic utilization of the FPGAs,
  - with minimum added delay and skew,
  - at minimum hardware cost.

- Rent's Rule applies:
  - Observation of Rent at IBM in 1960's:
    - pincount of arbitrary subpart of a digital system is proportional to a fractional power of the gatecount.

      P = K * G ^ r
  - Example in high-performance systems:

    pins = 2.5 * gates ^ 0.56

Interconnect Problem

- **Commercial FPGAs are sized for engineered applications:**
  - **Designed is designing for FPGA structures.**
  - **Designer architects system so that subparts fit into FPGA pincounts.**
  - **Vendors design FPGAs accordingly.**

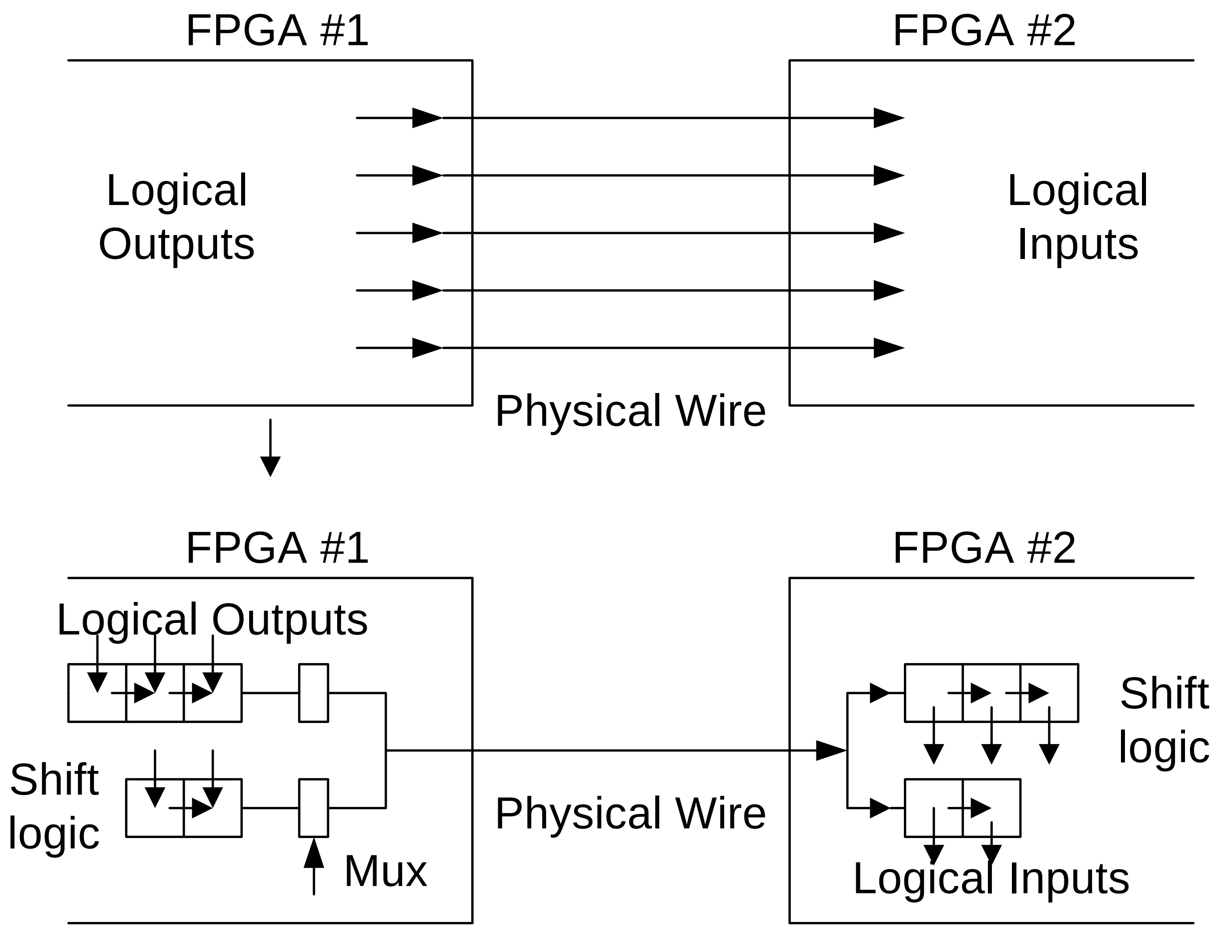
- **Emulated designs are completely arbitrary :**
  - **Structures are not optimum for FPGAs**
  - **FPGA size and pincount is arbitrary.**
  - **FPGA subparts are automatically extracted by partitioner.**
  - **Result :**
    - **FPGAs are pin-limited, not gate-limited.**
    - **Logic emulator gets 20-30% as much gate utilization as ordinary FPGA applications.**
  - **There is a challenge for interconnect architecture and software to maximize gate utilization FPGA pincount.**

Field Programmable Interconnects

- Aptix FPIC
  - A place and route architecture (not a crossbar)
  - Routing delay not controllable
  - 940 user programmable I/Os
- IQ 160
  - 176X176 crossbar
  - Every port can be configured to connect to any port
  - Routing delay is predictable
- FPGAs
  - A place and route architecture
  - Routing delay not entirely controllable
  - Typically < 300 programmable I/Os

Virtual Wires (IBM)

- **Increase bandwidth by multiplexing**
  - **> 80% gate utilty, but decrease emulation speed.**

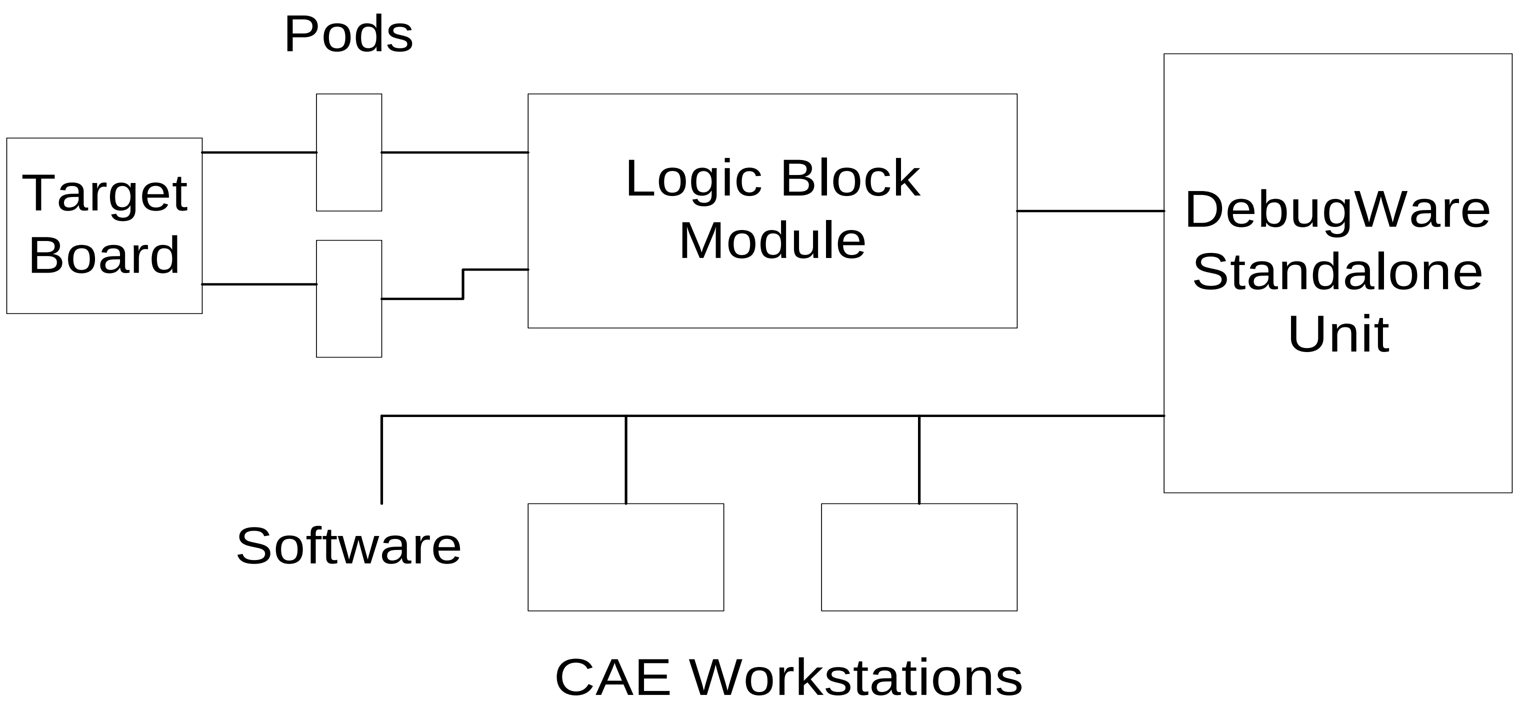


Partial Crossbar Interconnect

- Useful with the ability of FPGAs to freely assign pins
- Each small full crossbar chip is connected to the same subset of pins on each logic chip
- To configure the system, logic partitioning, placement, routing are performed
- Placement is insignificant, since the interconnection is symmetrical
- Interconnect router is a simple repetitive table-driven task

MARS (Quickturn)

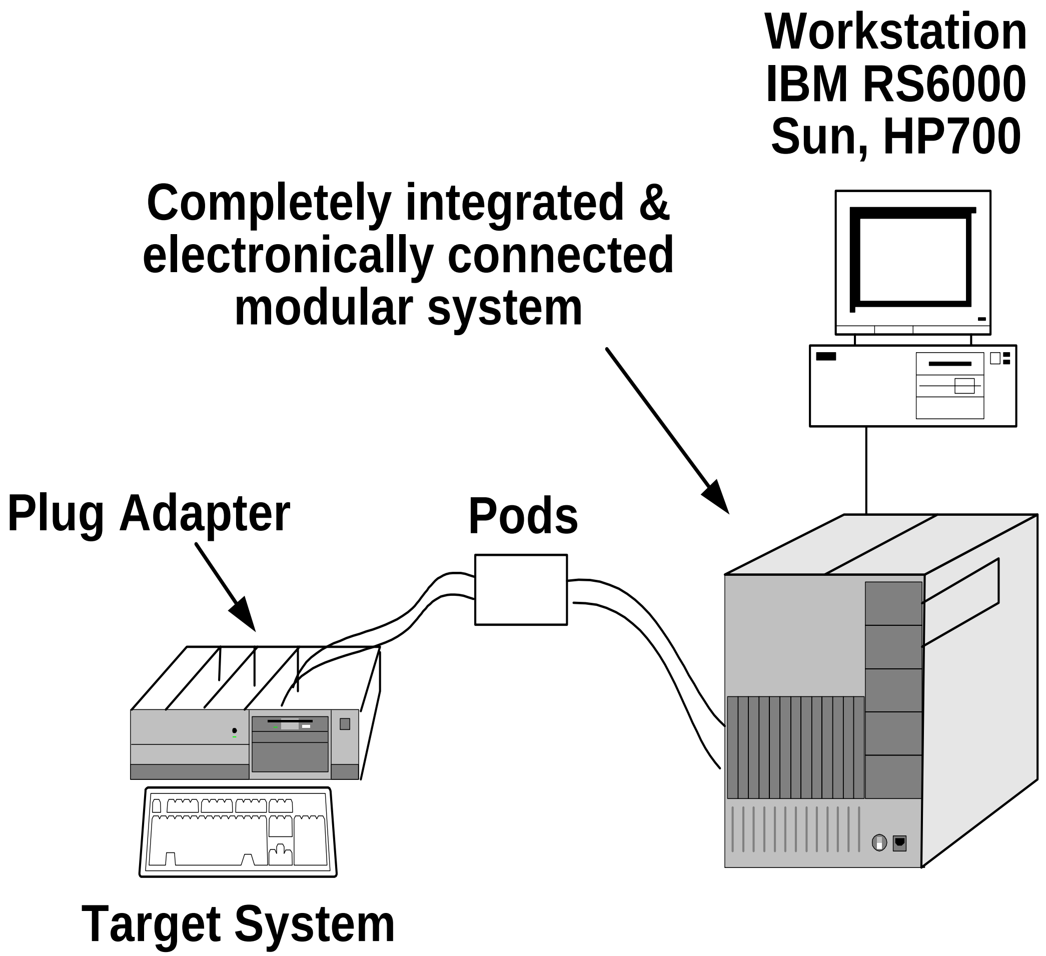
- **MARS System from PIE Design**
  - **Based on Xilinx XC4005 and XC4003**
  - **Partial crossbar system**
  - **500K gates in one box**
  - **Emulated 2 million gates design**



Realizer (Quickturn)

- **Automatically configures a network of FPGAs to implement large digital logic designs**
- **Logic and interconnections are separated to achieve optimum FPGA utilization**
- **Interconnection by partial crossbars reduces system-level complexity**
  - **Achieves bounded interconnection delay**
  - **Scales linearly with pin count**
  - **Allows hierarchical expansion in a fast and uniform way**
- **Applications**
  - **Prototyping for real time verification and operation**
  - **High-speed simulation accelerator**
  - **VHDL-driven architecture workbench**

Enterprise Emulation System


Workstation
IBM RS6000
Sun, HP700
Completely integrated & electronically connected modular system
Plug Adapter
Pods
Target System

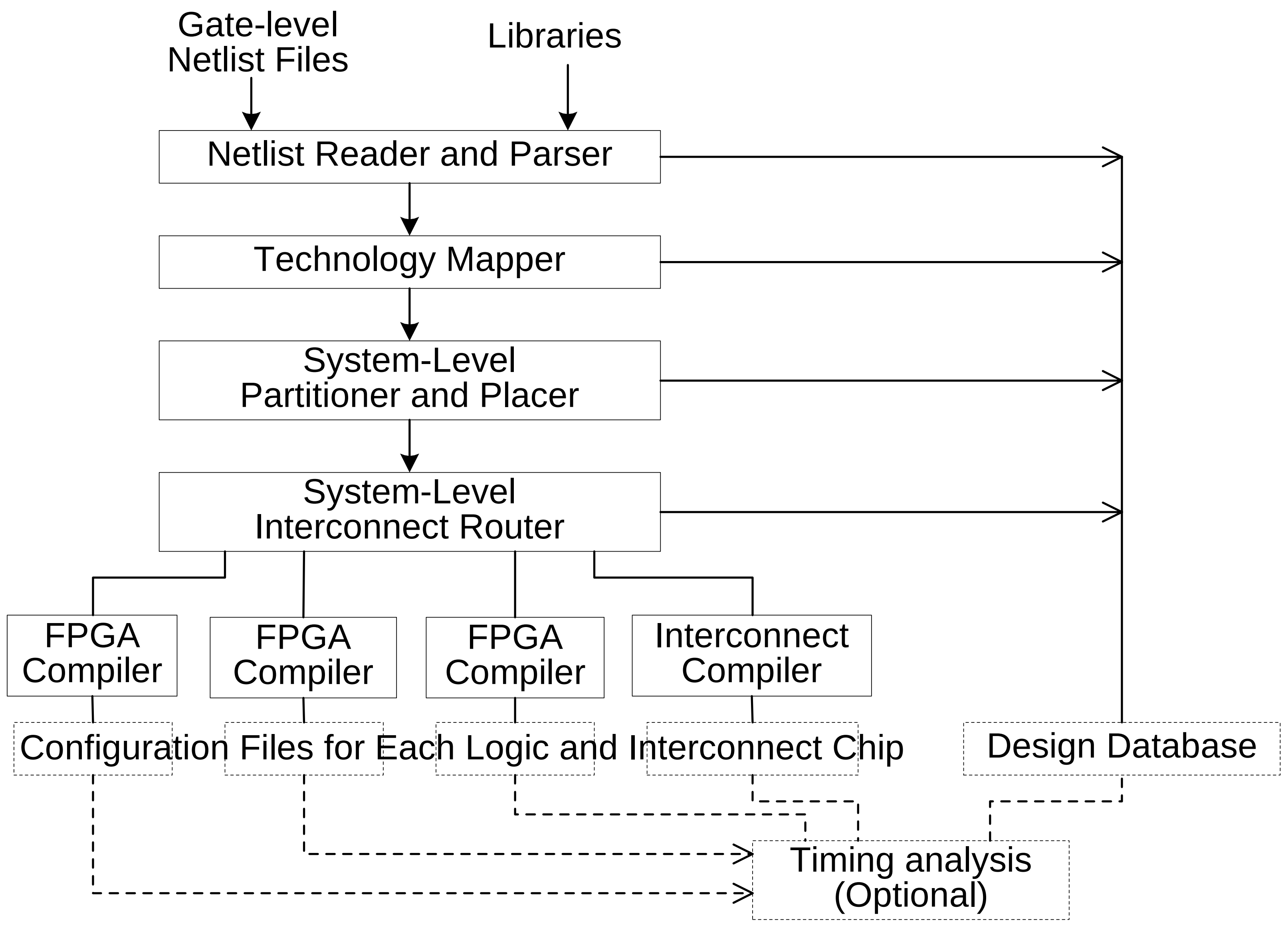
330K gates + 64 MB memory Enterprise system

- Precision Emulation Software^{TM}
- Electronically Programmable backplane bus
- Debug environment & software
- Built-in Logic Analyzer & Stimulus Generator
- Memory debug software with Memory Emulation Modules
- Built-in Workstation interface for co-simulation

Enterprise Emulation System

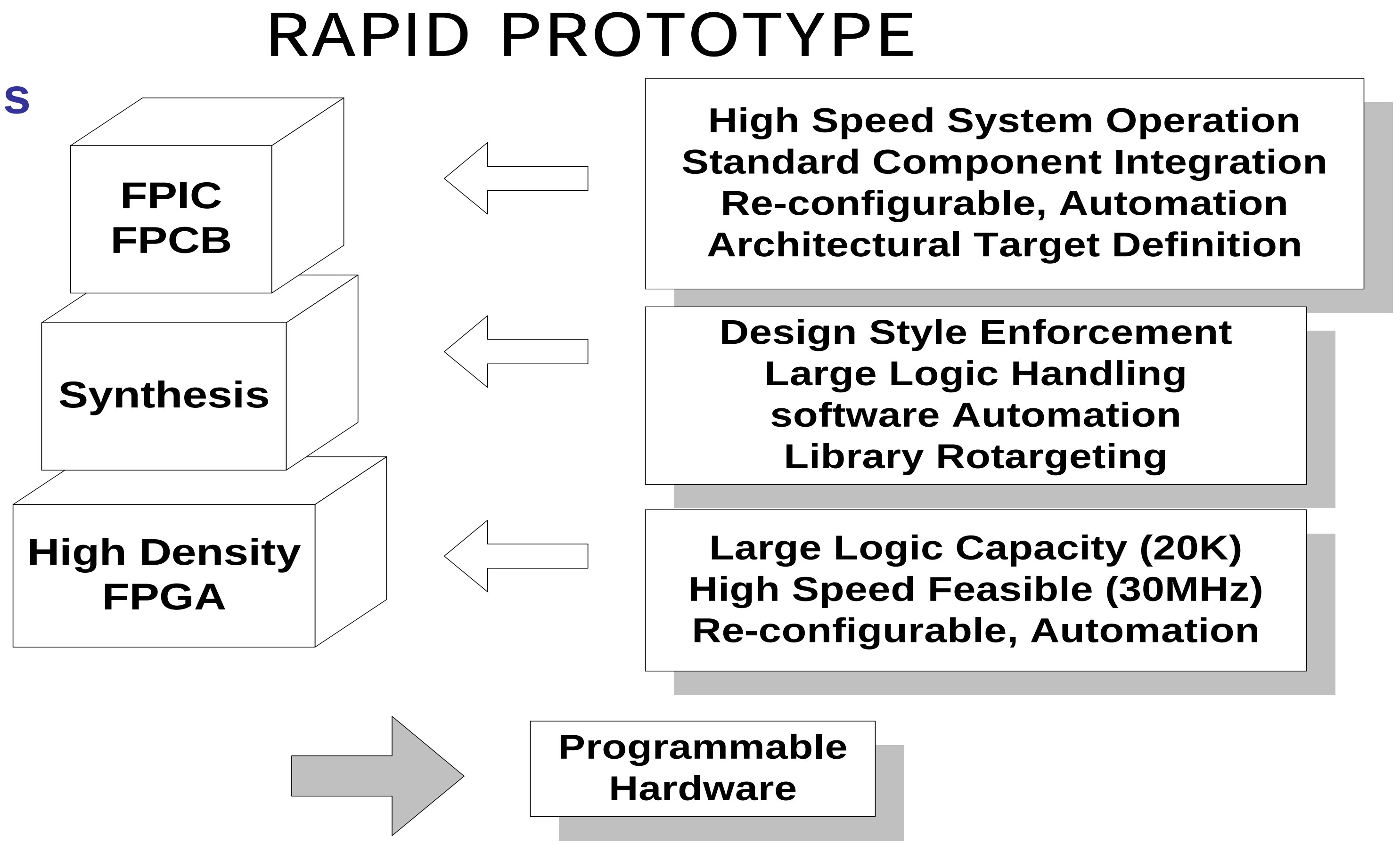
- Expandable system supports up to 11 independent Logic Emulation Modules (30,000 to 330,000 logic gates)
- System support up to 32 Memory Emulation Modules (Total 64 Mbytes, including multiport RAMs)
- Precision Emulation Software accurately synthesizes even the most complex design
- EDA system integration tools smooth the transition to emulation
- Fast, incremental design change capability shortens design iteration cycles

Enterprise Emulation System


Gate-level Netlist Files
Libraries
Netlist Reader and Parser
Technology Mapper
System-Level Partitioner and Placer
System-Level Interconnect Router
FPGA Compiler
FPGA Compiler
FPGA Compiler
Interconnect Compiler
Configuration Files for Each Logic and Interconnect Chip
Design Database
Timing analysis (Optional)

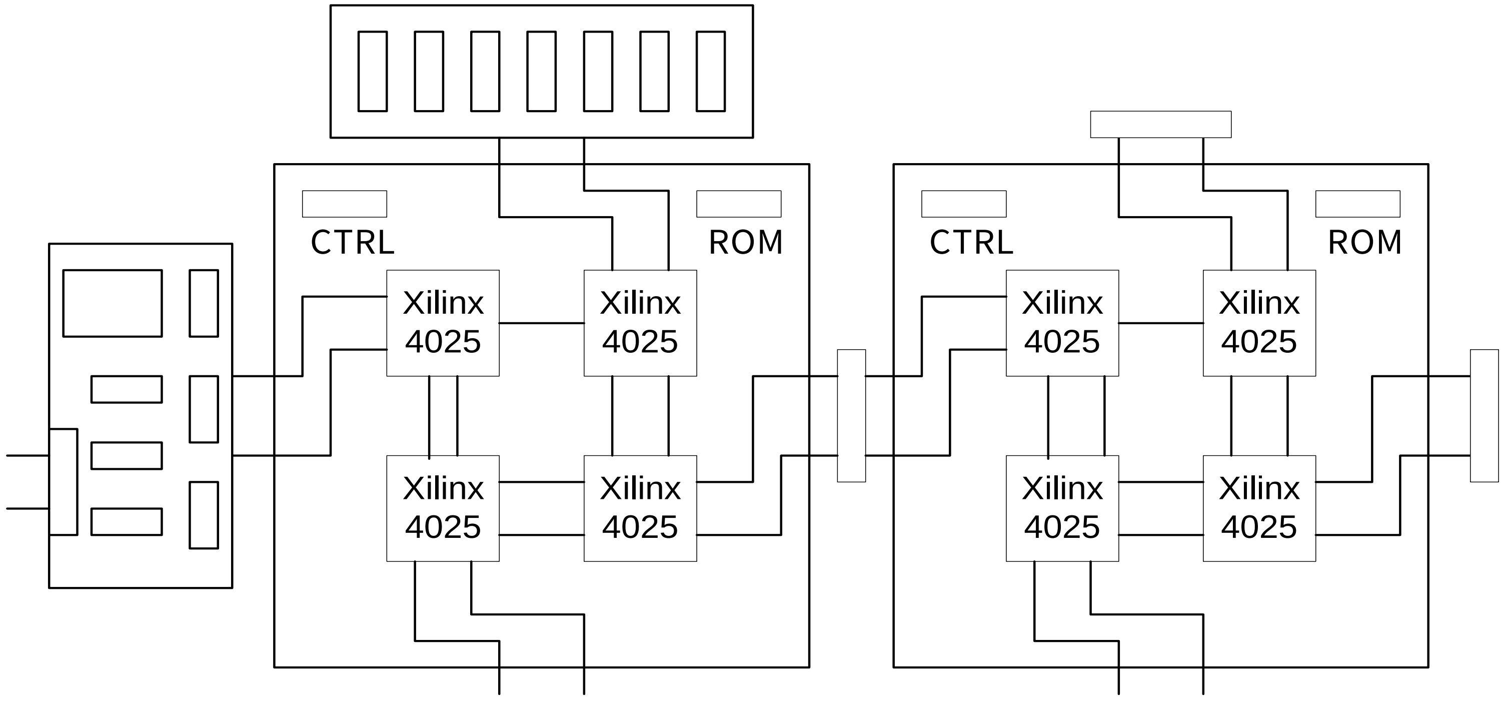
Aptix

- Rapid prototyping is made possible by
  - FPIC(Field Programmable Inter Connect)
  - FPCB(Field Programmable PCB)
  - FPGA
  - Synthesis


RAPID PROTOTYPE
FPIC
FPCB
Synthesis
High Density
FPGA
High Speed System Operation
Standard Component Integration
Re-configurable, Automation
Architectural Target Definition
Design Style Enforcement
Large Logic Handling
software Automation
Library Rotargeting
Large Logic Capacity (20K)
High Speed Feasible (30MHz)
Re-configurable, Automation
Programmable
Hardware

COBRA Prototyping Environment

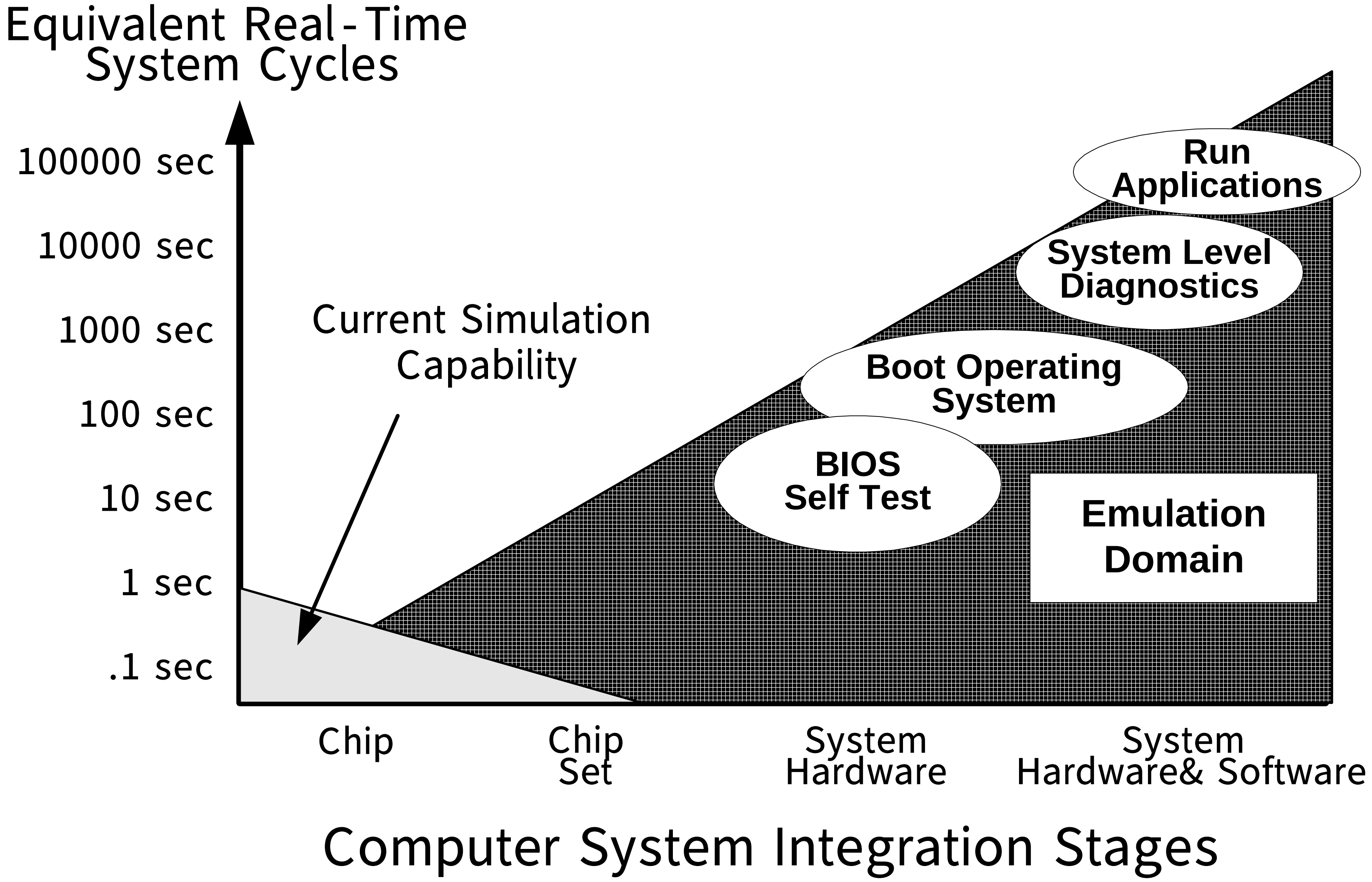
- **Connector : 90 pins/side, 75 bit link to two neighbors**
- **4 base modules (one with RAM module)**
- **Supporting software**
  - **Partitioning, synthesis, hardware debugging software**



Splash 2

- Based on Xilinx XC4010
- Scalable from 16 to 256 processing elements
- (1 ~ 16 boards)
- Higher I/O bandwidth (DMA from Sun SBus)
- Increased connectivity
- (linear data path + crossbar)
- Application programs in behavioral VHDL
- Array board :
  - 16 processing elements
  - 16 * 16 crossbar switch
  - 0.5 Mbyte memory
  - 36 bit bi-directional data paths

Case Study


Equivalent Real-Time System Cycles
100000 sec
10000 sec
1000 sec
100 sec
10 sec
1 sec
.1 sec
Current Simulation Capability
Run Applications
System Level Diagnostics
Boot Operating System
BIOS Self Test
Emulation Domain
Chip
Chip Set
System Hardware
System Hardware& Software
Computer System Integration Stages

UltraSPARC Emulation

- **Low-skew clock distribution**
  - **The vendor supplied clocktree analysis procedure**
  - **Gated clock are redesigned to either remove the gating or pull the gating to the root to allow use of low-skew nets**

- **Possible problem areas**
  - **Gated clocks (e.g. scan control)**
  - **Feed-thrus (designed to reflect the silicon floorplanning)**
    - **Only actual direct connections are important**
  - **Repeaters : Emulation does not need repeaters**
  - **Precharge logic & pass-thru latches must be redesigned into a static representation (careful verification required)**

UltraSPARC Emulation

- **Verification at the block/megacell level**
  - **All the reimplementation of megacells (due to internal clocking or memory) must be carefully verified for cycle accurate functionality**
- **Full chip configuration**
  - **takes 36 walk clock hours using 5 fast +70 computers**
  - **5 emulation systems form Quickturn**
  - **50 system to system cables**
- **Testbed**
  - **Over 5000 nets routinely probed (128k depth)**
  - **The internal & external logic analyzer samples are Integrated**
- **Return on emulation investment**
  - **Pre-tapeout : 25% (additional final design verification)**
  - **Post_tapeout to pre-silicon : 25% (stress testing)**
  - **Post-silicon : 50% (visibility for debugging, what-if experiments)**

Motorola 68060

- Methodology
  - Generating a gate level model
    - Handling custom logic and memory arrays
    - Vector generation and verification
  - Emulation library development
  - Emulation model build
  - Functional emulation
  - In-circuit emulation
- Time comparison (10^8 vectors)
  - Compiled RTL simulator : 6weeks
  - Functional emulation (1KHz) : 25hrs
  - In-circuit emulation (1MHz) : 90secs
  - 68060 silicon (50MHz) : 1sec

PowerPC

- **Demands :**
  - **PowerPC 603/604^{TM} bus speed :**
    - **2~4 times the speed of the external bus.**
  - **Pipeline controller to run at 75MHz**

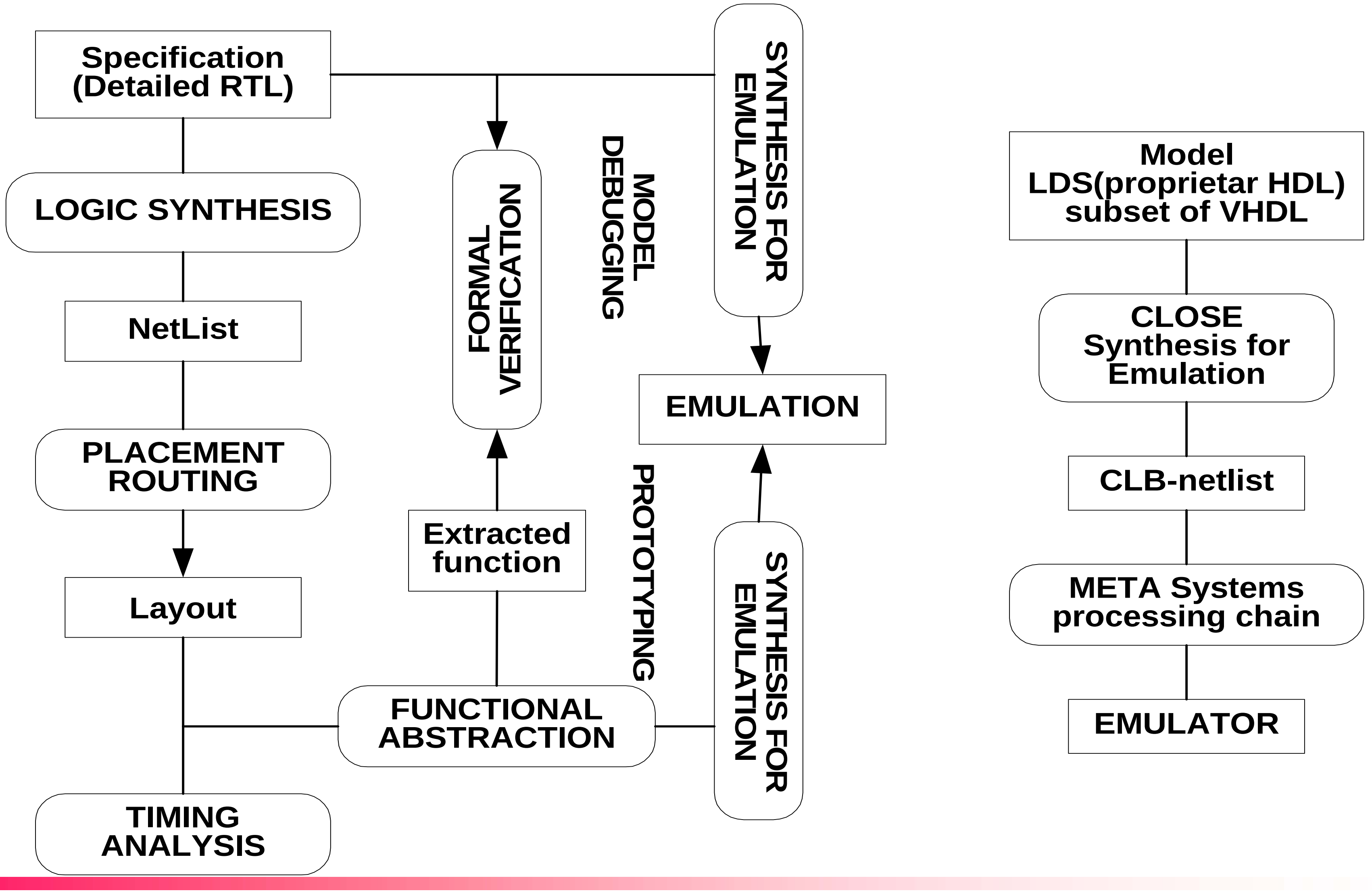
- **CPLD is a better choice than FPGA**
  - **CPLD provides a higher number of pterms in each cell**
  - **The state bits are visible**
  - **CPLDs boast a very predictable delay path**

Design for Emulation

- Synchronous design
- Pipeline design techniques
- Short arithmetic functions (minimize logic level)
- Minimize bit width
- Use of I/O FFs where possible (latches)
- Careful mapping between functions and FPGAs
- Minimize high fanout net or use available buses
- Use of small blocks (20-40K gates)
- Care when gating the clock tree (e.g. low power)
- Limit module I/O count

Synthesis to Support Emulation

- Synthesis to compile behavioral models for emulation



Application of Synthesis

- Efficient models to handle specific logic styles
  - Multiple clocks
  - Multiple input latches
  - Tristate and precharged signals
- Efficient logic optimization
- Module generators for specific operators
- Control of visibility and controllability

Conclusion : Ideal Emulation

- Cheaper : $1.25/gate in 1994
- Easier to use (reduce time-to-emulation)
  - Automatic design transformation (synchronous design: setup/hold time, gated clock, wired logic, precharge logic)
  - Compiler (partitioning and mapping)
  - Hierarchical modular architecture
  - Target interface for complex multi-chip system
- Powerful debug environment
  - Debugging environment or short debug turn-around time
  - Modular complication for incremental change
  - Powerful logic analysis
- Scalability to handle increasing complexity
  - Modular and flexible packaging
- Current Technology
  - 20-30M emulation gates, over 5MHz emulation speed