Power Matters

Microsemi

FPGA Technology & FPGA Design

Nizar Abdallah, Ph.D.

Workshop on FPGA Design for Scientific Instrumentation and Computing
International Centre for Theoretical Physics
November 2013

A Short Bio…

INFOTRON

ARCONIX

UPMC SORBONNE UNIVERSITÉS

ICTP The Abdus Salam International Centre for Theoretical Physics

Actel POWER MATTERS

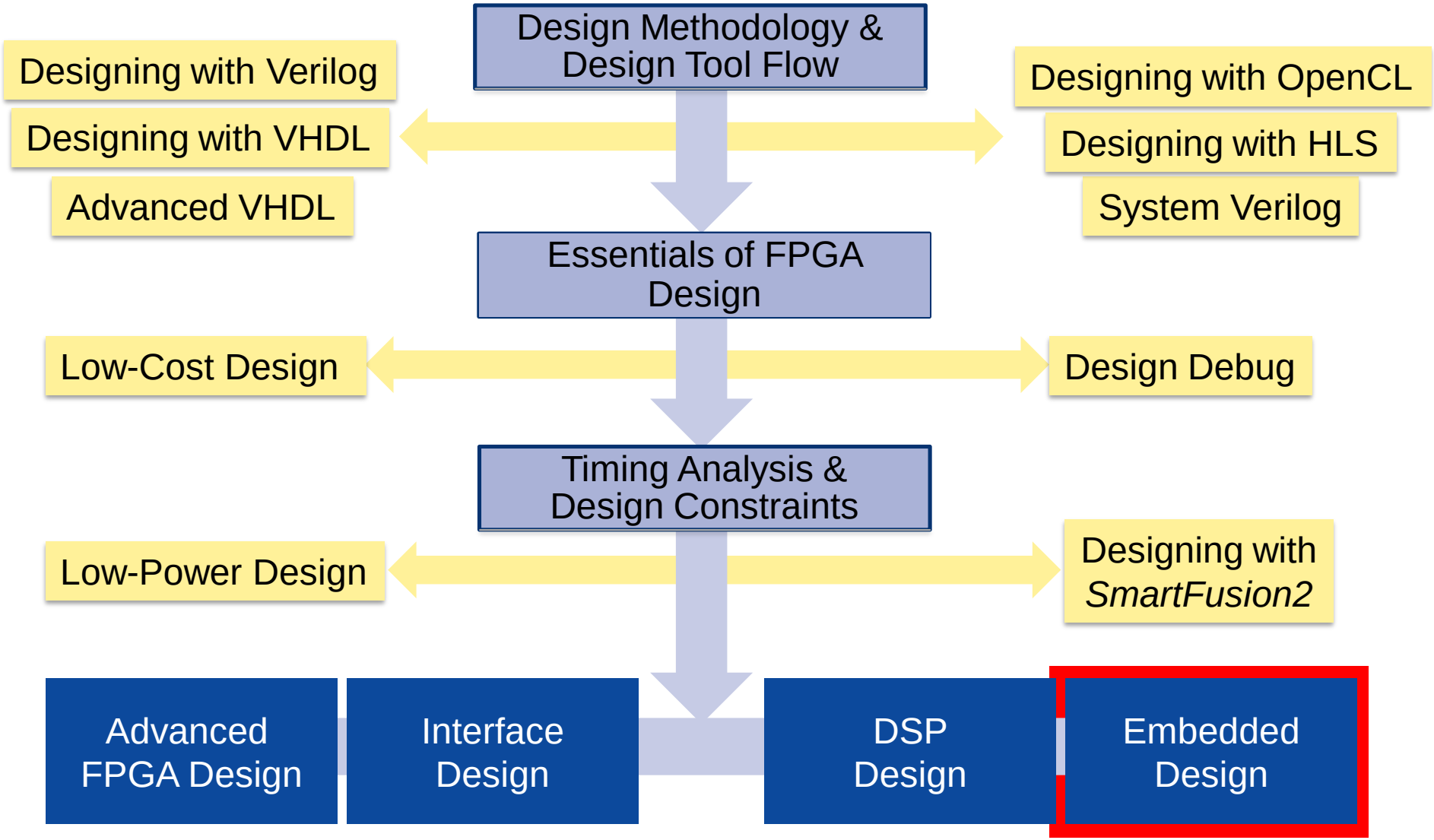
Microsemi SoC Products Group

PRESIDIO GRADUATE SCHOOL

Outline

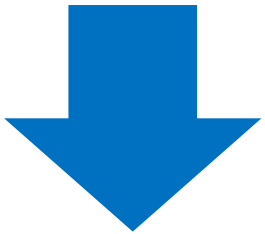
- Introduction
- What is a Field Programmable Gate Array (FPGA)?
- Essentials of FPGA Technology and FPGA Design

Modern FPGA Design Curriculum



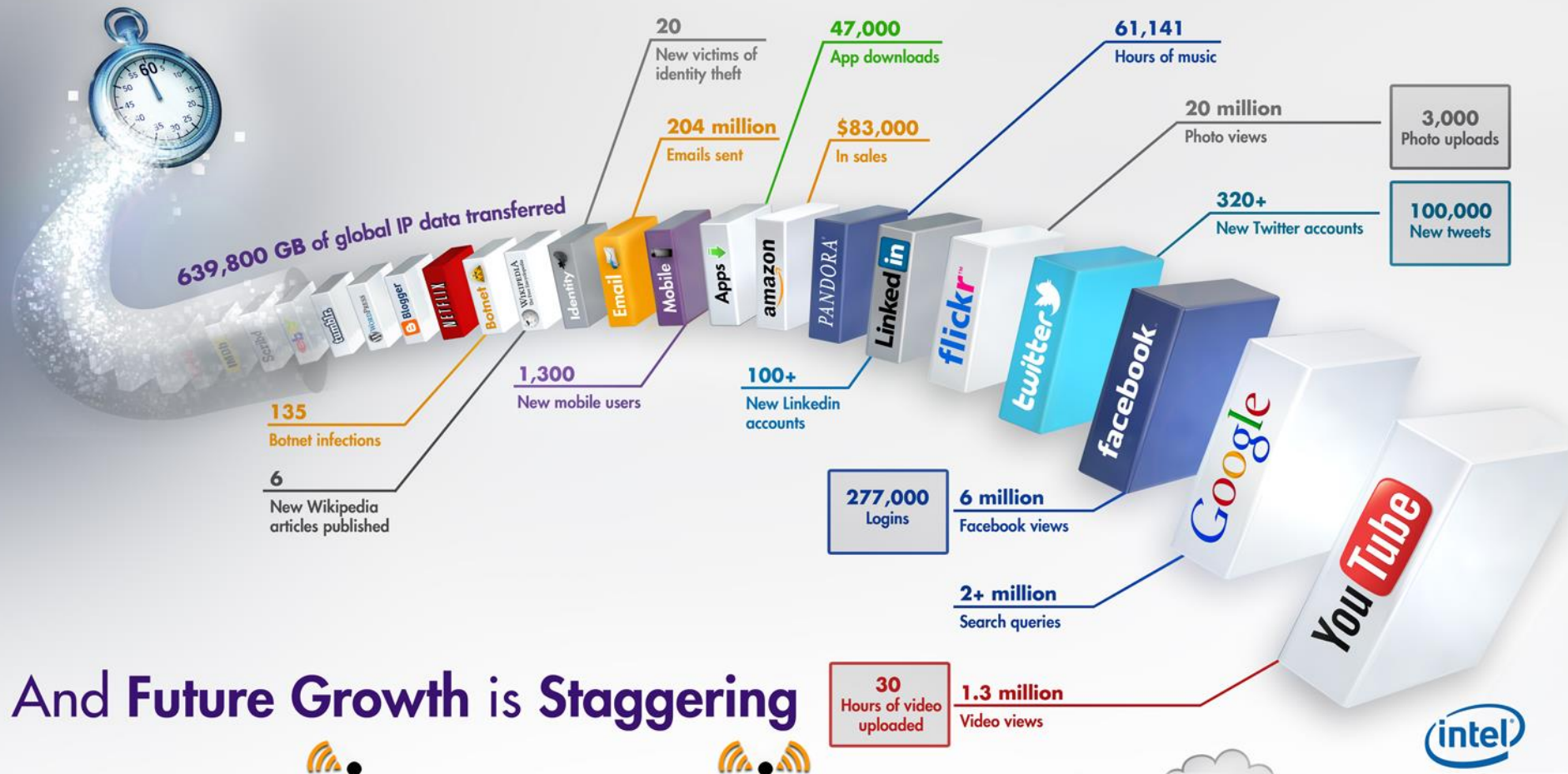
Today…

FPGAs & Processors
are meeting



the era of
Programmable SoC

What Happens in an **Internet Minute?**



And **Future Growth** is **Staggering**

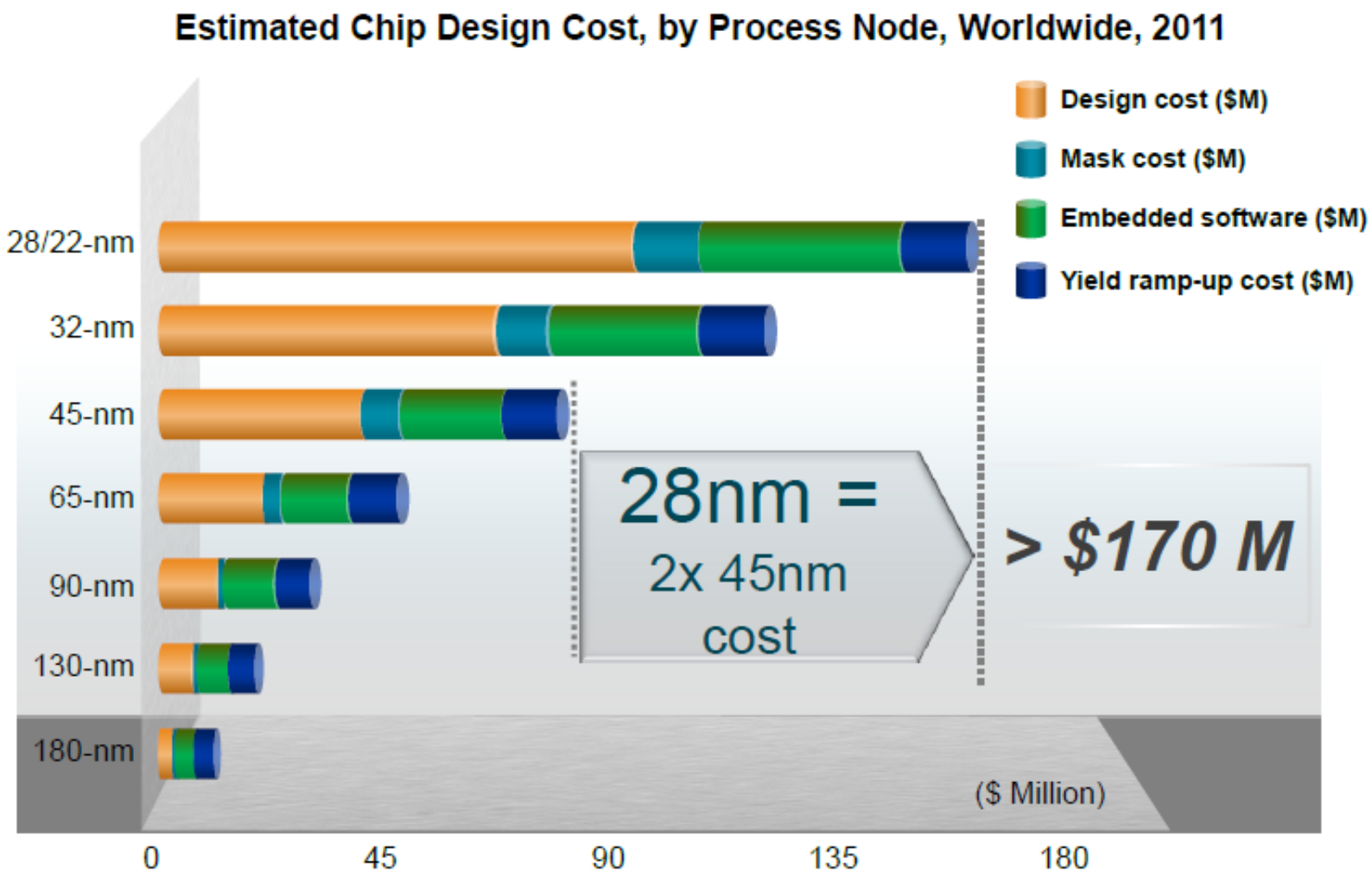
Today, the number of networked devices = the global population

By **2015**, the number of networked devices = **2x** the global population

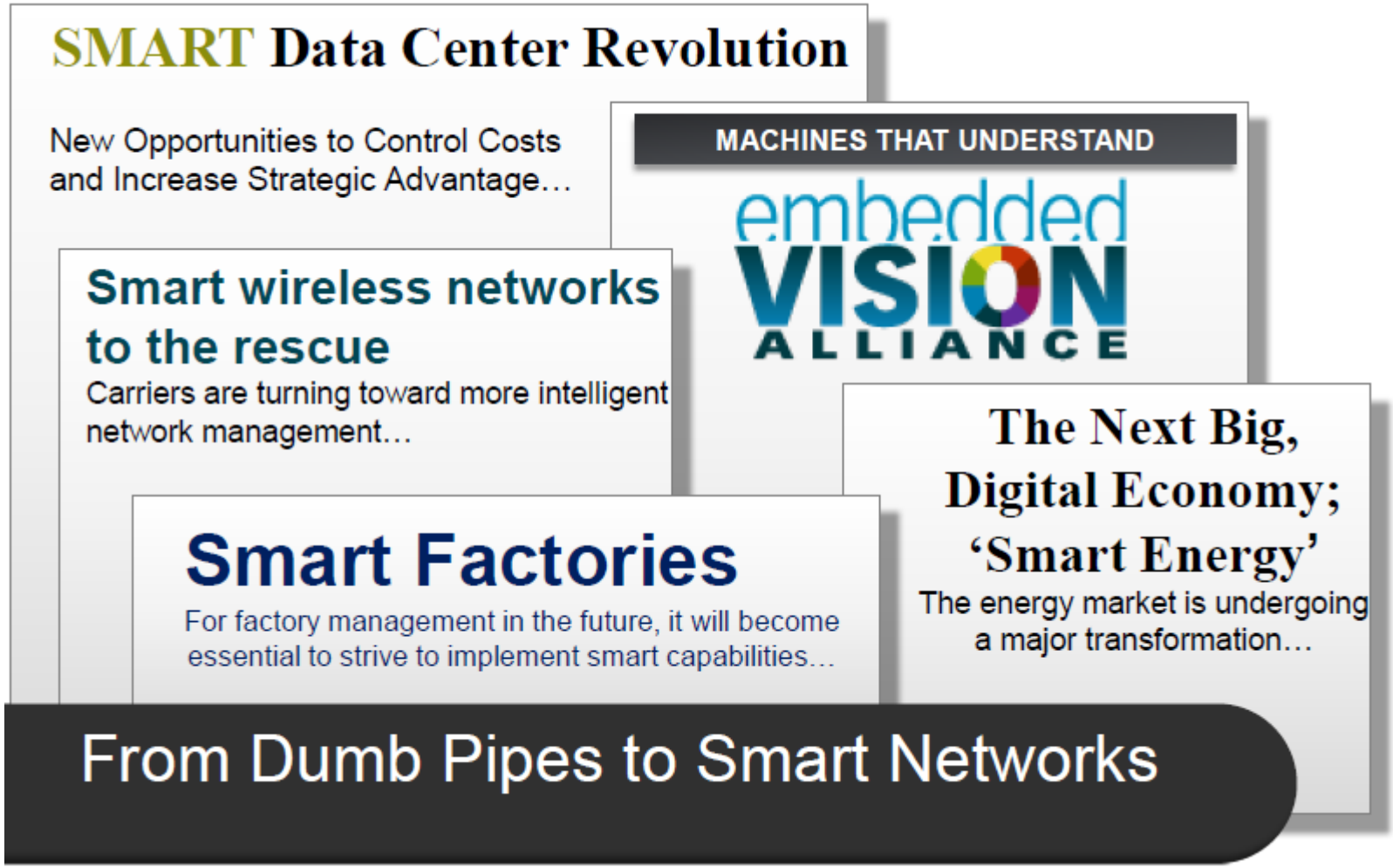
In **2015**, it would take you **5 years** to view all video crossing IP networks each **second**

intel

Design Cost


Estimated Chip Design Cost, by Process Node, Worldwide, 2011
Design cost ($M)
Mask cost ($M)
Embedded software ($M)
Yield ramp-up cost ($M)
28/22-nm
32-nm
45-nm
65-nm
90-nm
130-nm
180-nm
28nm =
2x 45nm
cost
> $170 M
($ Million)
0
45
90
135
180

More Intelligence in Every System



Trend Data Center Infrastructure: Cloud Computing

Big Data

Increasing Volume, Velocity, and Variety

Low power

Reduce operation and cooling costs

Security

Both outside and inside

Industry Mandates

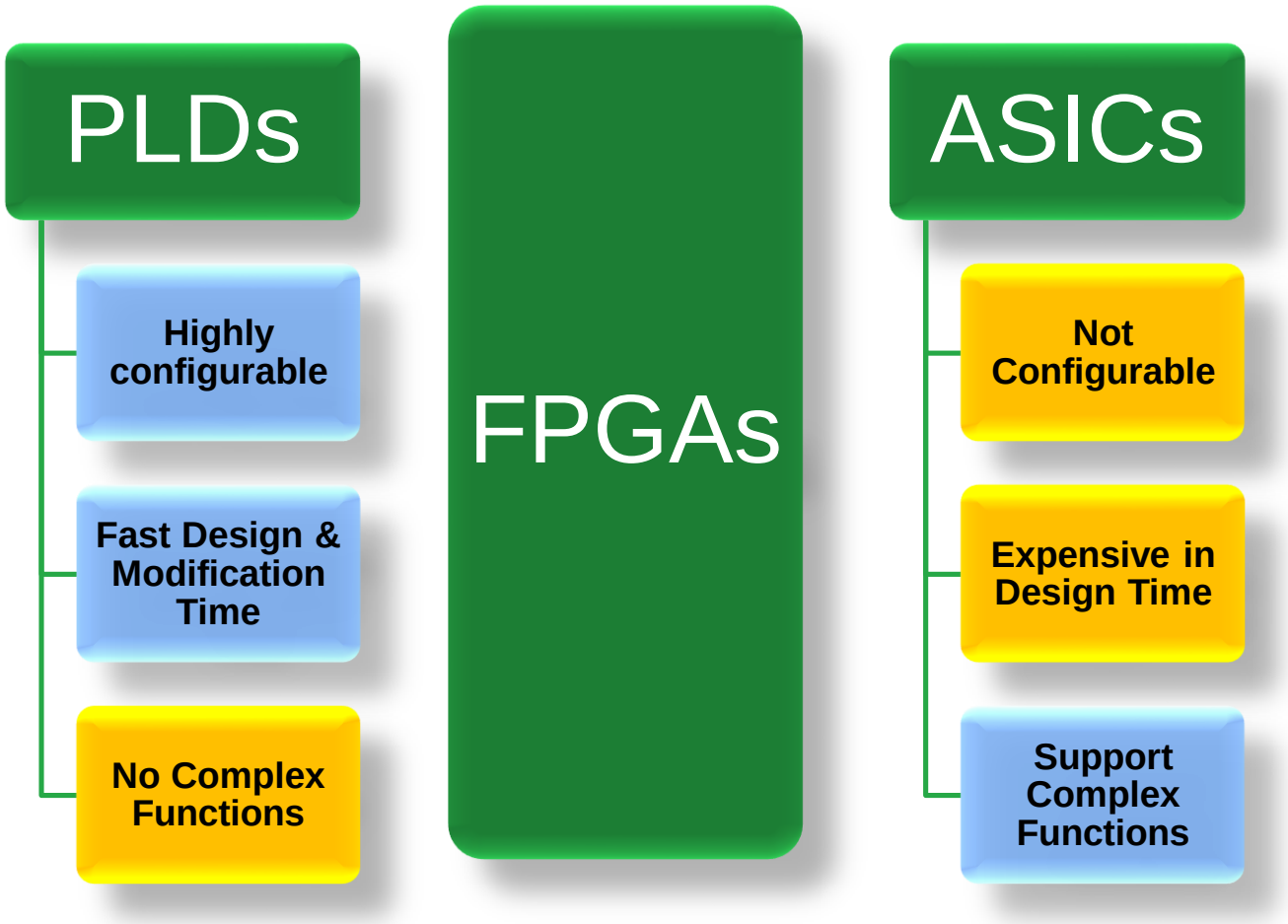
Programmable Imperative

FPGA Technology Overview

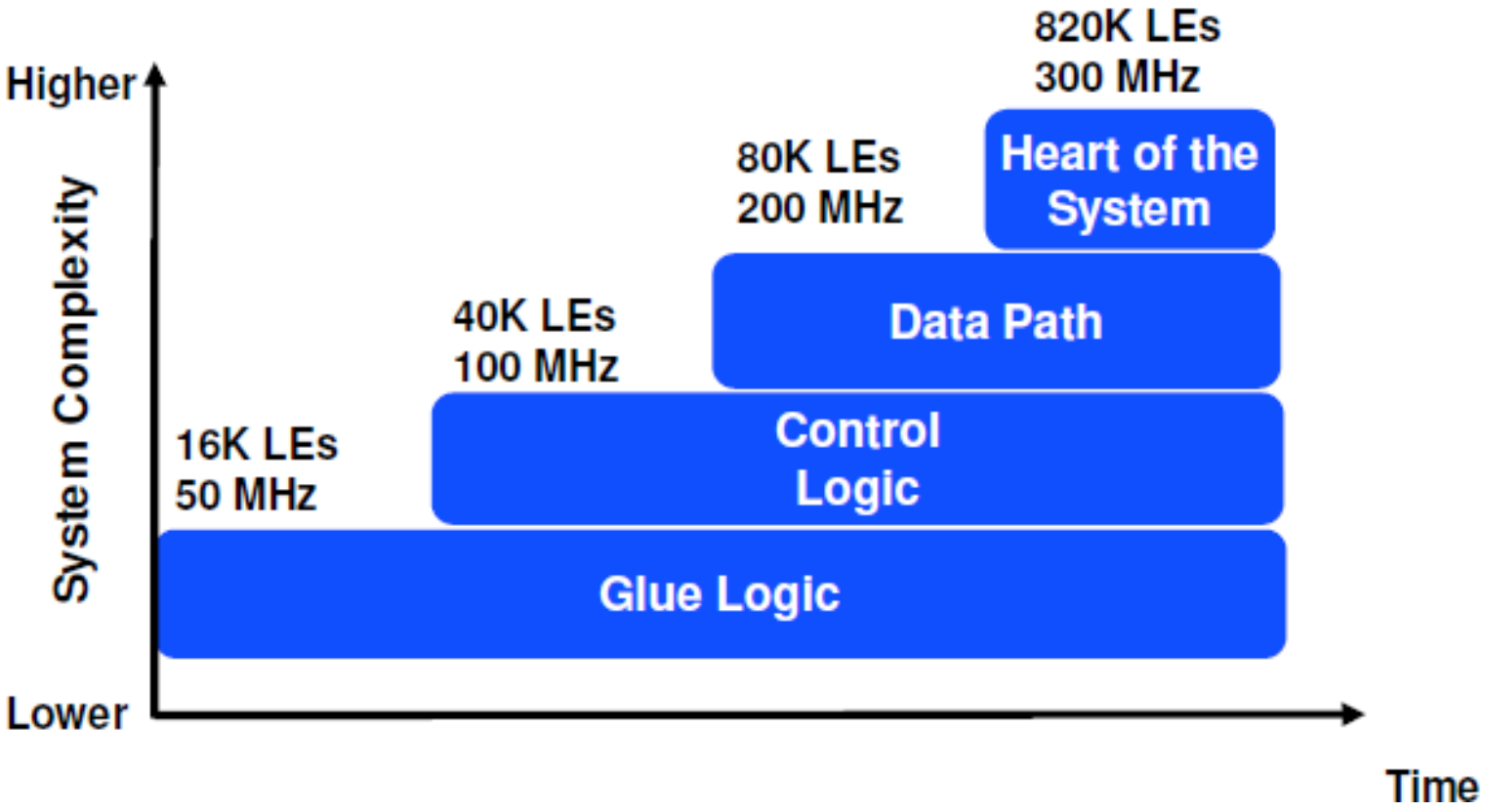
What's a FPGA?

Field PROGRAMMABLE Gate Array

The GAP


PLDs
Highly configurable
Fast Design & Modification Time
No Complex Functions
FPGAs
ASICs
Not Configurable
Expensive in Design Time
Support Complex Functions

Where Do FPGAs Fit?



What's a FPGA?

A simplistic old definition:

- a high capacity programmable logic device
- An array of programmable basic logic cells surrounded by programmable interconnects
- Can be configured (programmed) by end-users (field-programmable) to implement specific applications
- Capacity up to multi-millions logic gates and up to 500MHz core clock speed, supporting giga-sample per second data throughput rates
- Popular applications: prototyping, on-site hardware reconfiguration, DSP, logic emulation, network components, etc…

FPGA Definition

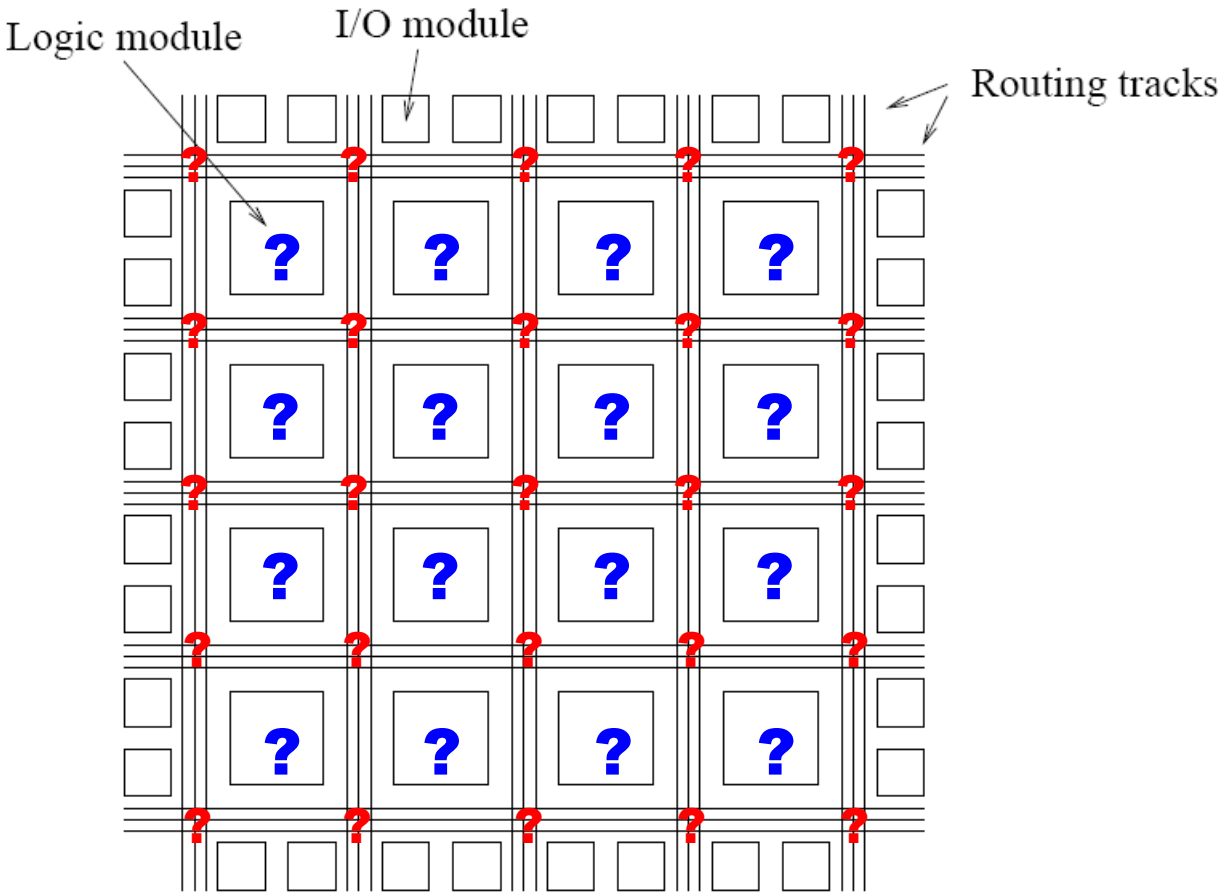
Field Programmable Gate Array

A large number of logic gates in an IC array that can be connected (configured) electrically

The Four Components of an old FPGA

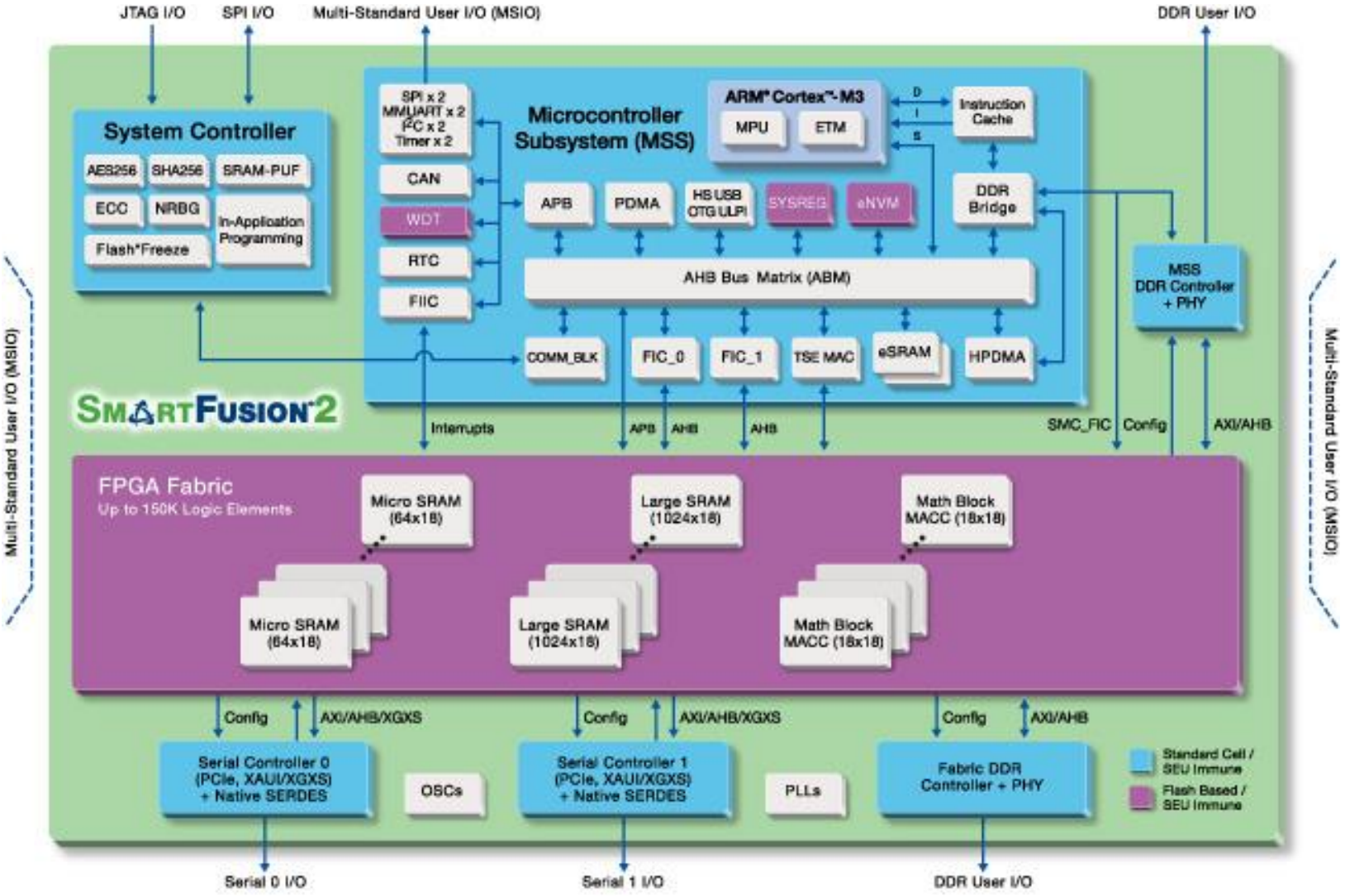
- The Configuration Element
- The Logic Module
- The Memory
- Control Circuits/Special Features

A Simplistic Old Architecture



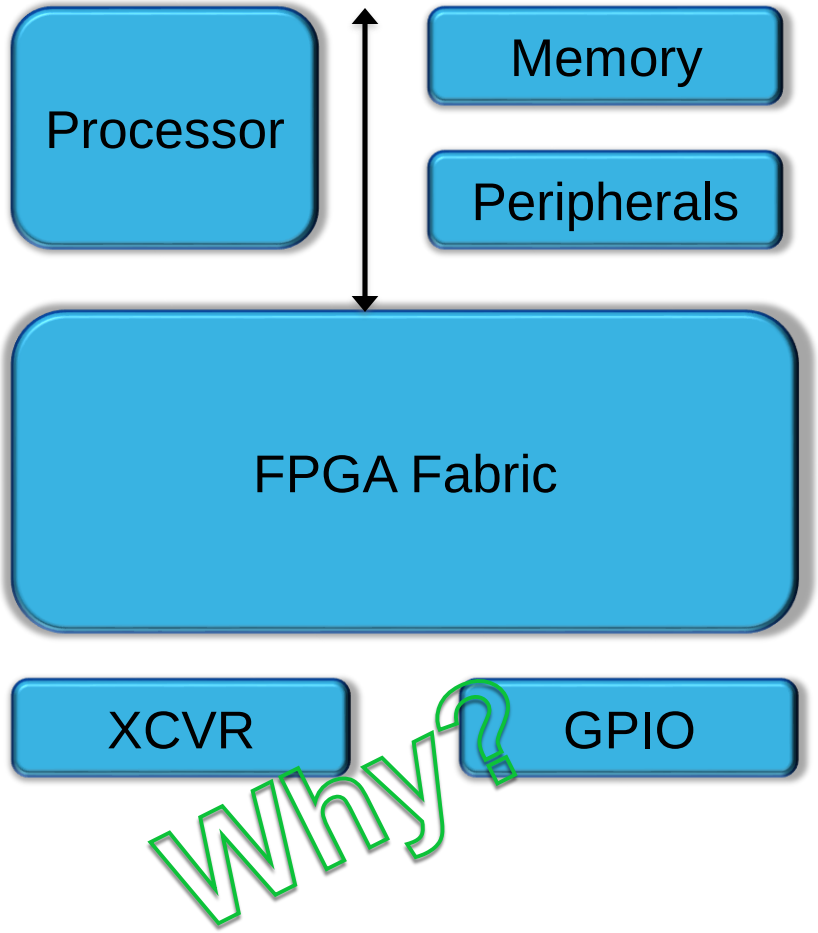
Generic FPGA Architecture

A More Modern Architecture



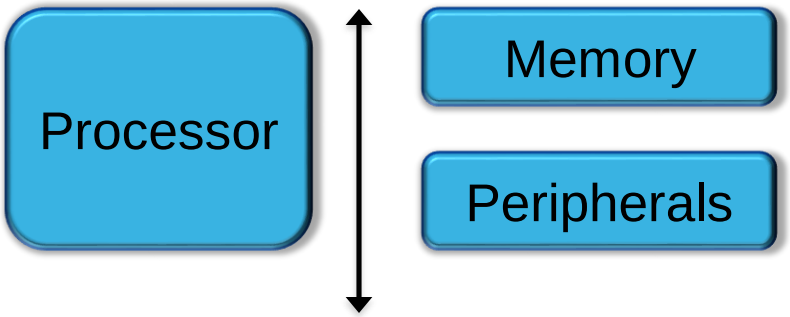
FPGAs with Embedded Processors

- As you would expect…
- Processor
  - With associated memory and peripherals
- FPGA Fabric
  - Logic blocks, Memory blocks, Math blocks, etc.
- Transceiver and GPIO



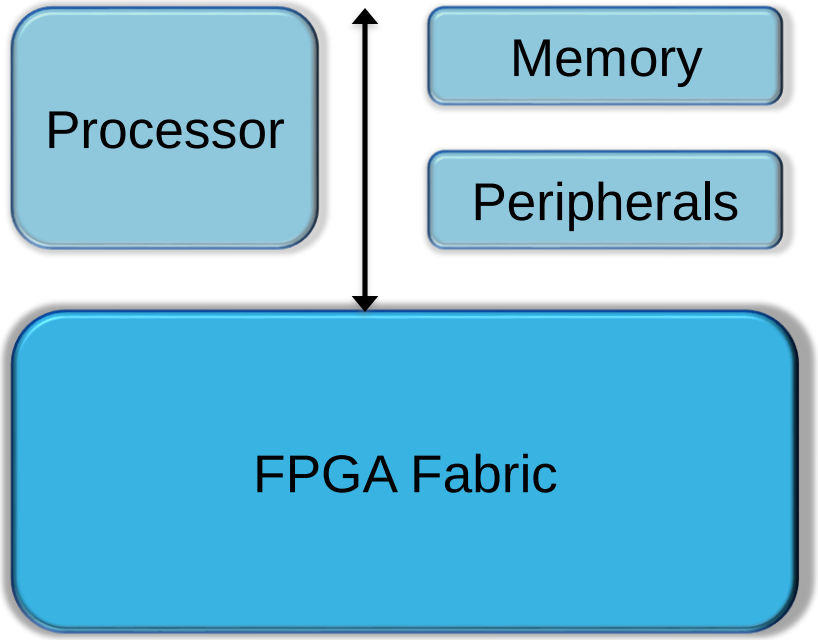
Processor, Memory & Peripherals

- Processor
  - ARM: Single/Dual, FPU
  - Interrupt, Debug, Cache
  - Bus Interface
  - Eco-system
- Memory
  - Instruction and Data
  - Usually SRAM (Flash from Microsemi)
  - Cache (L2)
- Peripherals
  - Internal: Timers, WDT, DMA, Security, etc
  - External: UART, SPI, I2C, CAN, SDIO, USB, ENET, Flash Controller, DDR Controller, ADC, DAC



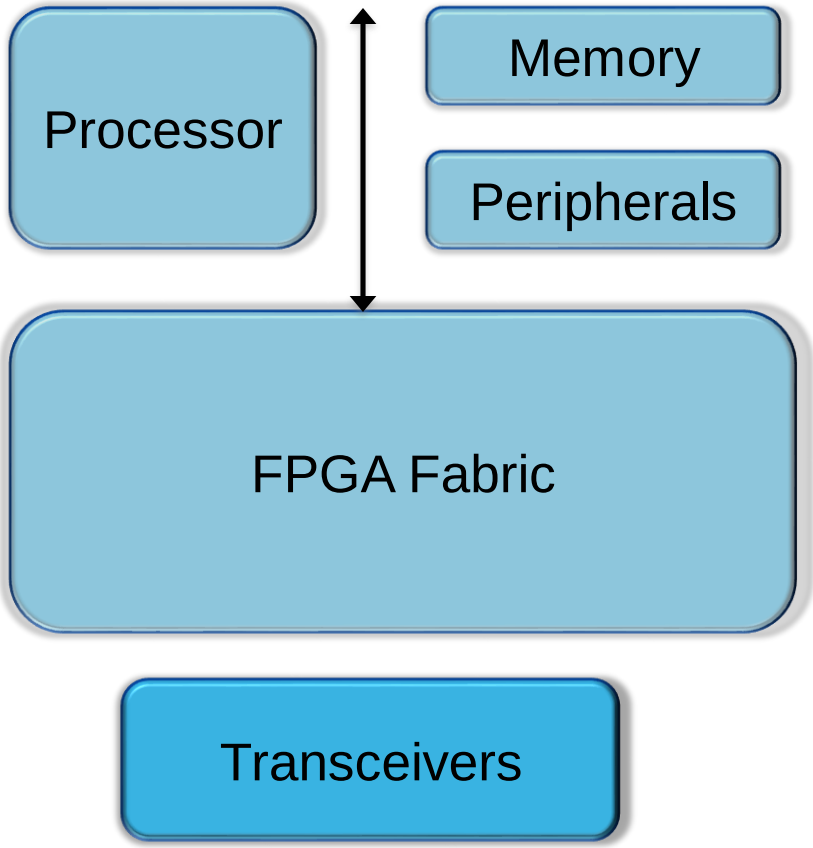
FPGA Fabric

- Look-up Tables
- Interconnect
- Carry logic for counters
- Block memory
  - Large, Small, Multi-port
- Math blocks
  - DSP, Fixed/Floating point
- Interface to Processor Subsystem



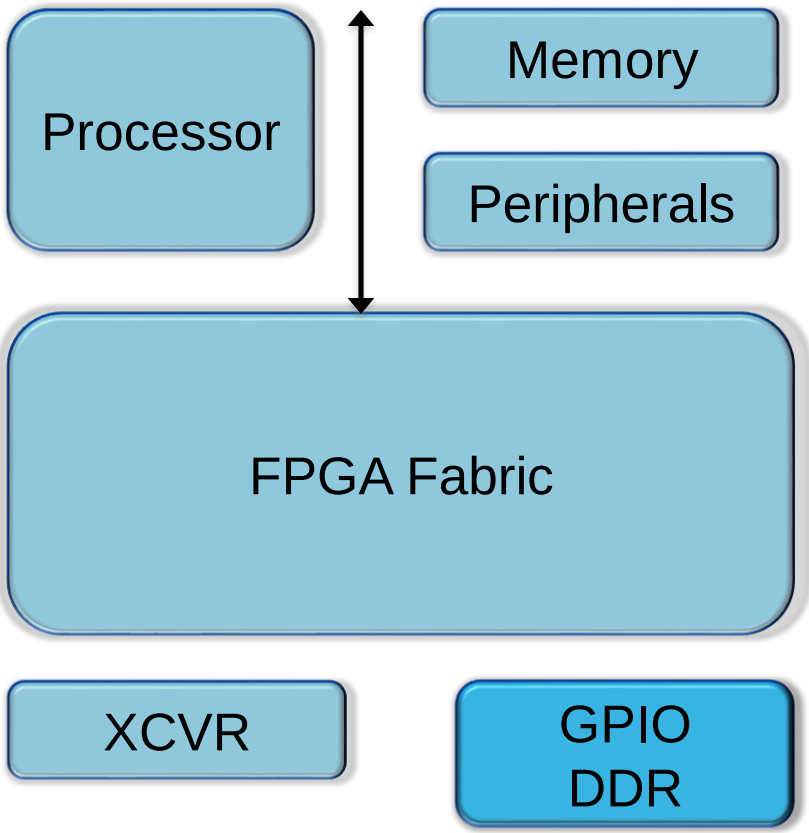
Transceivers

- aka, Serializer Deserializer
  - Low Level logic needed for high speed serial IOs
  - Programmable PHY
    - Advanced features
      - 6-10Gbps
      - Adaptive Equalization, Pre-Distortion, etc
    - Testing: On-chip Eye
  - Some also have hard MACs
    - PCIe



GPIO, DDR Controllers

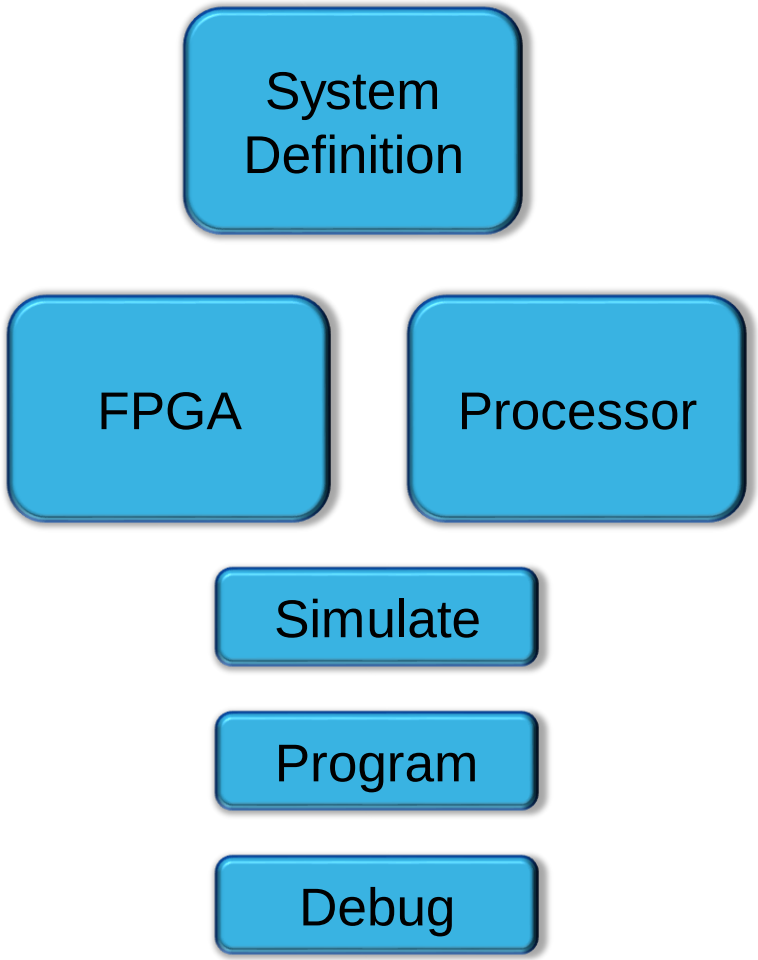
- GPIO
  - Support for many, many IO standards
  - Bank basis
  - Programmable features
- DDR Controllers
  - Range of standards
    - LPDDR, DDR2/3, etc
  - Advanced features
    - PHY, Access Optimization, ECC, etc



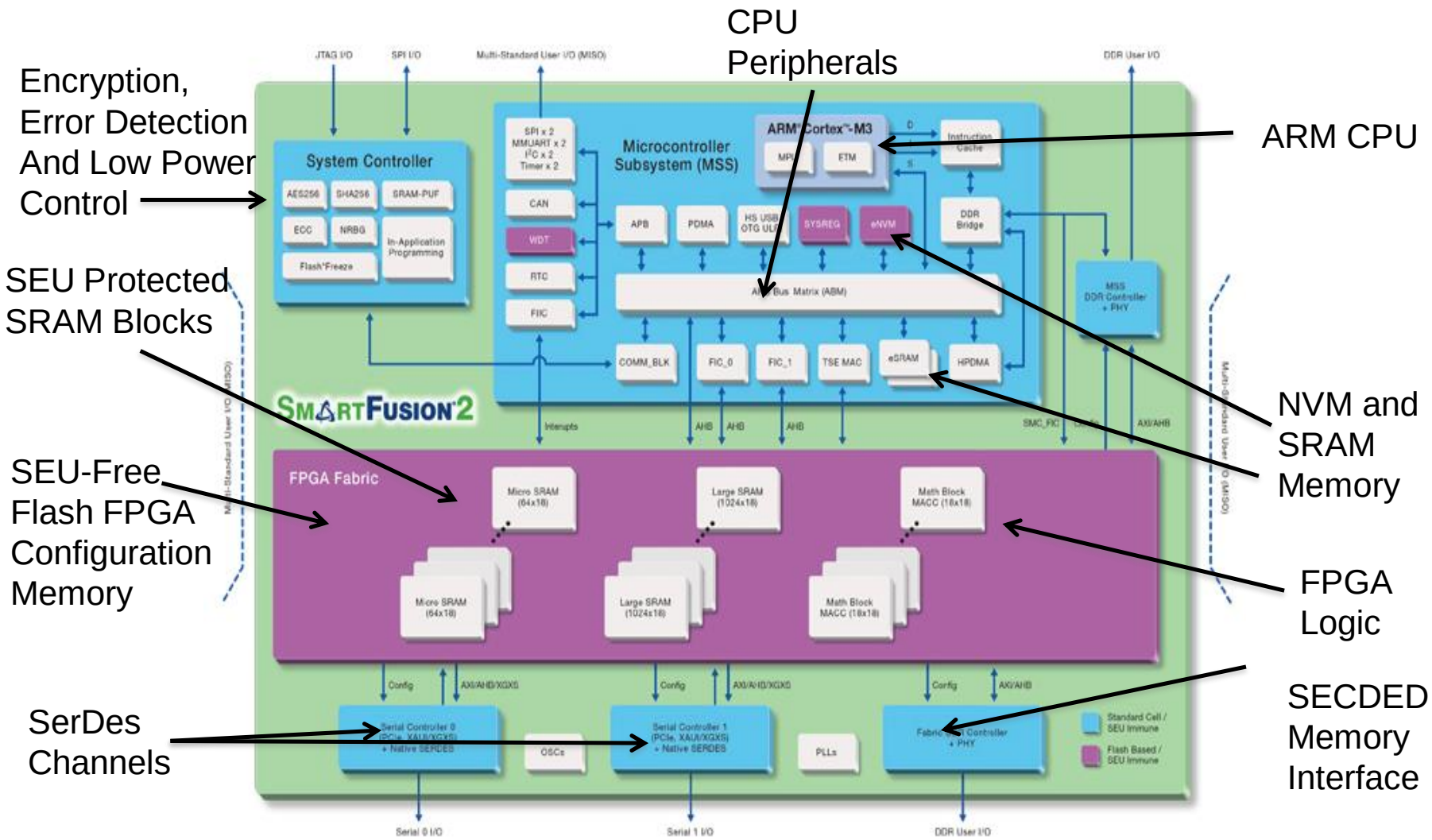
Software

- Define the system
- Program the Processor in “C” or Assembly
  - Libraries
- Program the FPGA in HDL
  - IP Blocks
- Simulate, Program and Debug

- Don’t forget to look at the software tools offered by the FPGA provider



Microsemi SmartFusion 2



Major FPGA Vendors

SRAM-based FPGAs

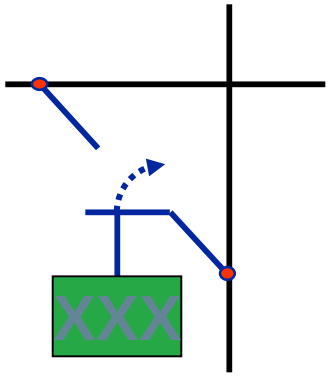
- Xilinx
- Intel (form. Altera)
- Lattice Semiconductor

Flash & antifuse FPGAs

- Microsemi (form. Actel)

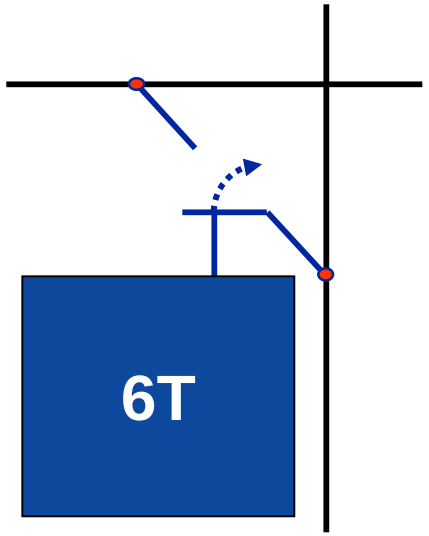
FPGA Routing Technologies

- The Interconnect Switch



FPGA Routing Technologies

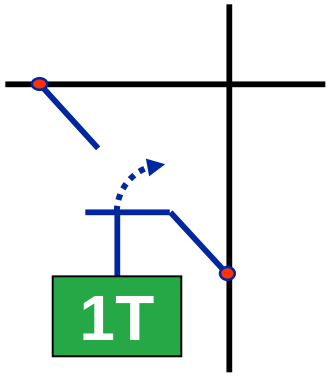
SRAM



Reprogrammable

Large Switch
expensive wires
Low Logic Utilization
typ 60%

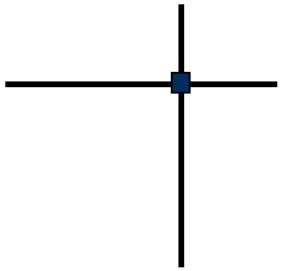
Flash



Best of Both Worlds
Reprogrammable
& Nonvolatile

Small Switch
cheap wires
High Logic Utilization
typ >85%

Anti-fuse



Nonvolatile

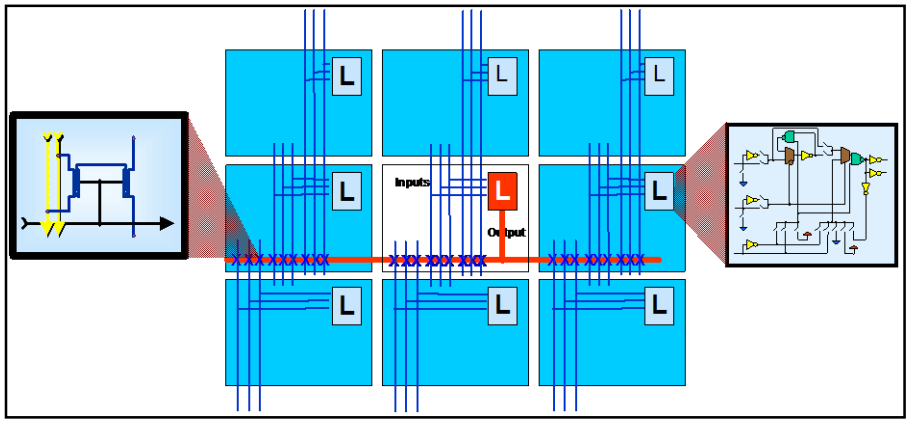
Smallest Switch
cheapest wires Highest
Logic Utilization
typ >90%

FPGA Routing Resources

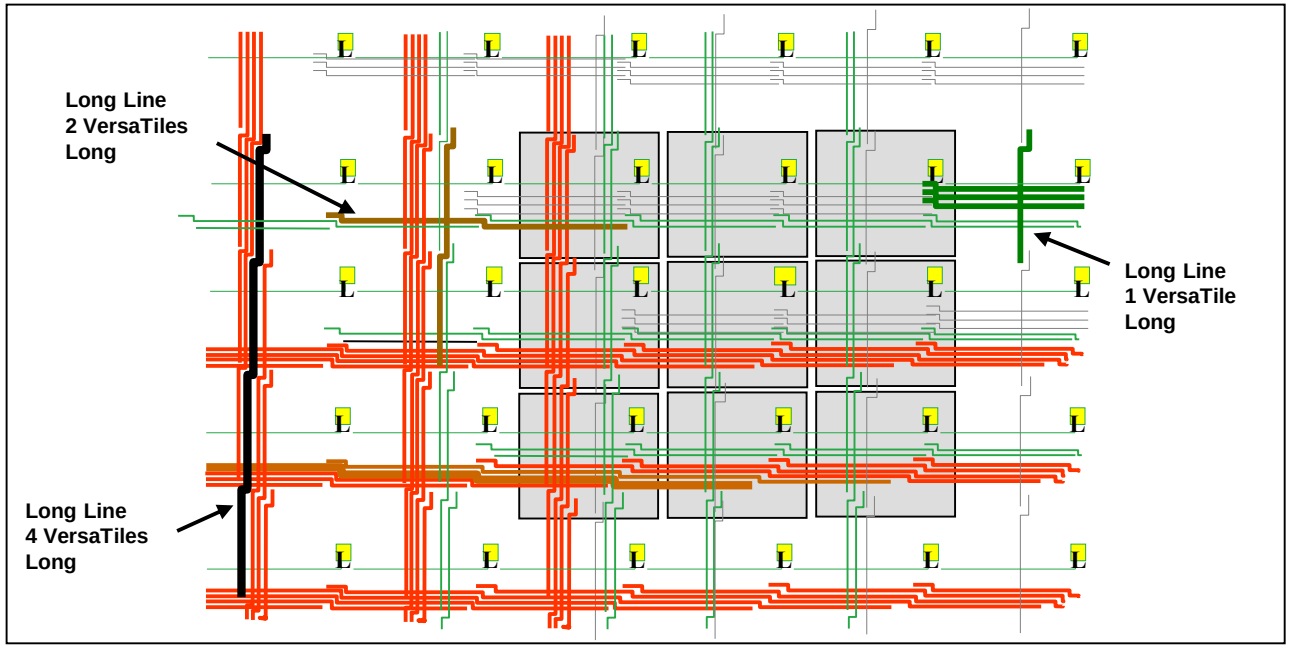
- Flexible High-Performance Routing Hierarchy
  - Ultra-fast Local Network
  - Efficient Discrete Long-line Network (1, 2 and 4 Tiles Long)
  - High-speed Very-long-line Network
  - Eighteen Low-skew Global Networks

- Benefits
  - Multiple Routing Path Alternatives for Low Congestion
  - Short Corner-to-corner Delays
  - Enables Rapid Timing Convergence

FPGA Local and Long Line Networks

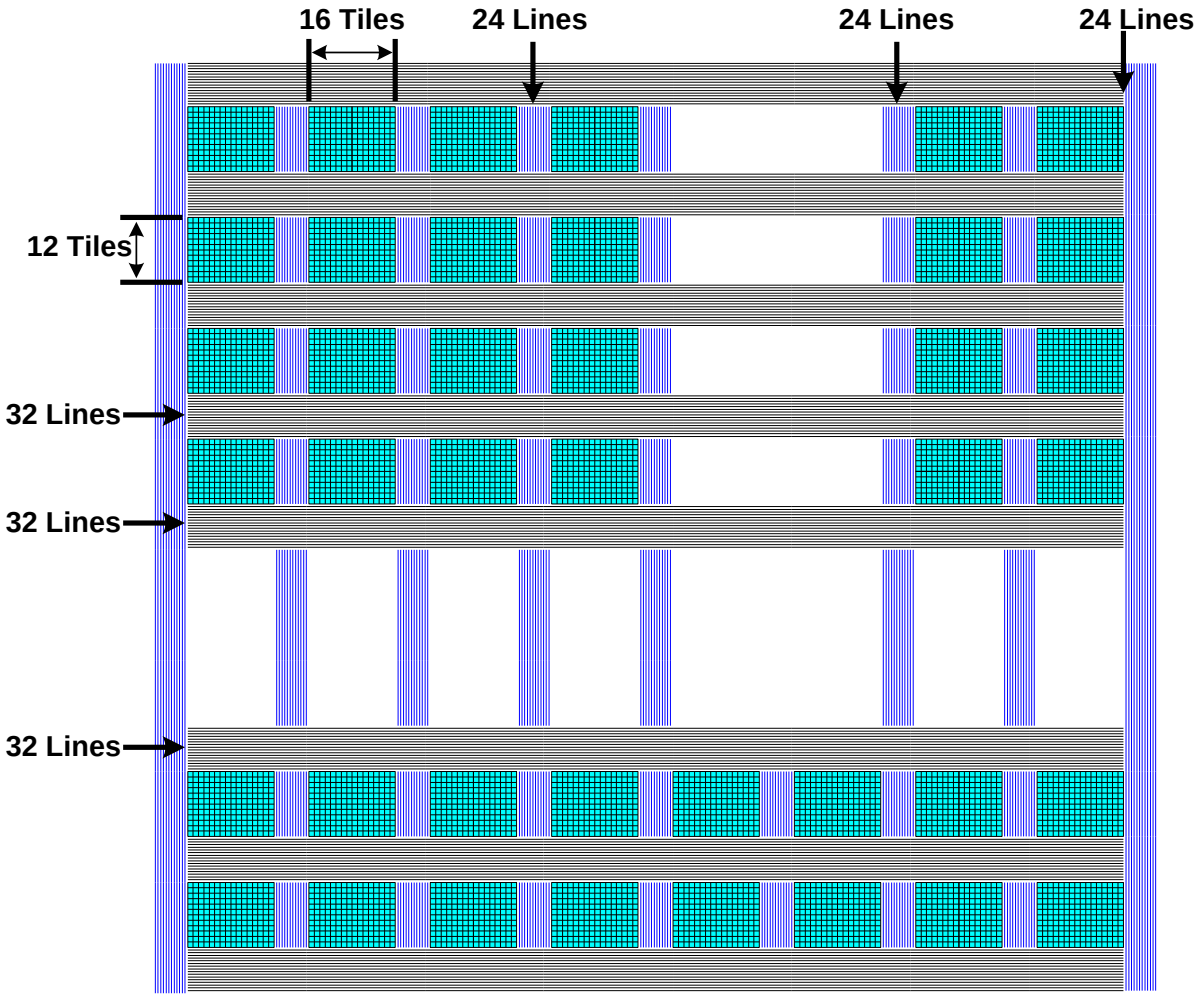


Local Lines connect VersaTile Outputs to nearest-neighbor VersaTiles, I/O Buffer, or Memory Block



Long Lines route longer distances and support higher fanout nets

FPGA Very Long Line Network



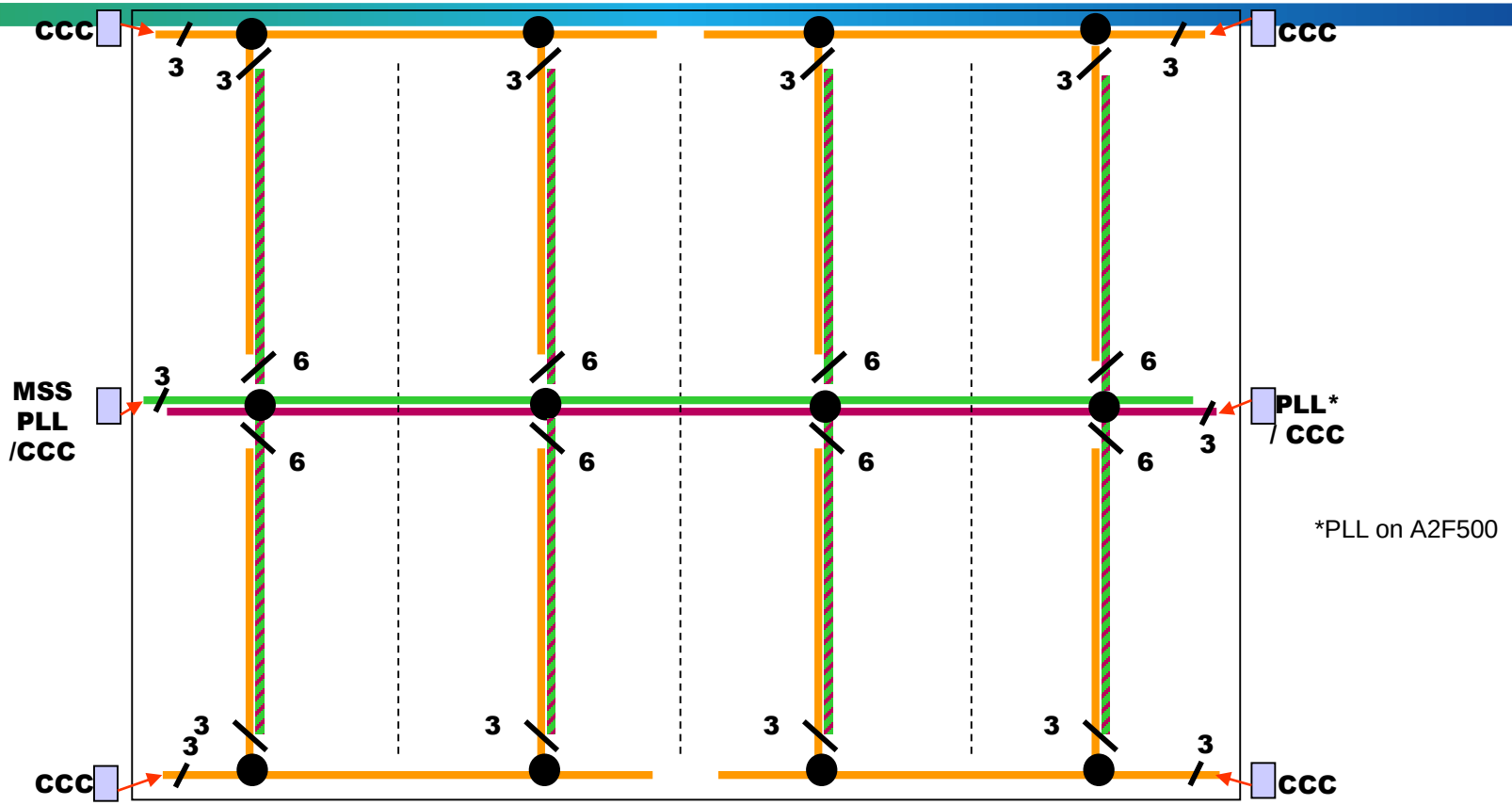
Very Long Lines reach ±12 Tiles vertically and ±16 Tiles horizontally (SmartFusion)

FPGA Global Networks

- FPGA fabric contains multiple Global Resources
  - Chip-wide Global Networks and Quadrant Global Networks
  - Chip-wide Global Networks
  - Can reach all Tiles (Ports, RAM, I/O, and CCC Tiles)
  - Driven by Clock Conditioning Circuitry (CCC)

- Quadrant Global Networks
  - Can Reach All Tiles Within the Quadrant
  - Driven by Clock Conditioning Circuitry (CCC), usually in all Corners of the Die

FPGA Global Network (SmartFusion)



- Left and Right CCCs Provide Access to 6 Chip-wide Global Networks
- CCCs in 4 Corners Provide Access to 12 Quadrant Global Networks (3 per Quadrant)
- Each Tile Has Access to 9 Global Resources

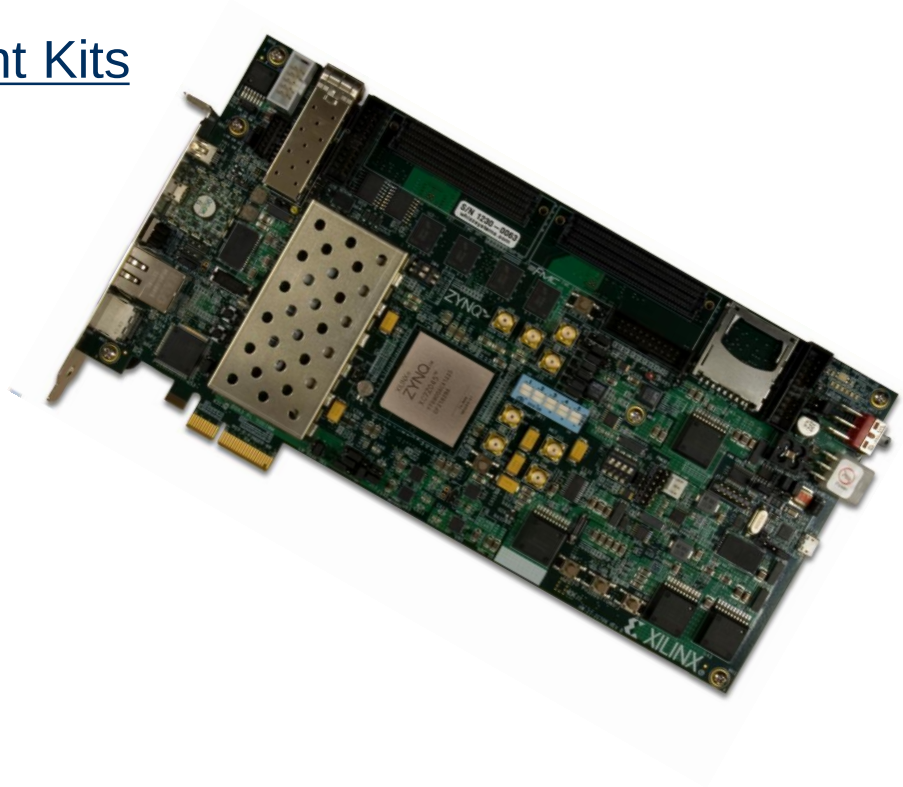
Additional Resources

Altera Arria Web Page
Altera Arria Development Kits

Microsemi SmartFusion2 Web Page
Microsemi SmartFusion2 Development Kits

Xilinx Zynq Web Page
Xilinx Zynq Development Kits

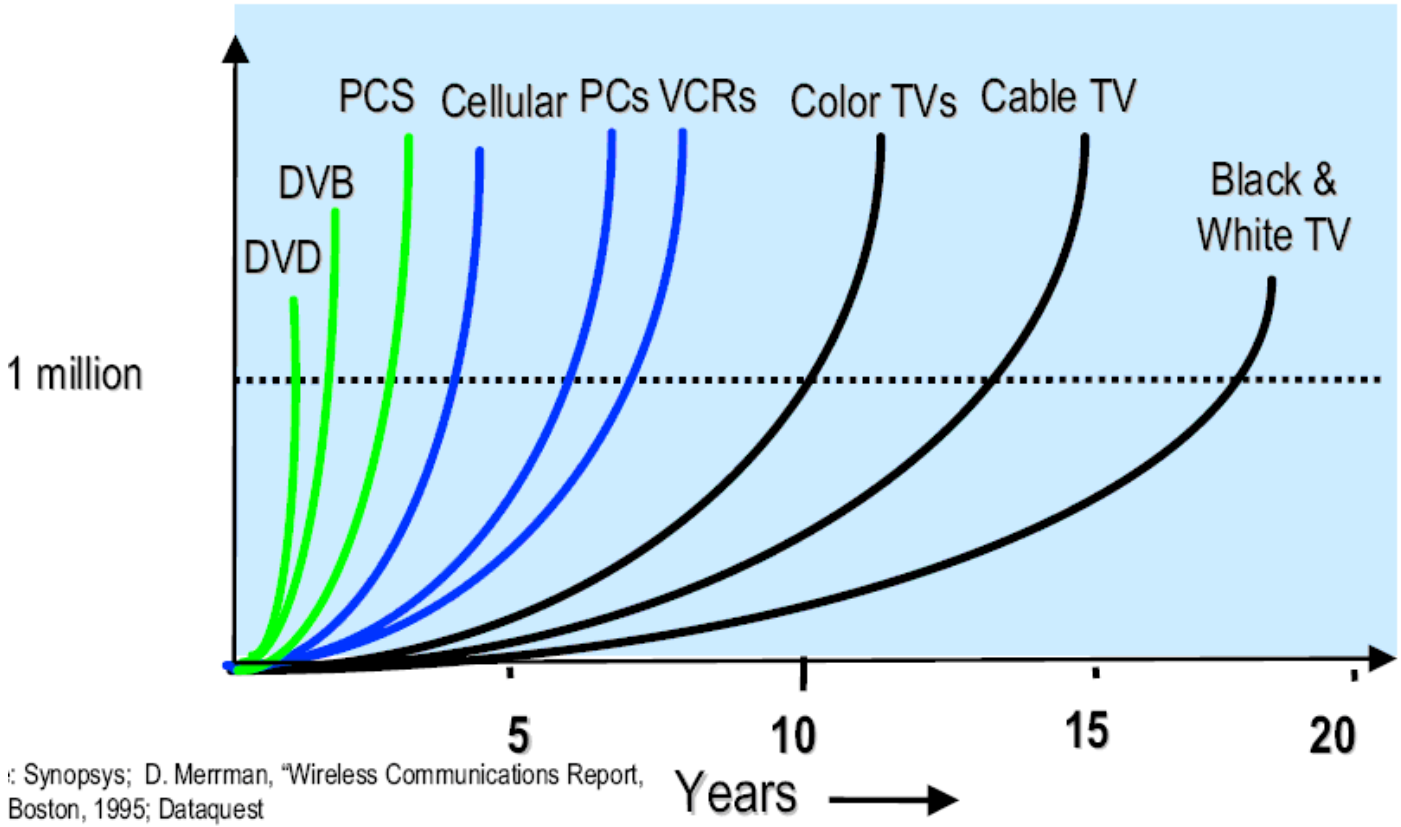
All Programmable Planet
Warren's APP Blog (SerDes)



Design Methodology

Motivations

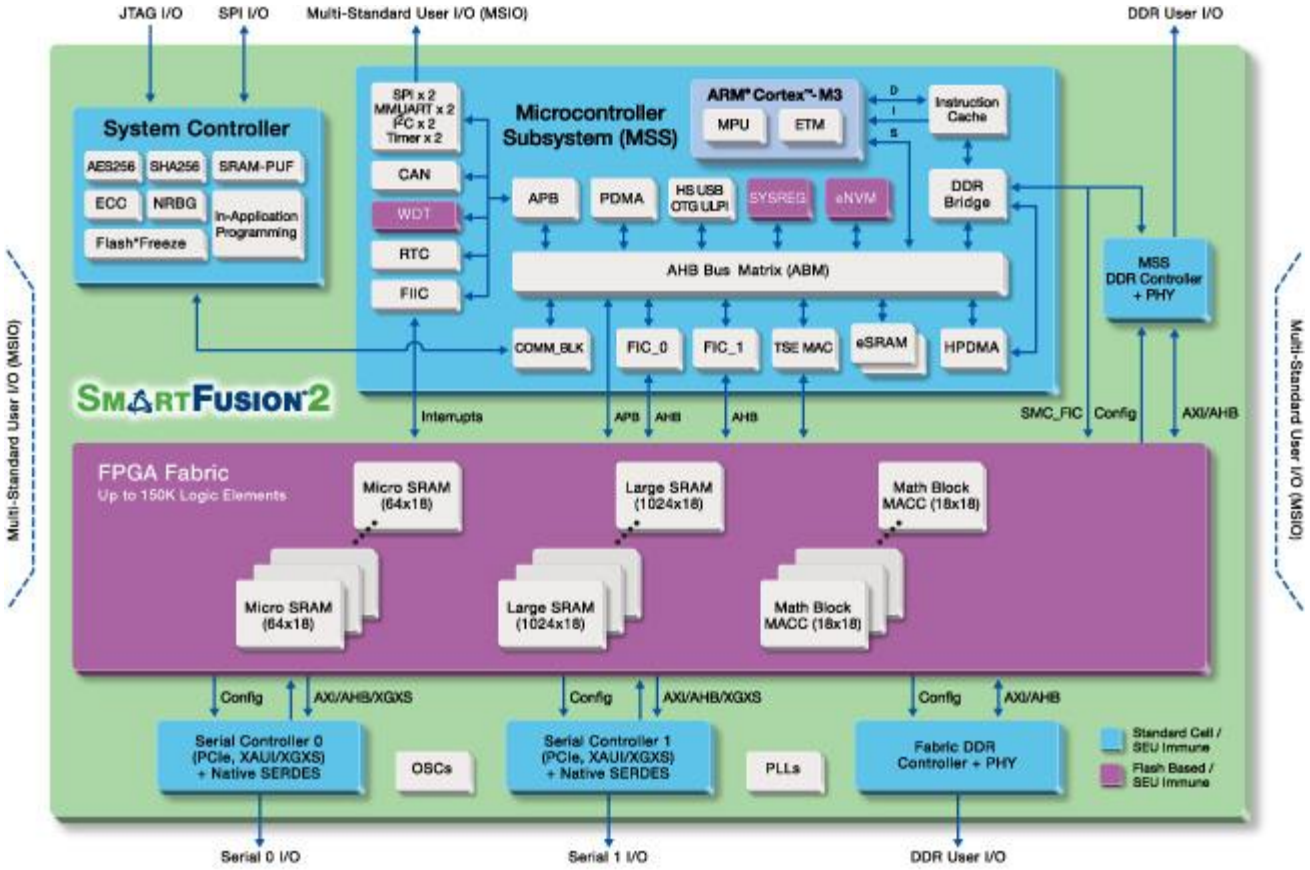
- Accelerated time-to-market and reduce life-cycle
  - Flexibility needs



: Synopsys; D. Merrman, "Wireless Communications Report, Boston, 1995; Dataquest

Motivations

- High integration
  - Basic: memory, logic, I/Os… plus
  - More: PLL, DSP, Micro-controller, Flash, SerDes, clock oscillator…



Motivations

- Design skills: One person cannot do it all
  - Ideal team: System level, DSP algorithms, SW/HW co-design, HDL modeling, Design methodology, Project management, Board level, Signal integrity, High-speed I/Os

Motivation

EE Times BETA News & Analysis

Home | News & Analysis | Business | EE Life | Embedded.com | Design | Products | Education & Tr

News & Analysis
Latest News
Semiconductor News

Don't miss the definitive event for chip, board, and system designers

EE Times Home > News and Analysis

News & Analysis

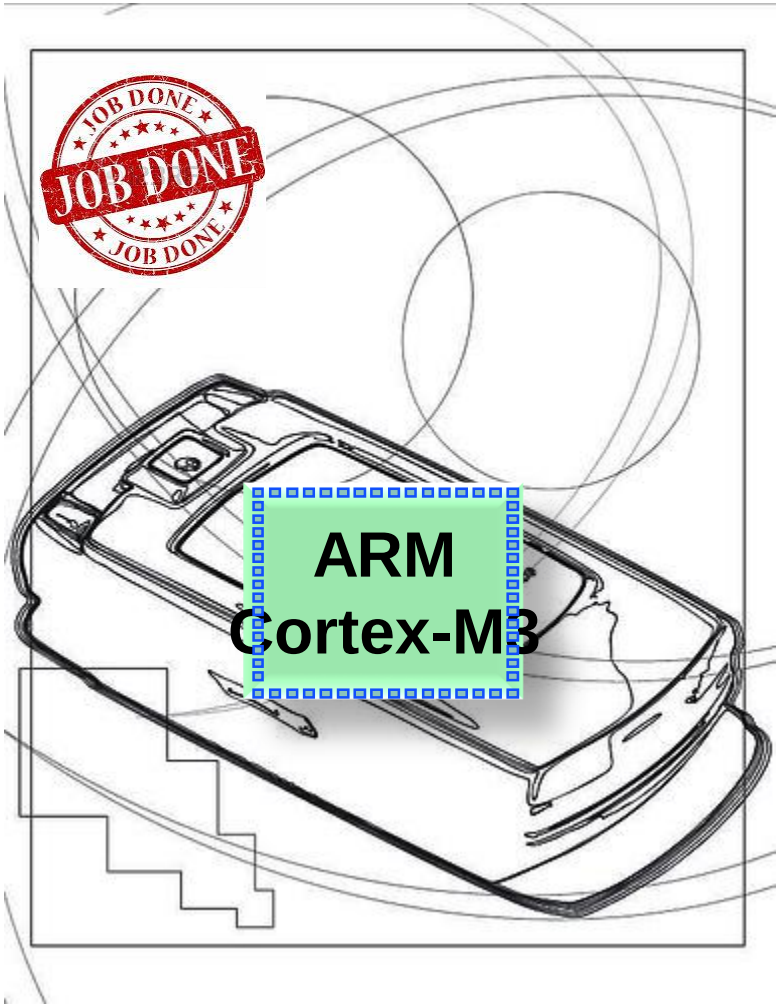
Intel finds design error in chip

Peter Clarke

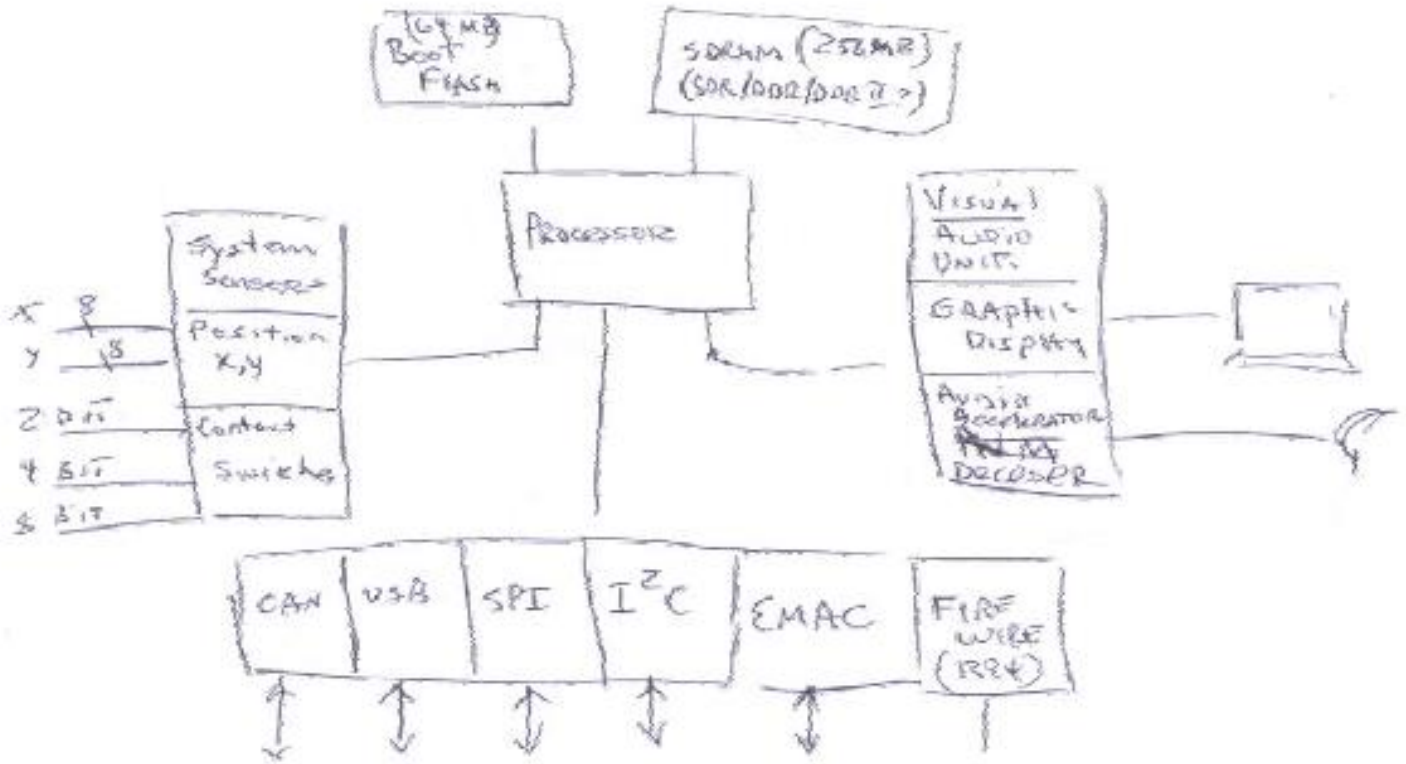
1/31/2011 10:58 AM EST

SANTA CLARA, Calif. - Intel has found a design error in a support chip for the recently announced Sandy Bridge processor. The company says it has implemented a solution but that the error could cost the company $700 million.

SYSTEM DESIGNER'S DREAM



First Step in System Design

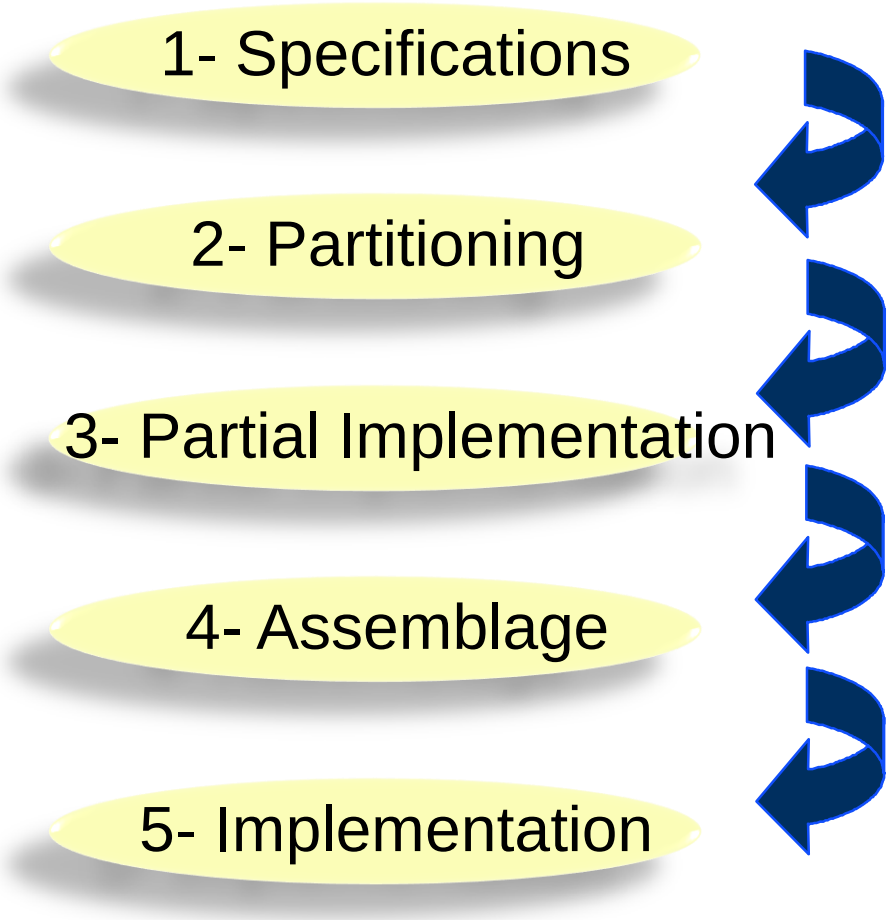


Design Principles

- Hierarchy
  - Divide & conquer
  - Simplification of the problem
- Regularity
  - Divide into identical building blocks
  - Simplifies the assemblage verification
- Modularity
  - Robust definition of all components (entity)
  - Allows easy interfacing
- Locality
  - Ensuring that interaction among modules remains local
  - Makes designs more predictable and re-useable

Team Design Methodology

- Top-Down design methodology in 5 steps



Step 1: Specifications

- Put down the circuit concept
  - Easy verification
  - A reference manual for communication
  - How?
- Put down the requirements
  - Timing budget
  - Power budget
  - Area budget
  - Financial budget

Step 2A: Partitioning

- Divide and conquer strategy
  - Driven by technology, teams, availability (IPs), etc…

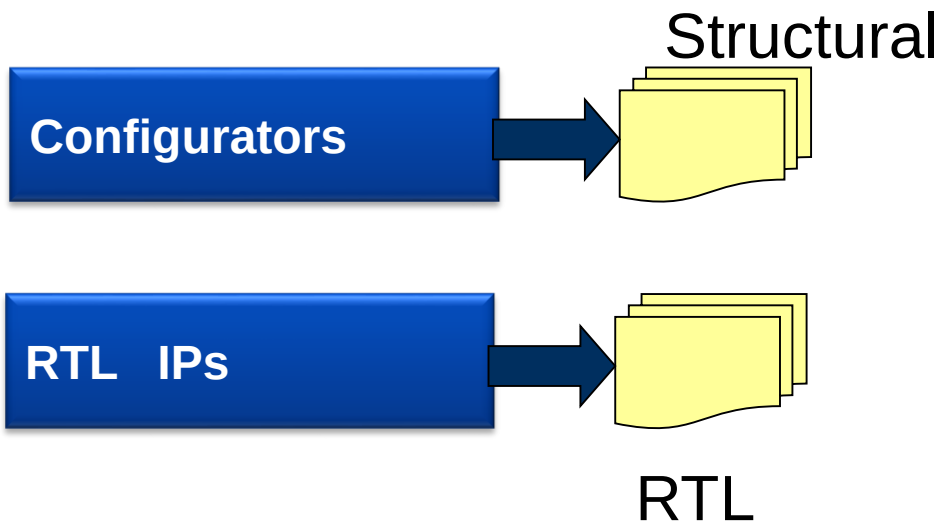
SOFTWARE

HARDWARE

Step 2B: Partitioning

- Divide and conquer strategy
  - Driven by technology, teams, availability (IPs), etc…

Step 3: Partial Implementation

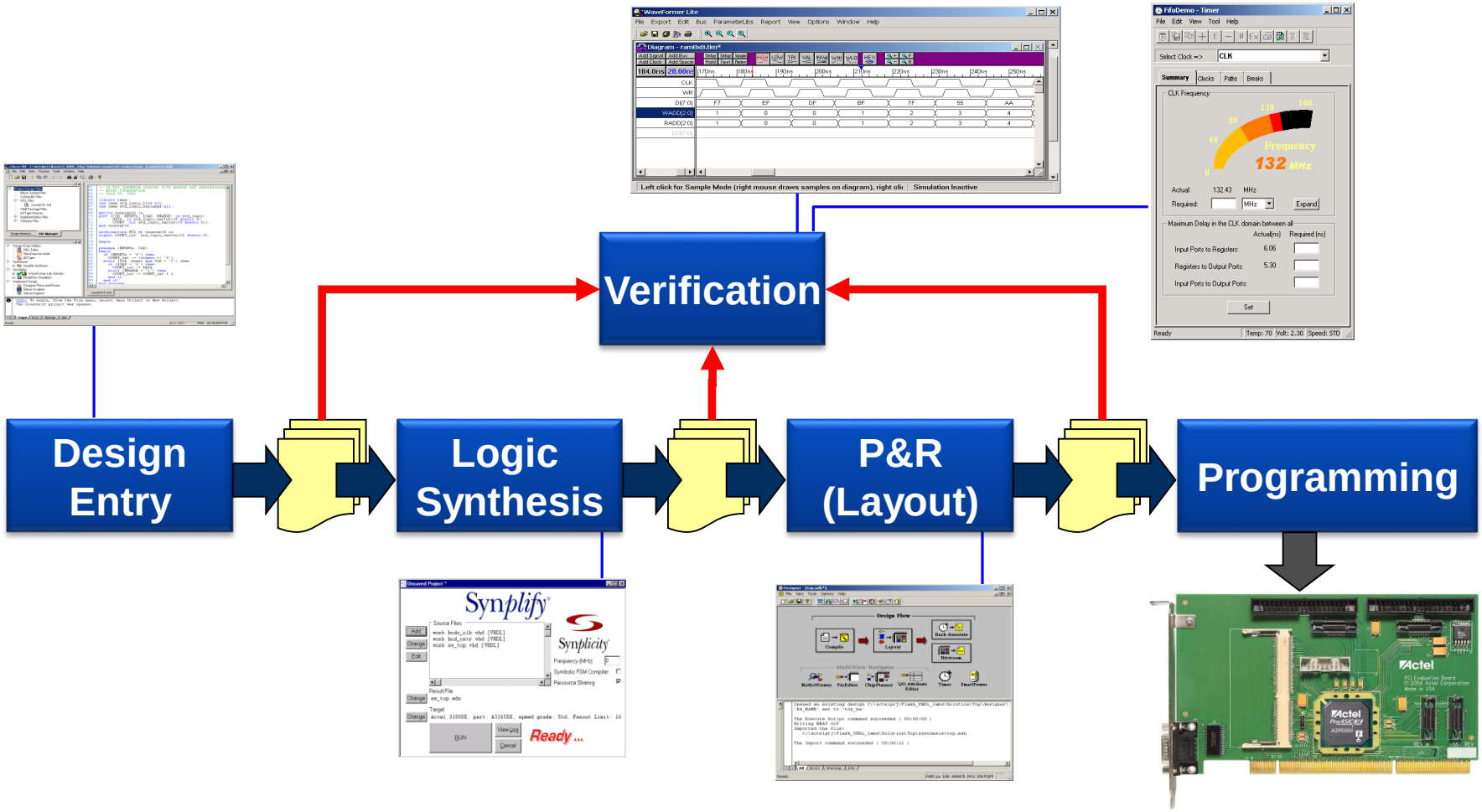


Step 4: Assemblage

- Hierarchical way
- Start from the lowest level
- Includes mixed-level description
- Final product validation is now possible
  - Compare to original specifications
  - Simulate
  - On-board verification

Step 5: Full Design Implementation

- Simplified FPGA design implementation flow



Step 6: Validation

- Simulation
  - Bus Functional Model (BFM)
  - Mixed language HDL simulation

- Hardware Prototype for system validation

Timing Analysis & Design Constraints

Barto's Law

"Every circuit is considered guilty until proven innocent"

Timing Golden Rule

Detecting problems as early as possible

Timing in the Design Flow

- Timing Driven Synthesis
- Timing Driven Optimization
- Timing Driven Floor-Planner
- Timing Driven Place & Route

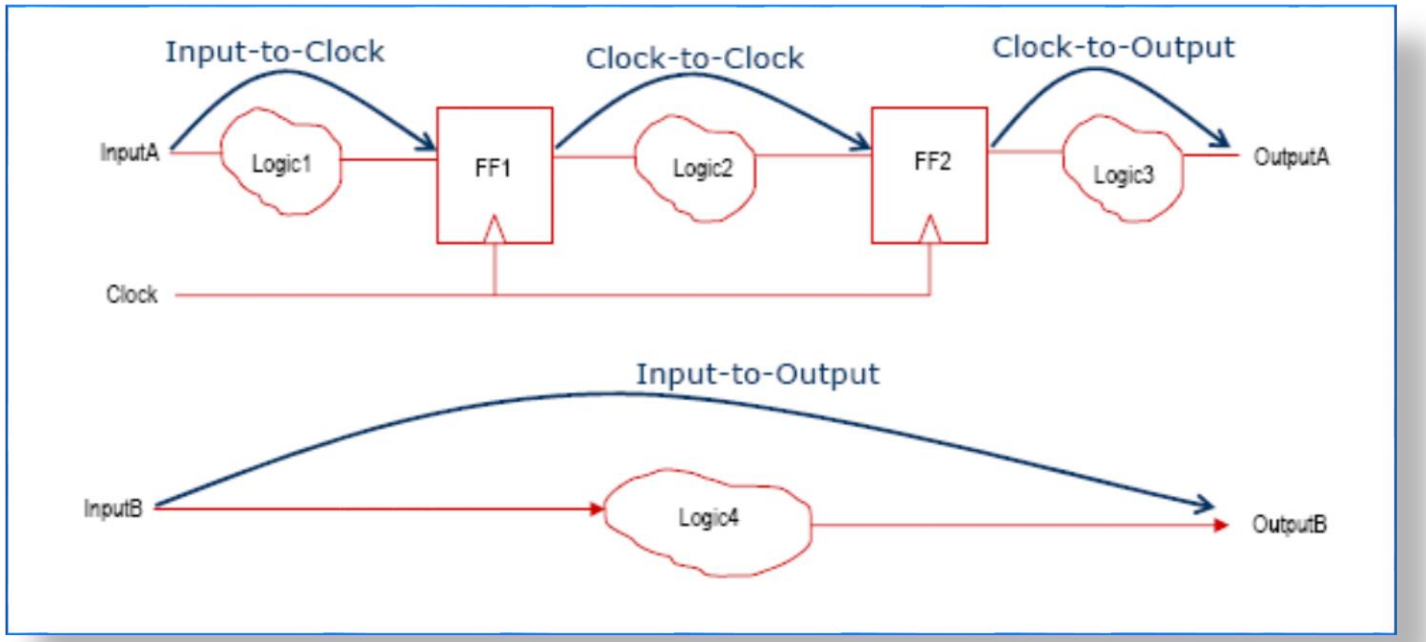
Simulators versus Verifiers

- Simulators
  - Circuit-Level
  - Timing
  - Switch-Level
  - Logic-Level

- Verifiers (Pattern Independent)
  - Static Timing Analysis

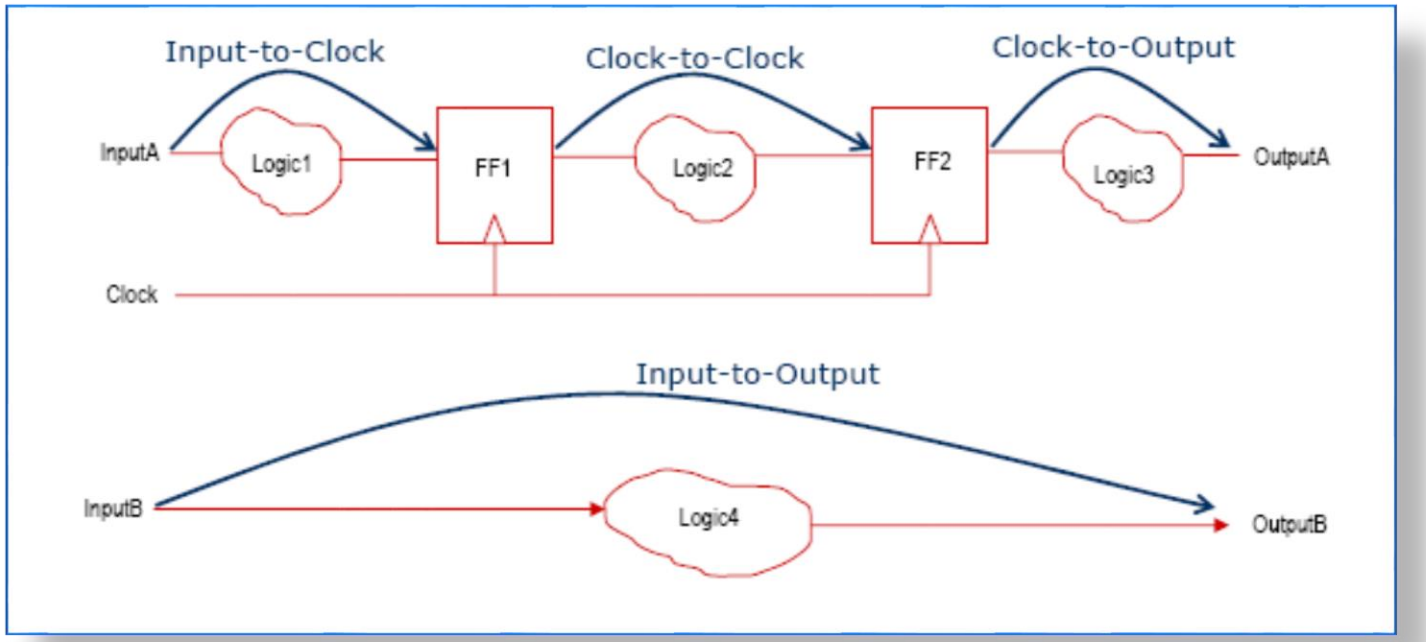
Timing Paths

- 4 types of basic timing paths in a synthesized digital design:
  - Input to registers
  - Registers to output
  - Input to output
  - Registers to registers



Timing Paths

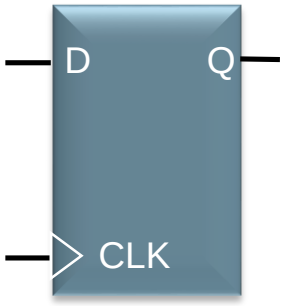
- These basic timing paths can apply to:
  - A module within an ASIC/FPGA
  - A whole ASIC/FPGA
  - A system with multiple chips



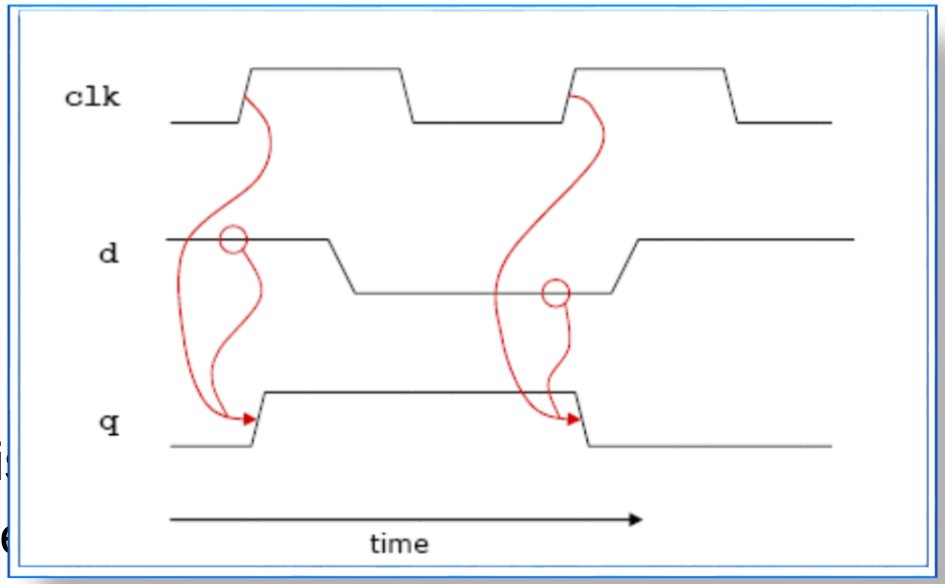
What are Timing Constraints?

- Maximum or minimum limits placed on timing paths
  - Input to registers: External Setup / Hold
  - Registers to output: Maximum / Minimum Clock-to-Out
  - Input to output: Maximum / Minimum Delay
  - Registers to registers: Maximum Clock Frequency
- Other exceptions: False Paths, Multi-Cycle Paths
- Usually expressed in ns or ps
- First understanding flip-flop timing parameters

Flip-Flop Timing: Overview

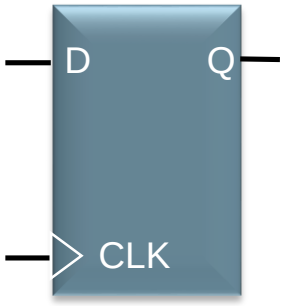


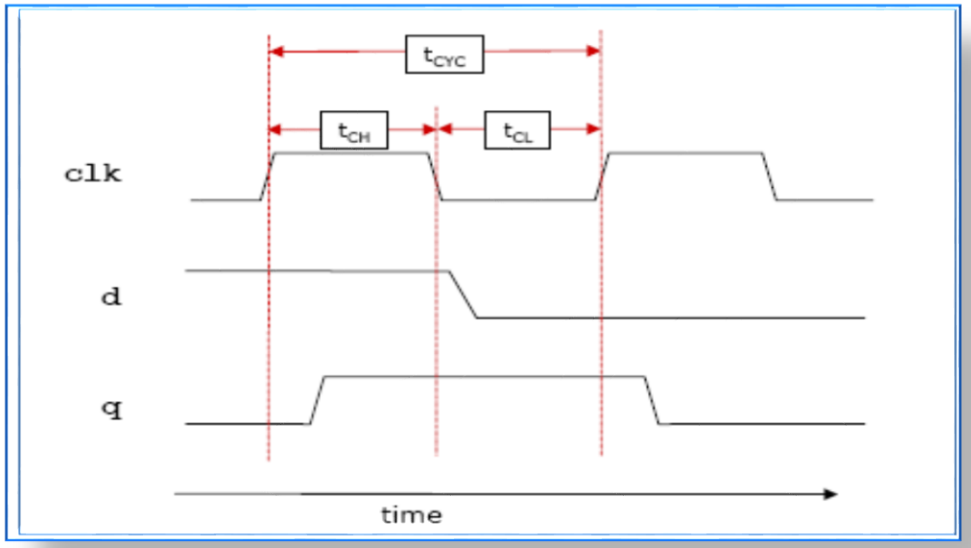

```
INFER: process (CLK) begin
  if (CLK'event and CLK ='1') then
       Q  <= D;
  end if ;
end process INFER;
```



- The level of d must be stable for some amount of time before and after the sampling clock edge

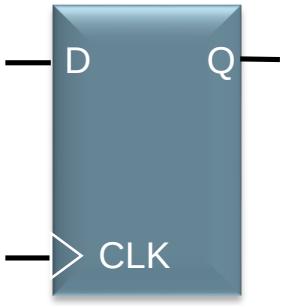
Flip-Flop Timing: Clock Requirements

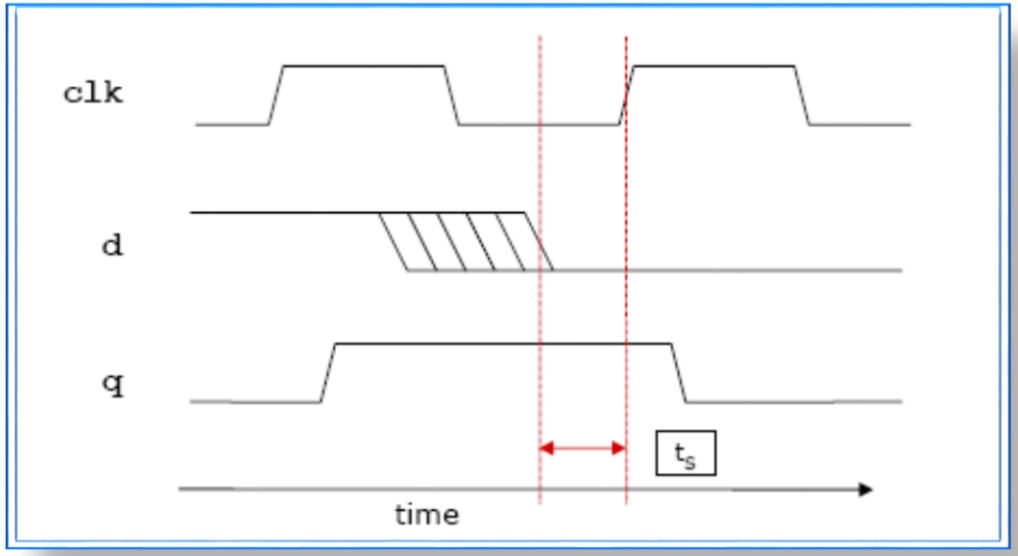




- Clock Parameters
  - Clock cycle time (t_{CYC}), minimum
  - Clock pulse width high (t_{CH}), minimum
  - Clock pulse width low (t_{CL}), minimum

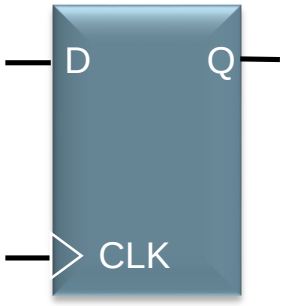
Flip-Flop Timing: Input Setup

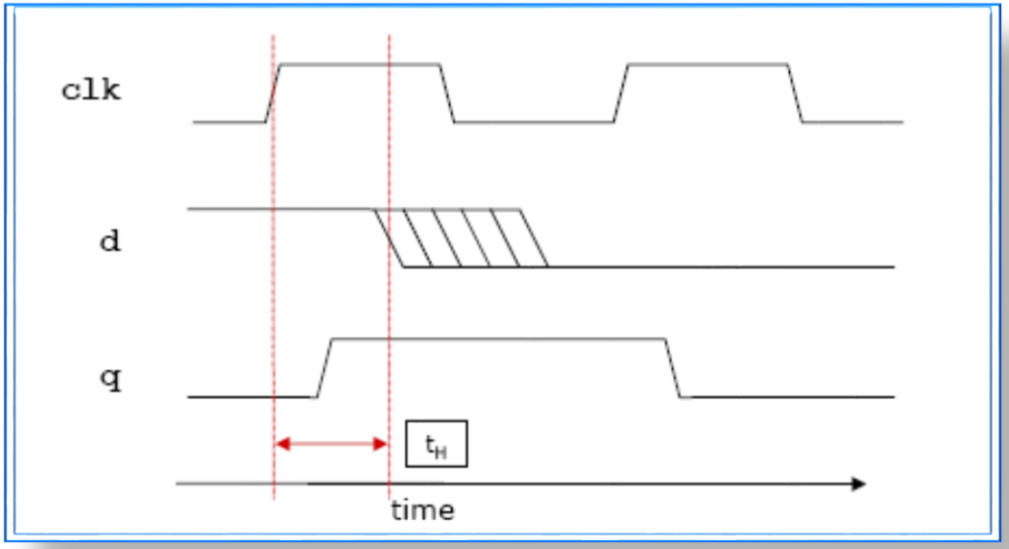




- Input Setup (t_S)
  - The minimum time that the D input must be stable before the active (rising or falling) edge of the clock

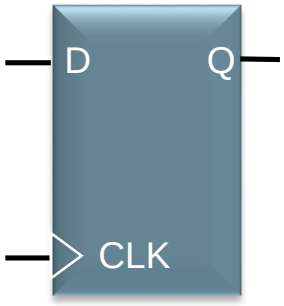
Flip-Flop Timing: Input Hold

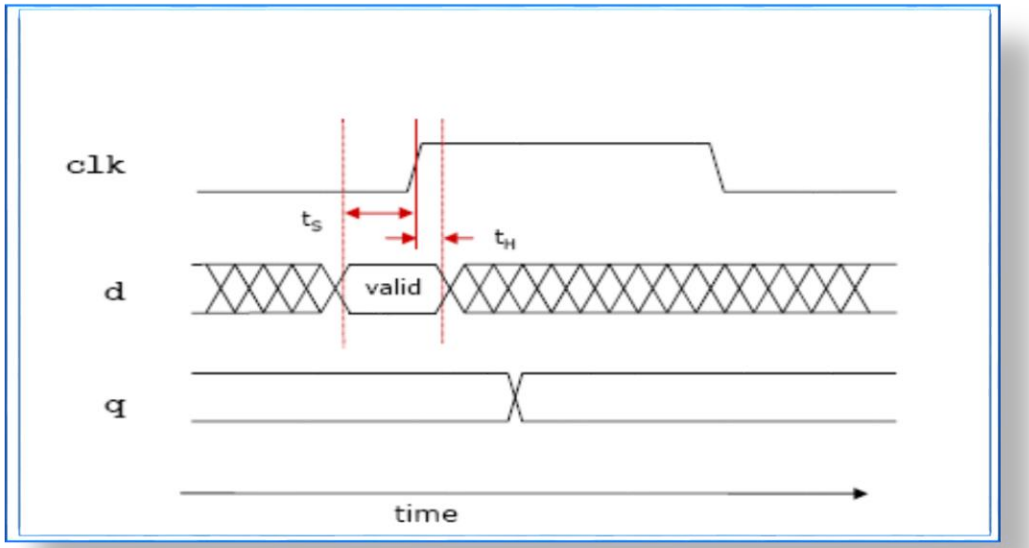




- Input Hold (t_H)
  - The minimum time that the D input must be stable after the active edge of the clock

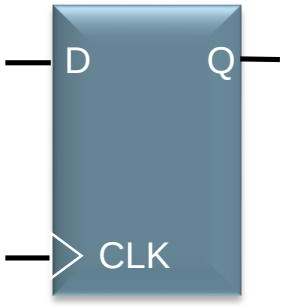
Flip-Flop Timing: Stability Requirements

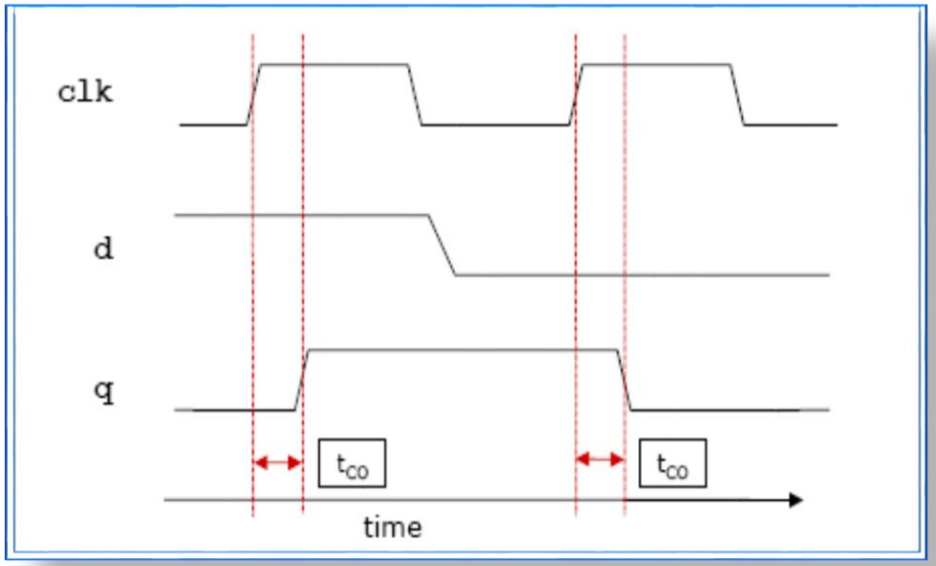




- Setup and Hold define a minimum window around the active clock edge during which D must be stable
- t_S or t_H may be negative, but $t_S + t_H > 0$

Flip-Flop Timing: Clock-to-Out

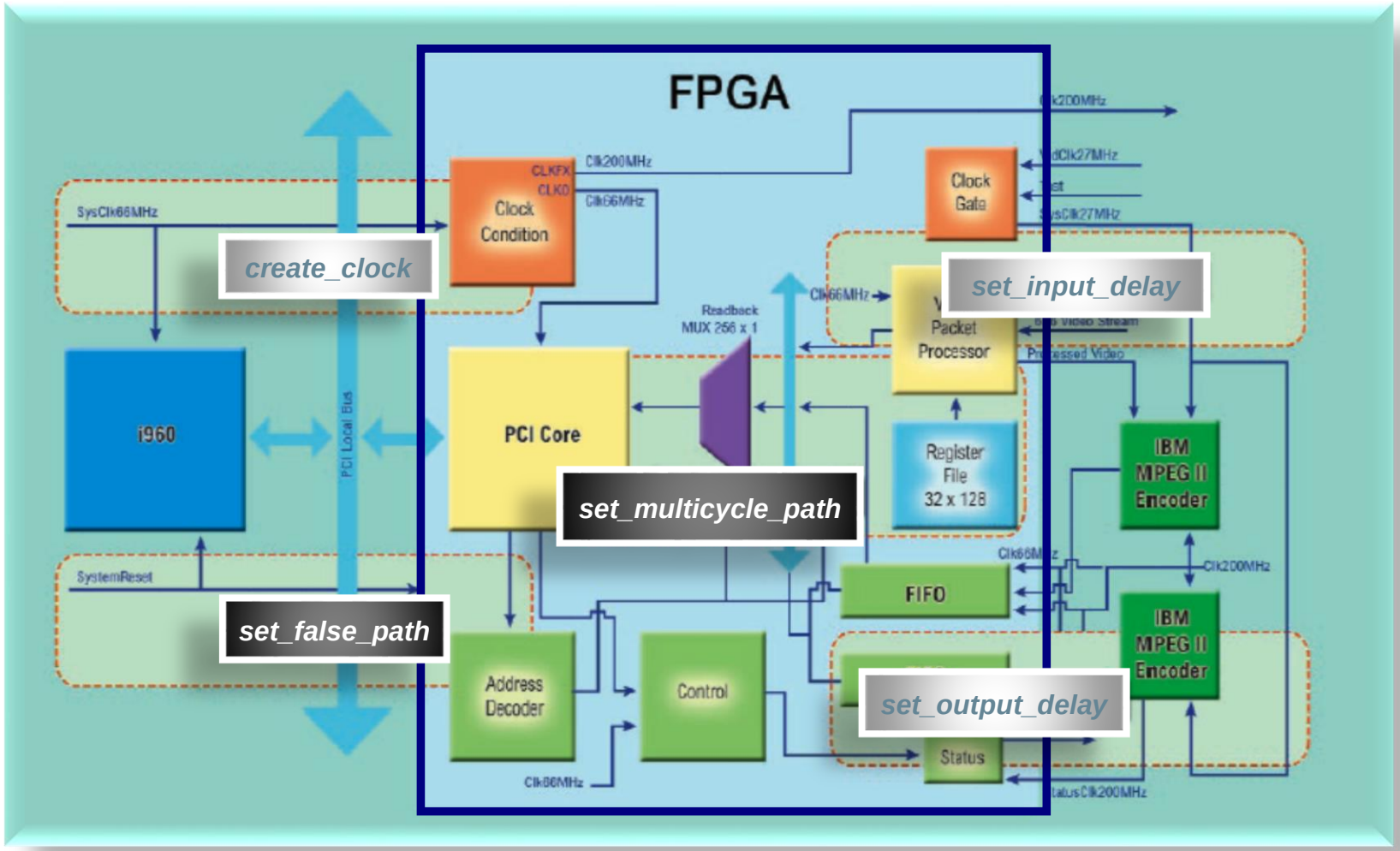




- Clock-to-out (t_{CO}) a.k.a. Clock-to-Q
  - The time delay from the active edge of the flip-flop's clock input to the resulting change in the Q output
- Specified minimum and maximum times

Constraining Designs @ Board-Level

- Constraining FPGA Designs with SDC



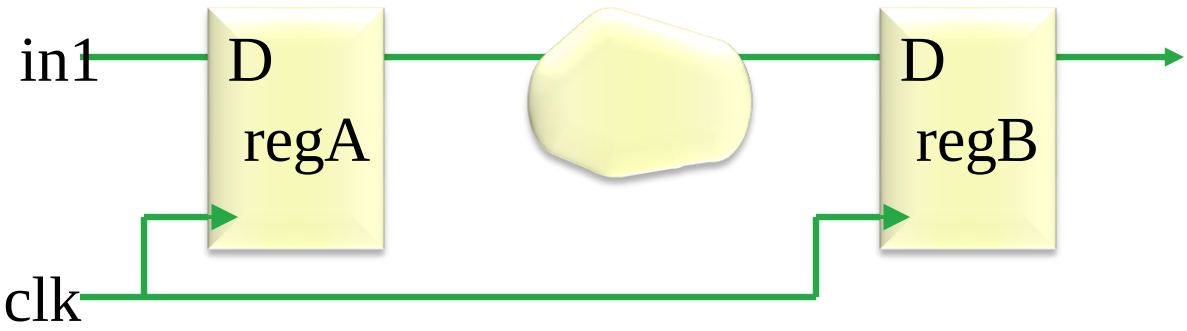
Source: Mentor Graphics Corporation ©, 2002

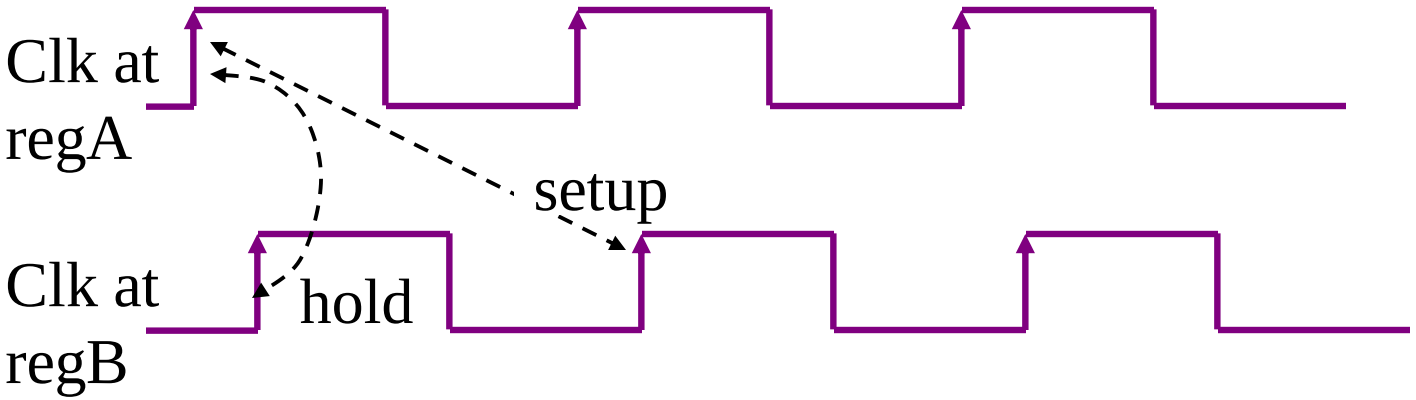
Constraining Designs

- Design Environment: Corner Analysis
  - Operating conditions (Process / Voltage / Temperature)
    Worst: to fix the setup violations
    Typical: mostly ignored
    Best: to fix the hold violations
- Timing Assertions (Design-level)
  - Clock Characteristics
  - Arrival Time at Each Input Port
  - Required Time at Each Output Port
- Timing Exceptions
  - False Paths
  - Minimum / Maximum Path Delay
  - Multi-cycle Paths

Timing Analysis: Setup/Hold Check

- Create Clock: reg-to-reg requirement

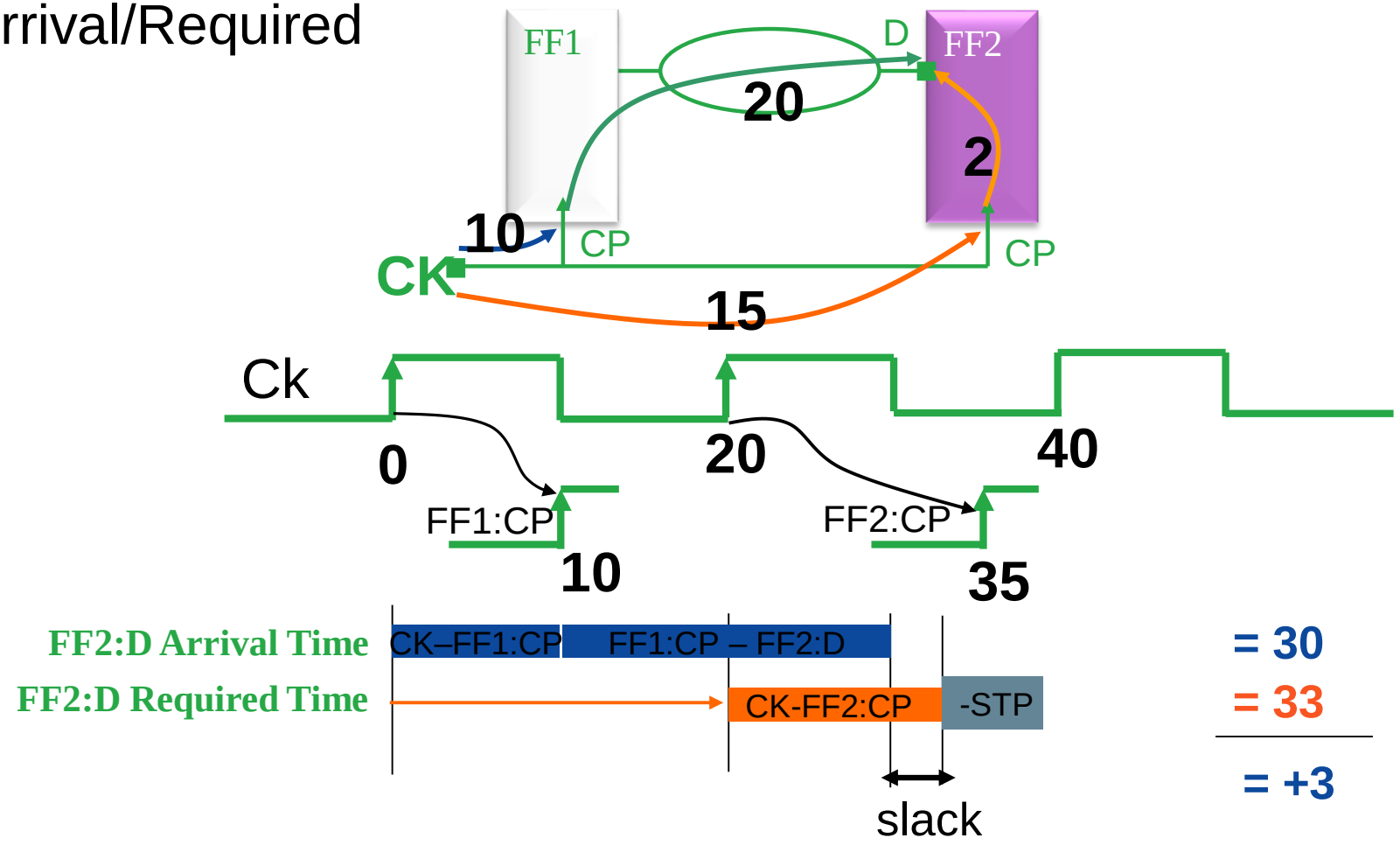




Single-cycle timing relationship

Setup Check

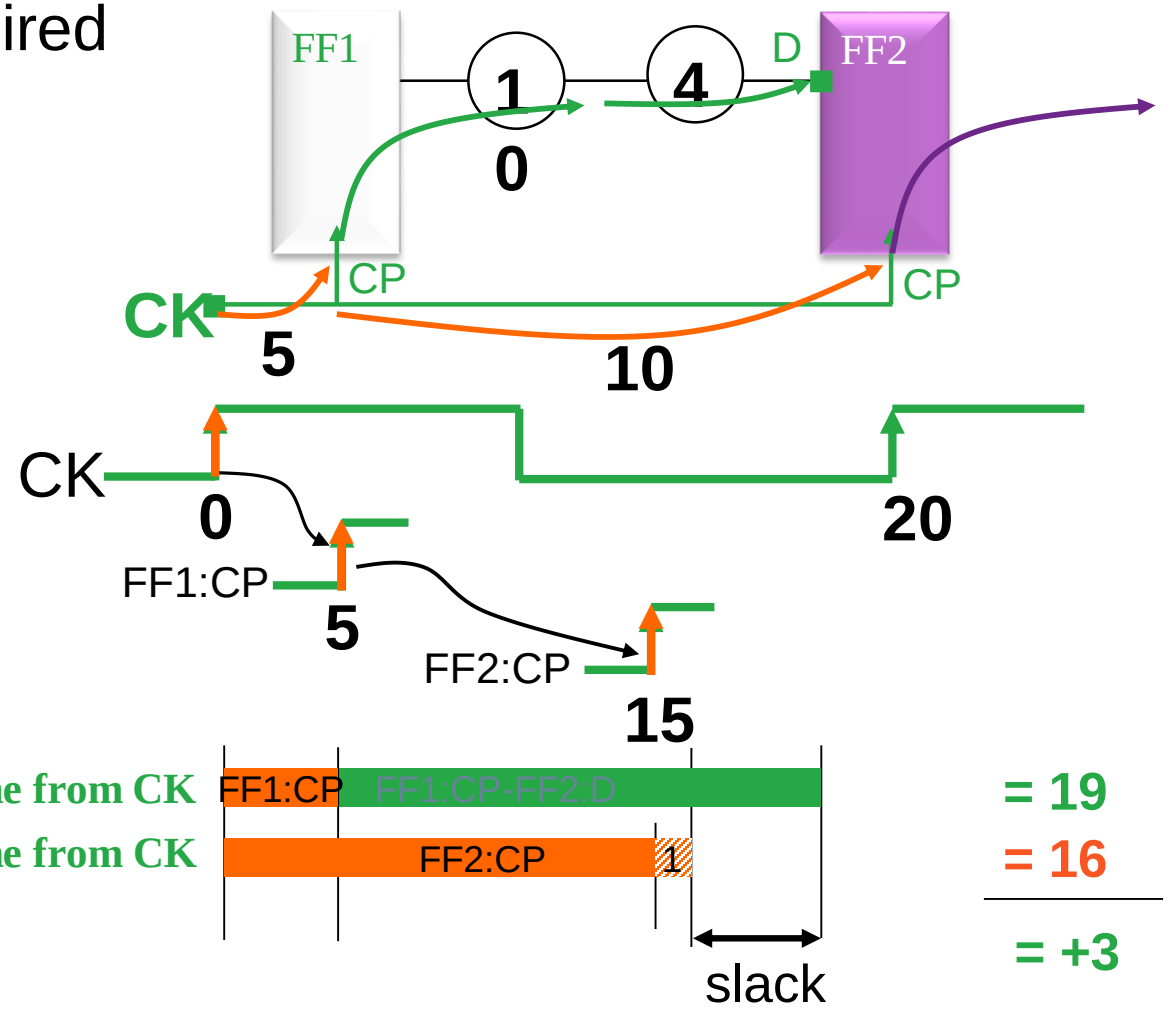
- Arrival/Required



Slack = Required_time – Arrival_time (Violation if < 0)

Hold Check

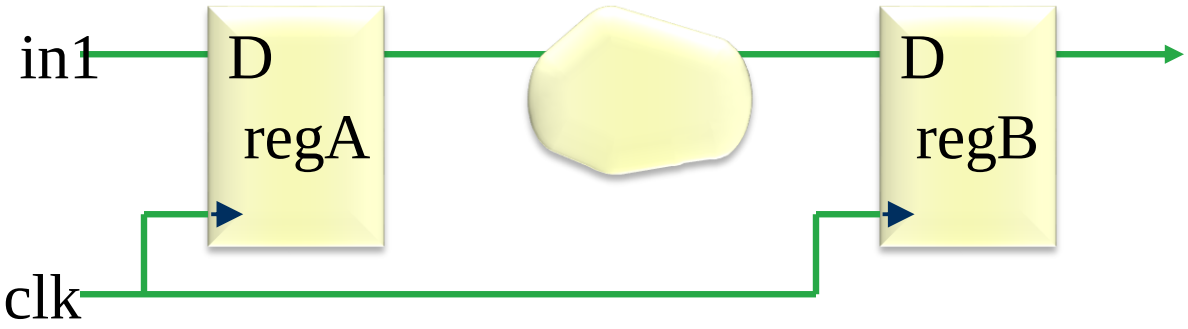
- Arrival/Required

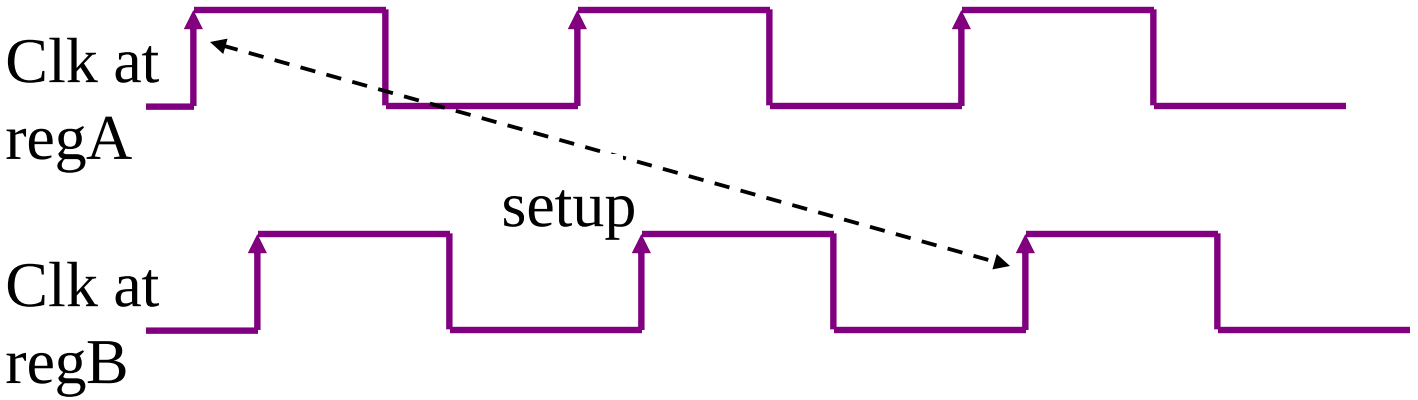


Slack = ArrivalTime – RequiredTime (Violation if < 0)

Timing Analysis: Multi-cycle

- set_multicycle_path: reg-to-reg exception

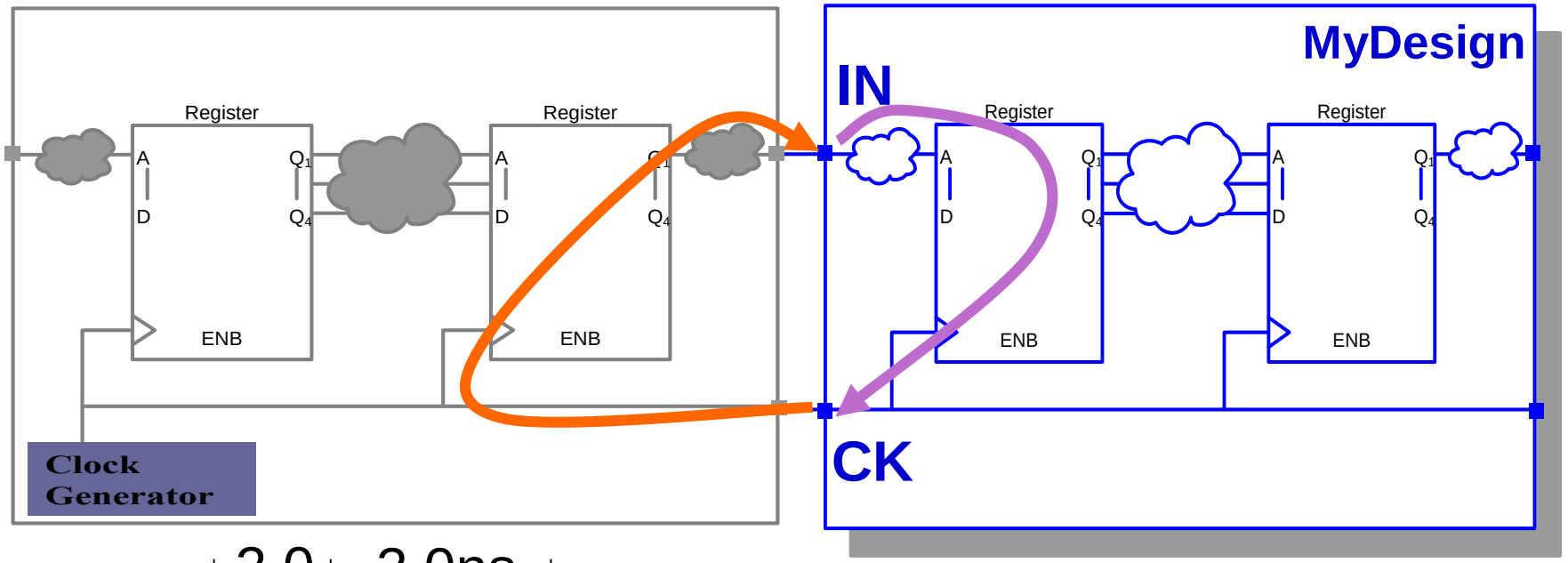


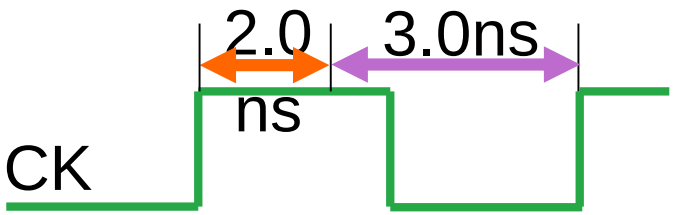


multi-cycle timing relationship

Input Delay Constraints

- Captures External Setup/Hold Requirements

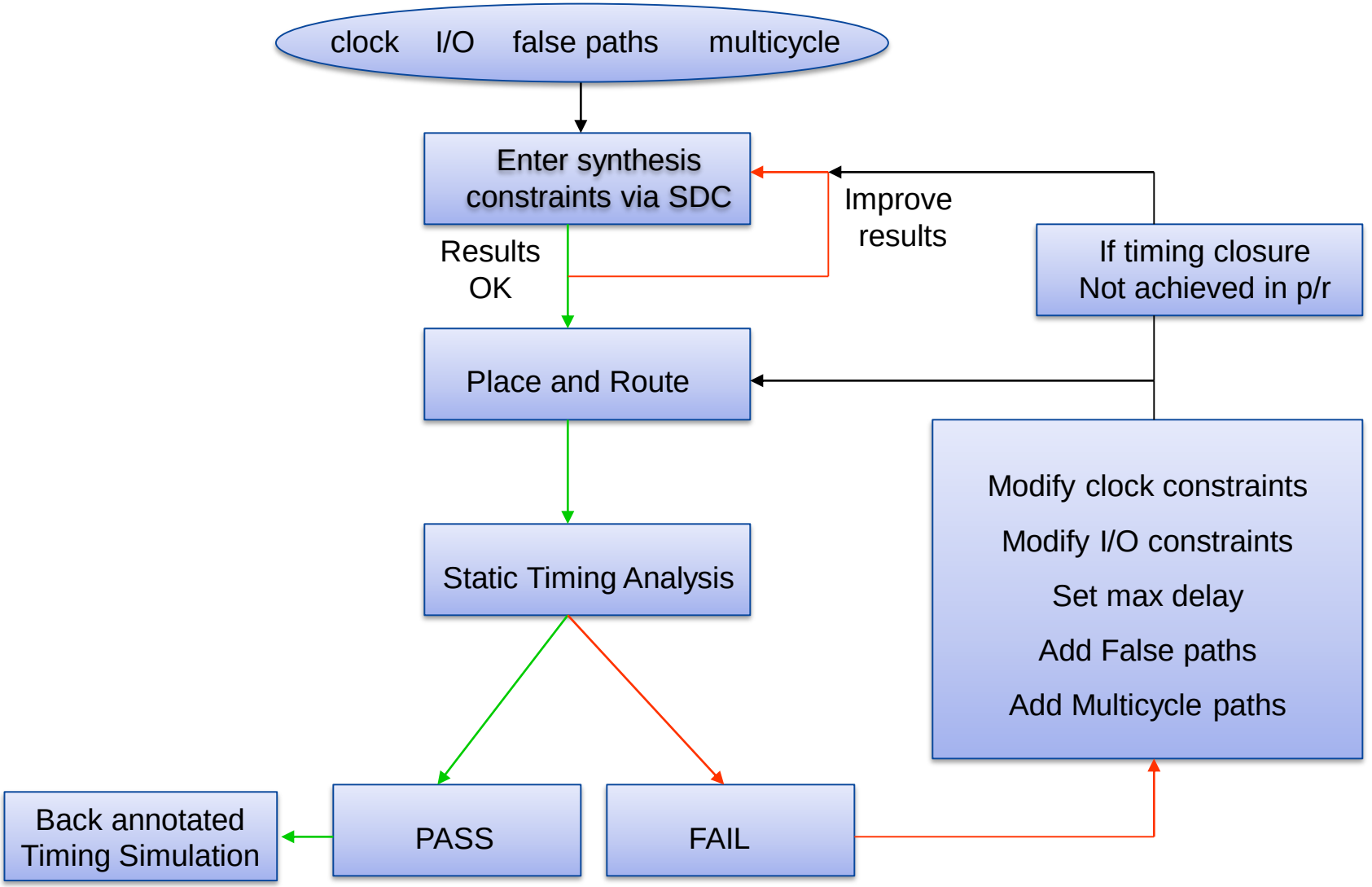




Option "-min" for hold check
Option "-max" for setup check

SDC: set_input_delay 2.0 -clock CK {IN}

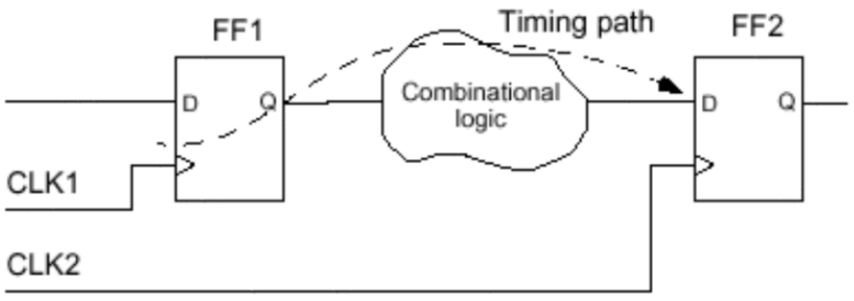
Design Methodology: Timing Closure Loop



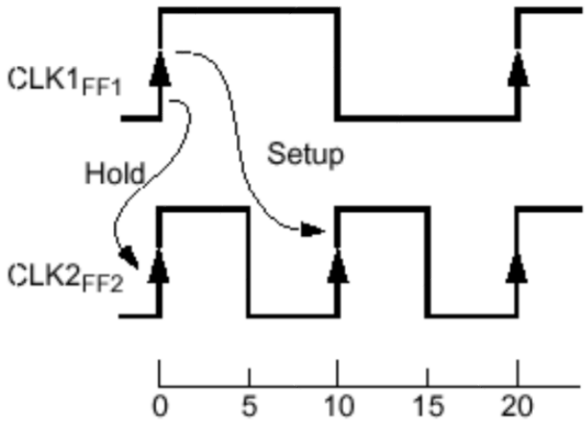
Clock-Domain-Crossing (CDC)

- Study over the Least Common Multiplier

common period




```
pt_shell> create_clock -period 20 -waveform {0 10} CLK1
pt_shell> create_clock -period 10 -waveform {0 5} CLK2
```

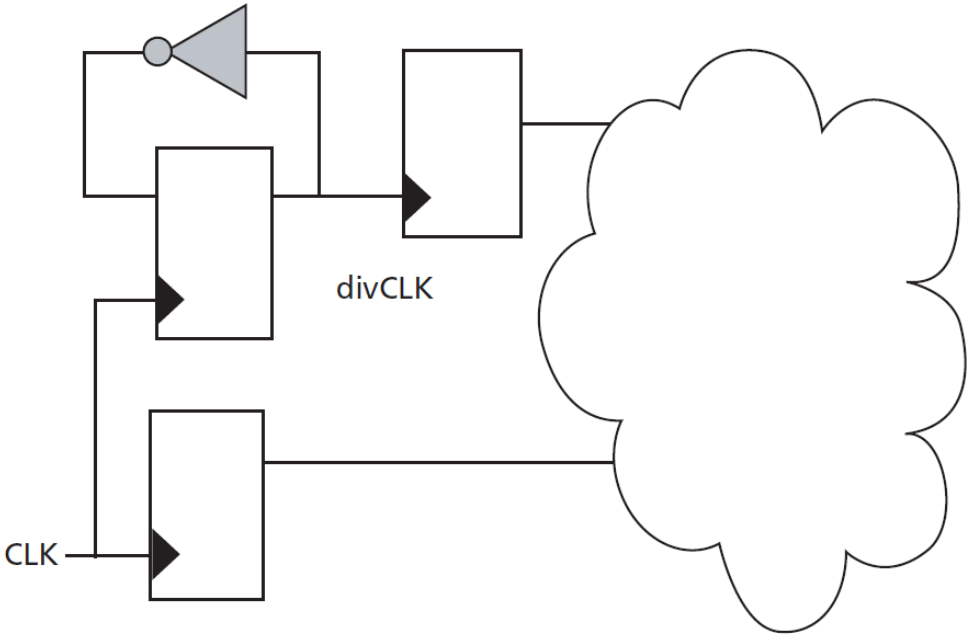


Clock-Domain-Crossing Analysis

- Today's systems have a multitude of components working with different clock-domains running at varying speeds
- Examples range from the small mobile phone chips to huge graphics or microprocessors that interface with a variety of busses and I/Os
- Signals that cross the clock-domain boundaries can be typically classified into two types:
  - Synchronous
  - Asynchronous
- Synchronous crossings are those where the receiving domain has a phase/frequency relationship with the sending domain

Synchronous Clock-Domain-Crossing

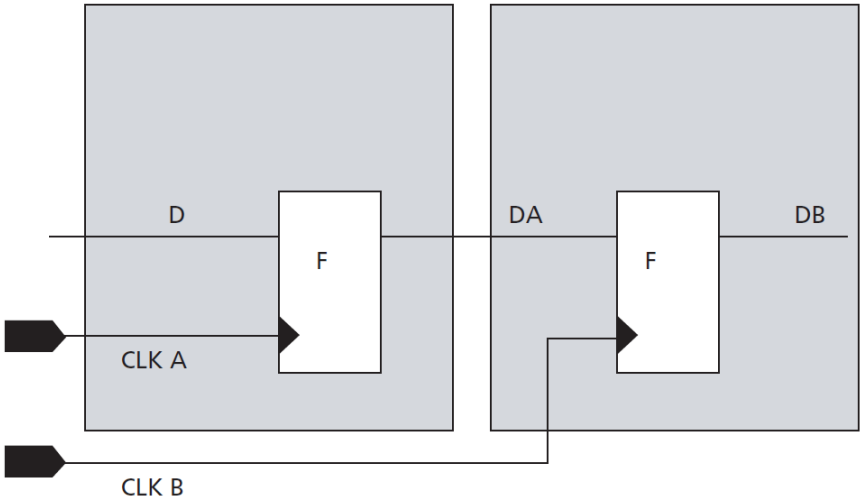
- Synchronous crossings are those where the receiving domain has a phase/frequency relationship with the sending domain



- These crossings are timed and verified robustly that they meet the timing requirements

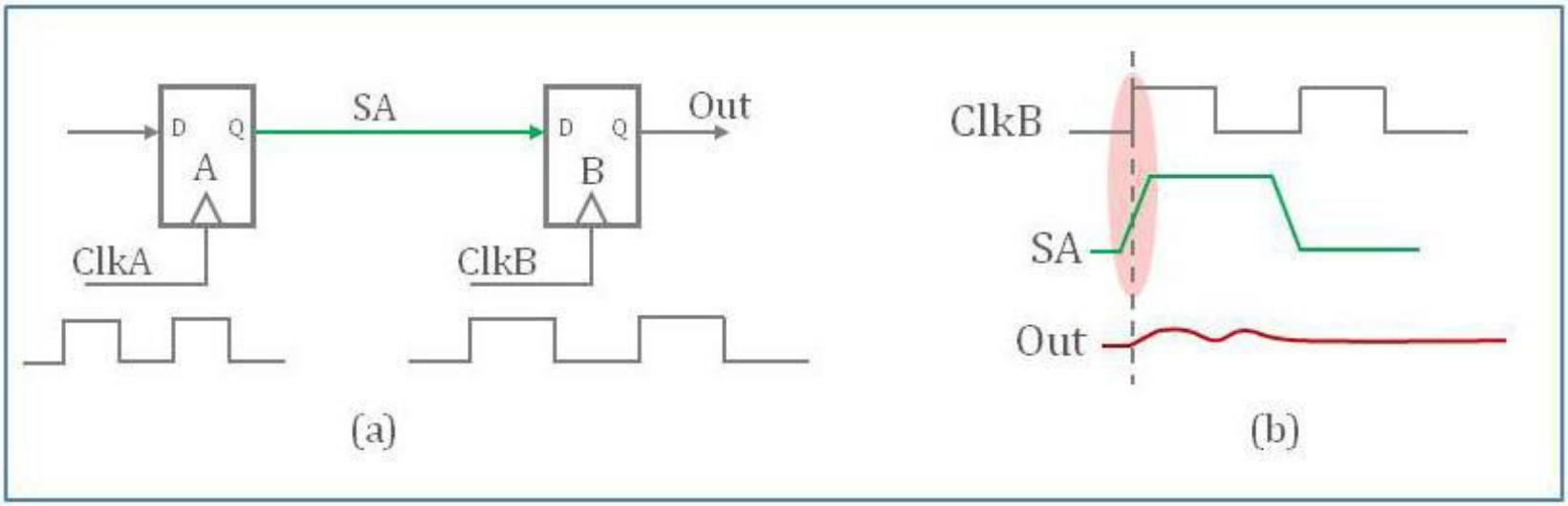
Asynchronous Clock-Domain-Crossing

- Asynchronous crossings are those where there is no relationship between the sending and receiving clocks
- These clocks originate from different clock generators or derivatives of those



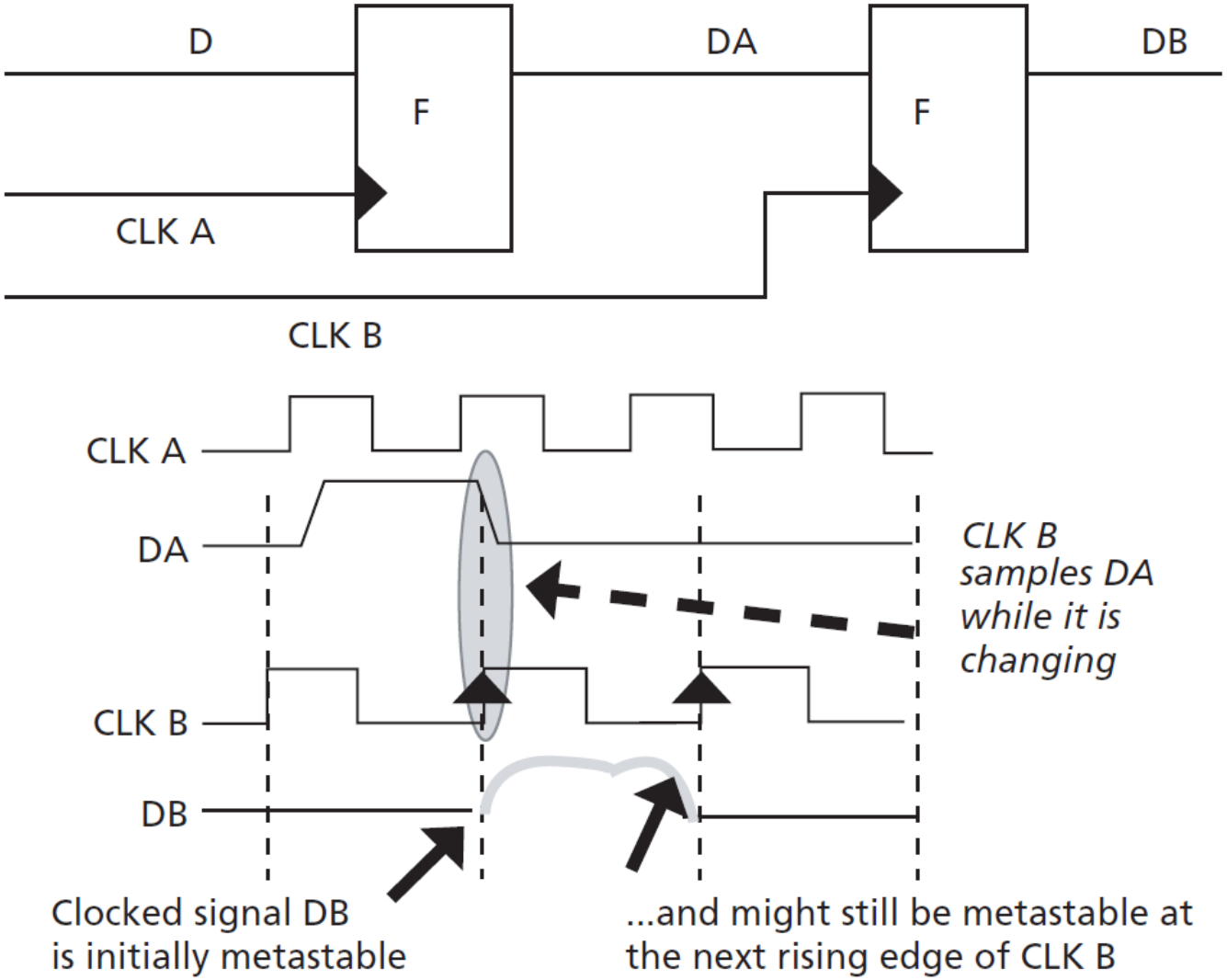
- As a result, timing cannot be accurately verified since the order of clock edges cannot be guaranteed

Metastability



- Metastbility refers to signals that do not assume stable 0 or 1 states for some duration of time at some point during normal operation of a design
- In a multi-clock design, metastability cannot be avoided but the detrimental effects of metastability can be neutralized

Metastability



Metastability

- Typical usages of signals flowing from one asynchronous domain ClkA to another domain ClkB can be categorized into the following types:

1. Reset signals
2. Single-bit Data signals
3. Multi-bit Data signals
4. Synchronized Control signals

Mean Time Between Failures

- The metastability occurrences can be predicted by using the mean time between failures (MTBF) formula

$$MTBF = \frac{e^{C2*tMET}}{C1 * fclk * fdata}$$

- Where *C1* and *C2* are constants that depend on the technology used to build the flip-flop; *tMET* is the duration of the metastable output; and *fclk* and *fdata* are the frequencies of the synchronous clock and the asynchronous input, respectively

Synchronization, Control & Data Paths

- Reset signals are used to reset the logic in the receiving domain during chip-reset phase or whenever interrupts such as software resets or aborts happen in the sending domain
- Single-bit data signals are typically used to convey some sort of status to the receiving domain; for example, to convey that the sending domain is busy or to gate the receiving domain clock
- Before usage in the receiving domain, both these kind of signals need to be synchronized (usually with a double-flop synchronizer)
- Synchronization is sufficient because these signals are intended to transition intermittently and be stable the rest of the time

Synchronization, Control & Data Paths

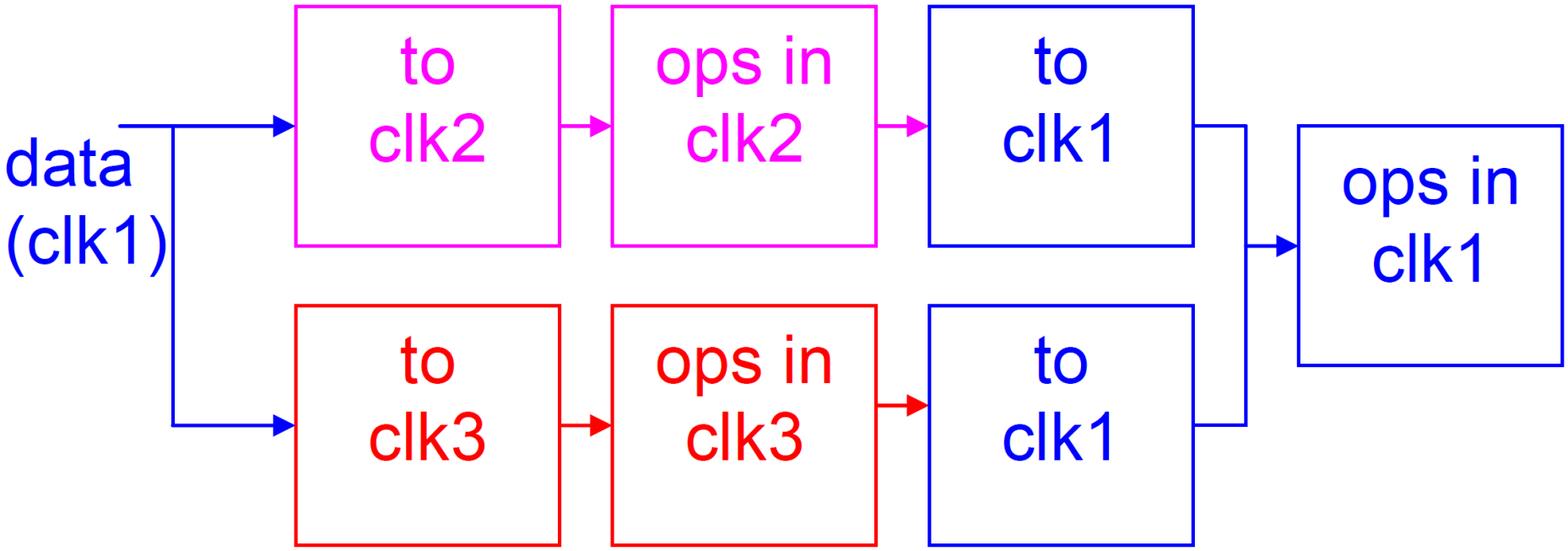
- On the other hand, multi-bit data transfer is used to transfer buses of data signals between domains
- If each of these bits is individually synchronized, the outputs of the synchronizers lose their correlation due to the metastability problem
- Hence, some sort of common control mechanism is required to

  (i) let the receiving domain know that the transmitted multi-bit data is valid and

  (ii) let the receiving domain capture that data only when it is stable
- This is often accomplished via handshake based synchronized control signals or via FIFO based synchronization

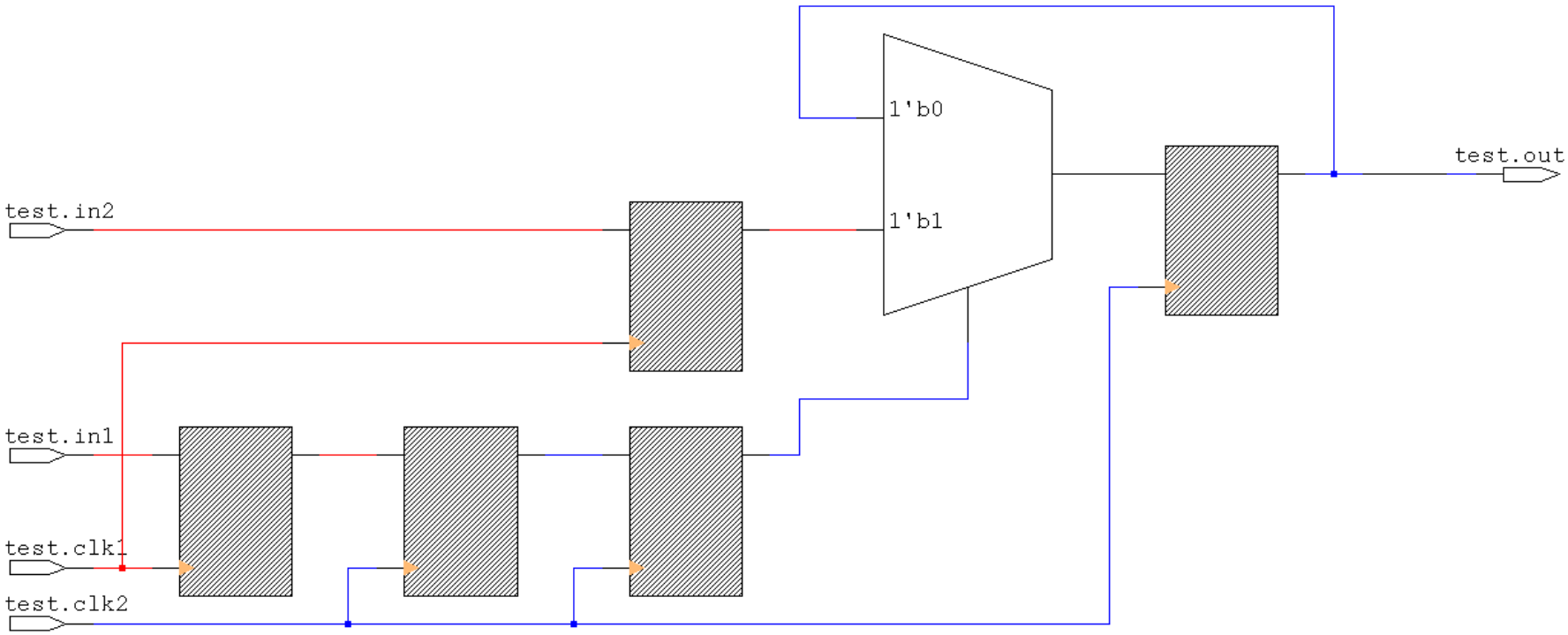
Synchronization, Control & Data Paths

- This kind of transfer has to be designed very carefully



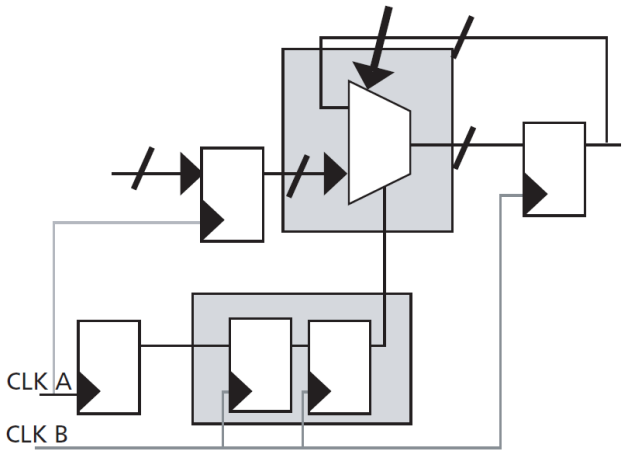
Synchronization, Control & Data Paths

- MUX synchronizer



Synchronization, Control & Data Paths

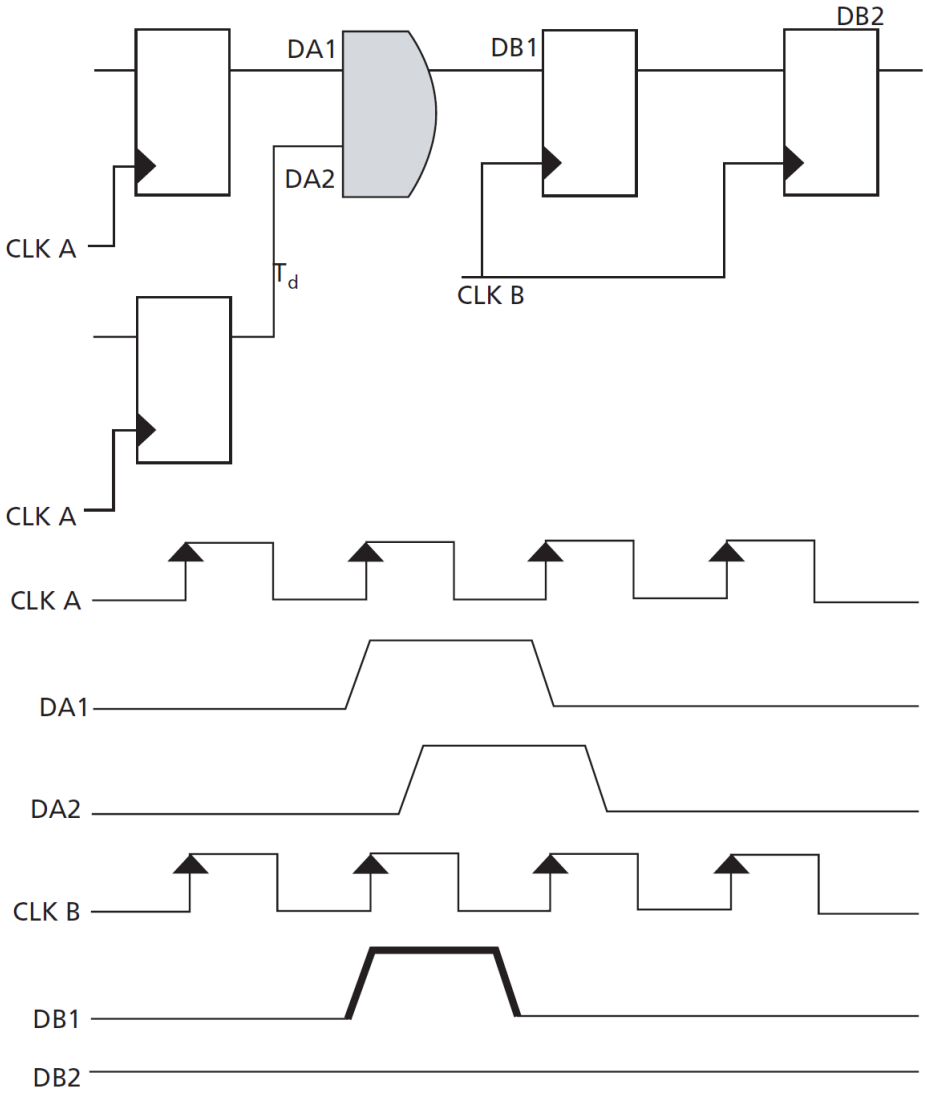
- The MUX synchronizer has a critical requirement for all input in terms of the domains and functionality:



  - The select input of the MUX comes from the destination domain (domain into which the signal is being synchronized)
  - One of the MUX inputs is coming from the destination domain—that is, the holding loop
  - The MUX inputs can be source, destination, or user-specified static signals
  - The logic between the MUX synchronizer and the destination flop is driven by the destination domain or static signals

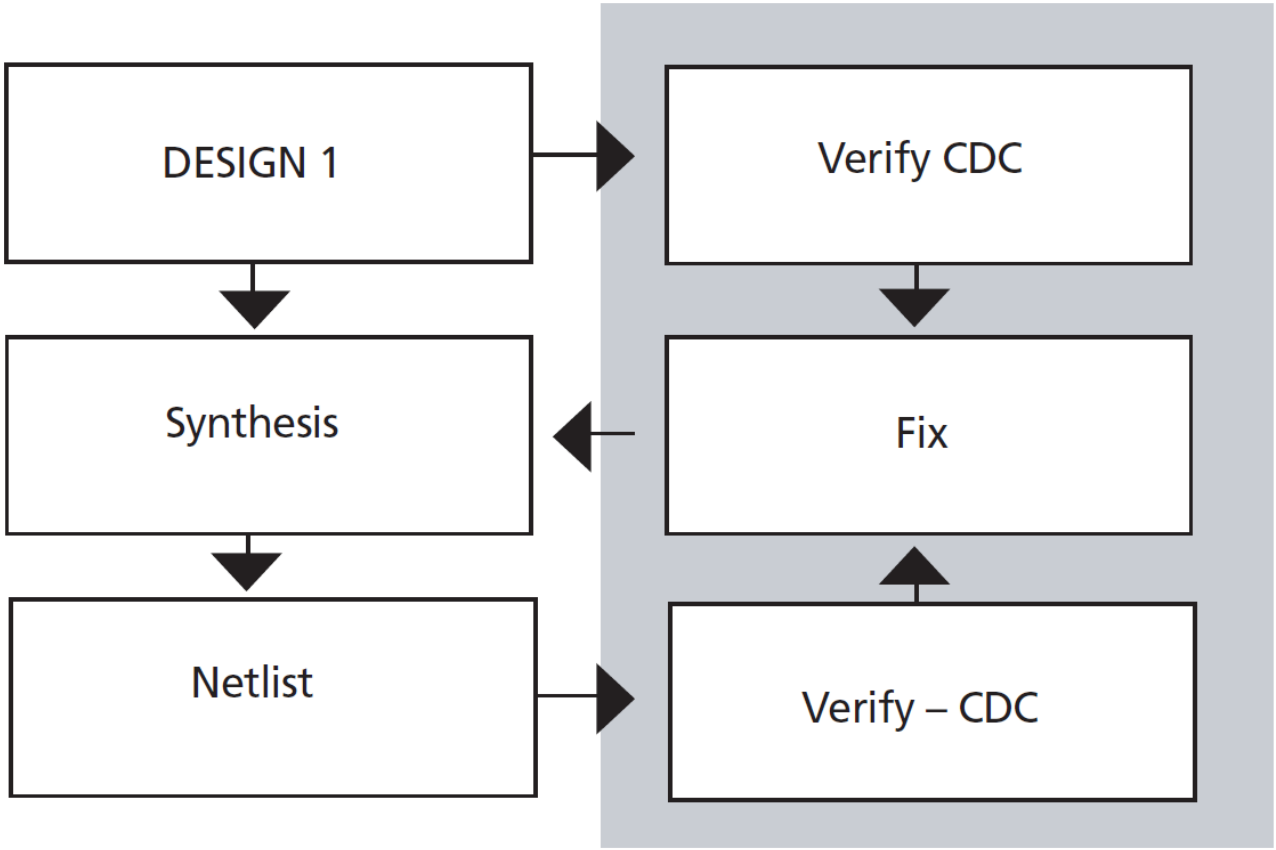
Convergence in the Crossover Path

- Clock domain crossover paths are false paths for timing tools
- Any logic in this path must be carefully crafted and verified, because the logic can cause glitches and create functional errors downstream



Other Challenges in Verification of CDC

- Adding CDC verification in the early design stages verifies and validates the unverified portion of the design

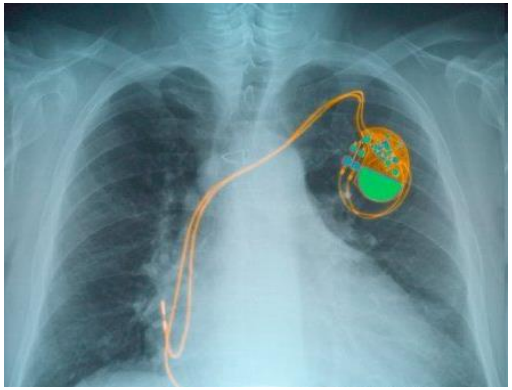


Questions ?

Why Power Management?

- “When I talk to companies, power is the number one problem they have to solve.” – Lip-Bu Tan, Cadence
- “Now, power is also becoming a system problem. You have to start at the top.” – Wally Rhines, Mentor Graphics
- “The world faces one mega-issue, which is power. Electronics could help cut power consumption by 20-30%.” – Aart de Geus, Synopsys

Why Power Management? (cont.)



Battery life

Environmental concerns

Cooling Cost

System Reliability

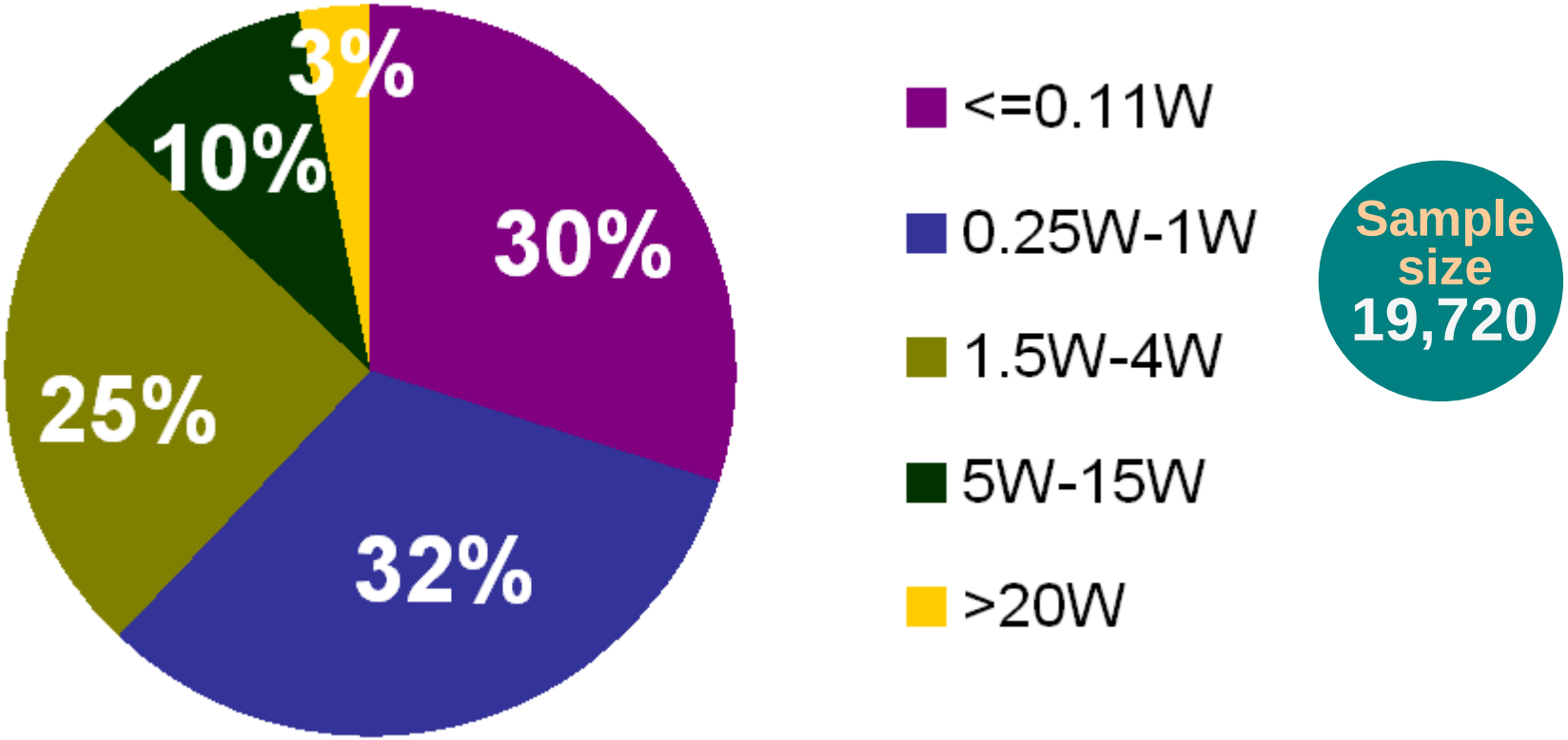
Why Power Management? (cont.)

- 19 % of total electricity worldwide consumed by electrical lightening
- 5 to 25% of power wasted by standby TV, PC, games, printers!!!
- 80% of IT power wasted (globally 100 Million MWH)

Why Power Management? (cont.)

- Low Voltage
  ⇨ Leakage + Noise
- Technology Shrink
  ⇨ Variability + Design Complexity
- High Density & High Frequency
  ⇨ High Temperature + EM

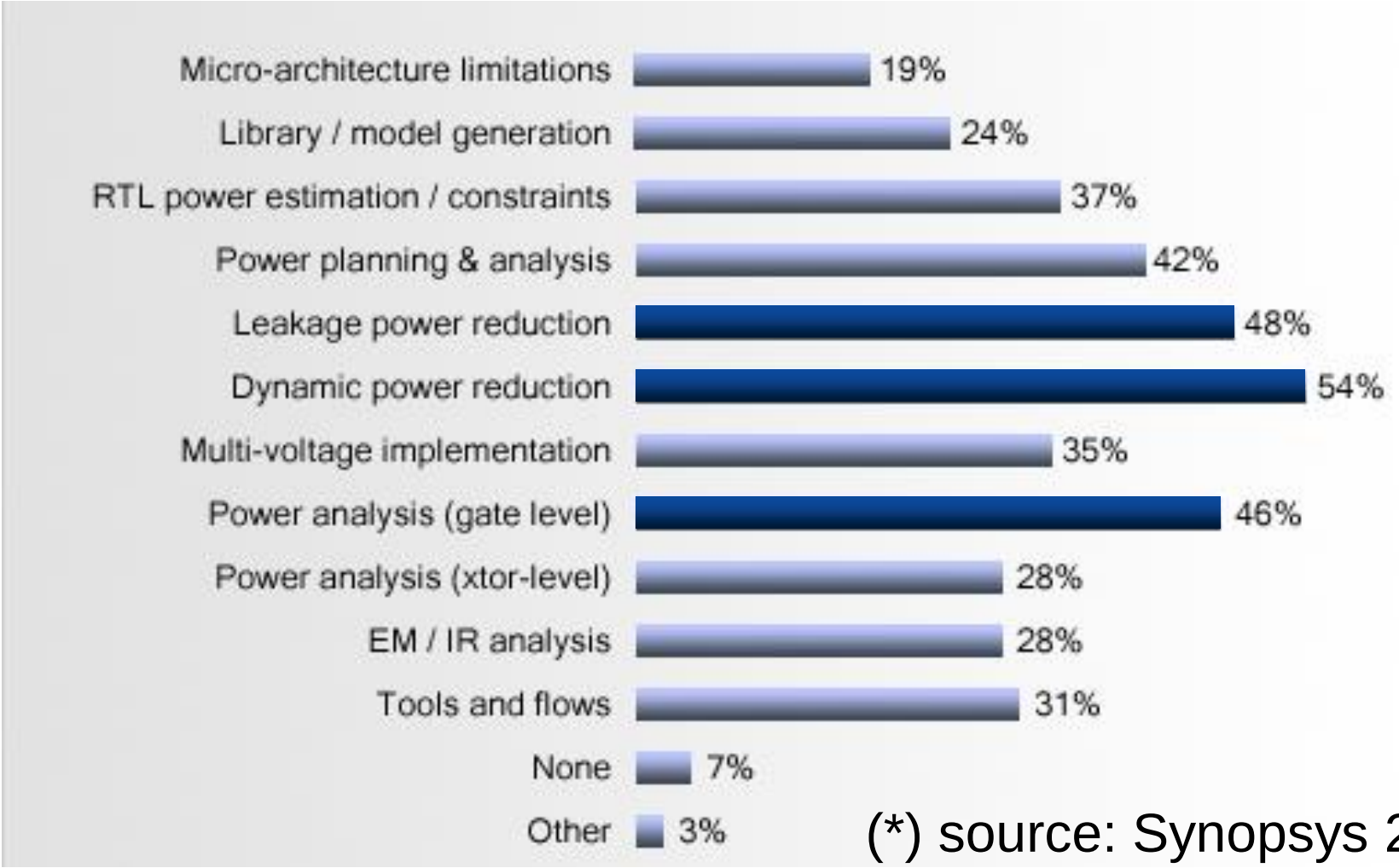
Why Power Management? (cont.)



Source: Chip Design Trends Newsletter, John Blyler, April 2007

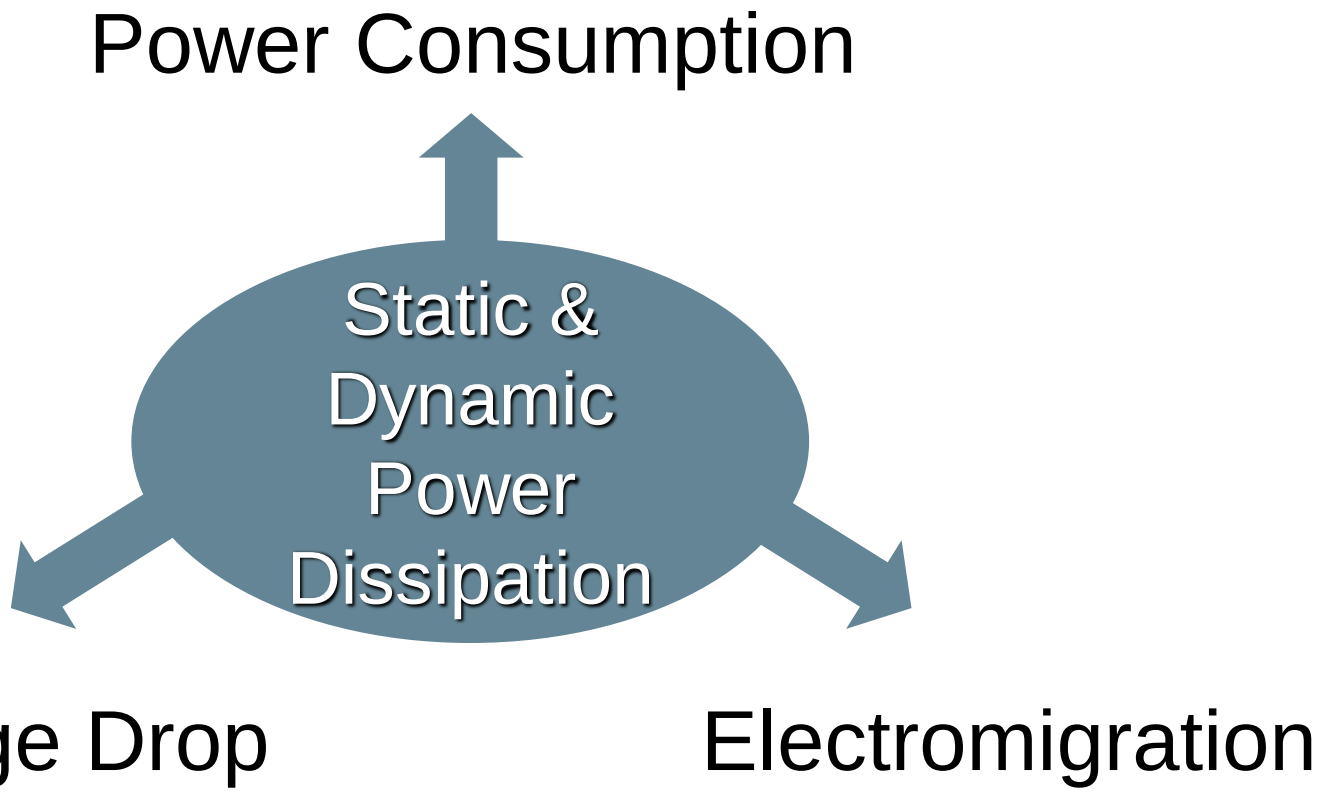
Why Power Management? (cont.)

- What Power-related issues did you encounter on last project?

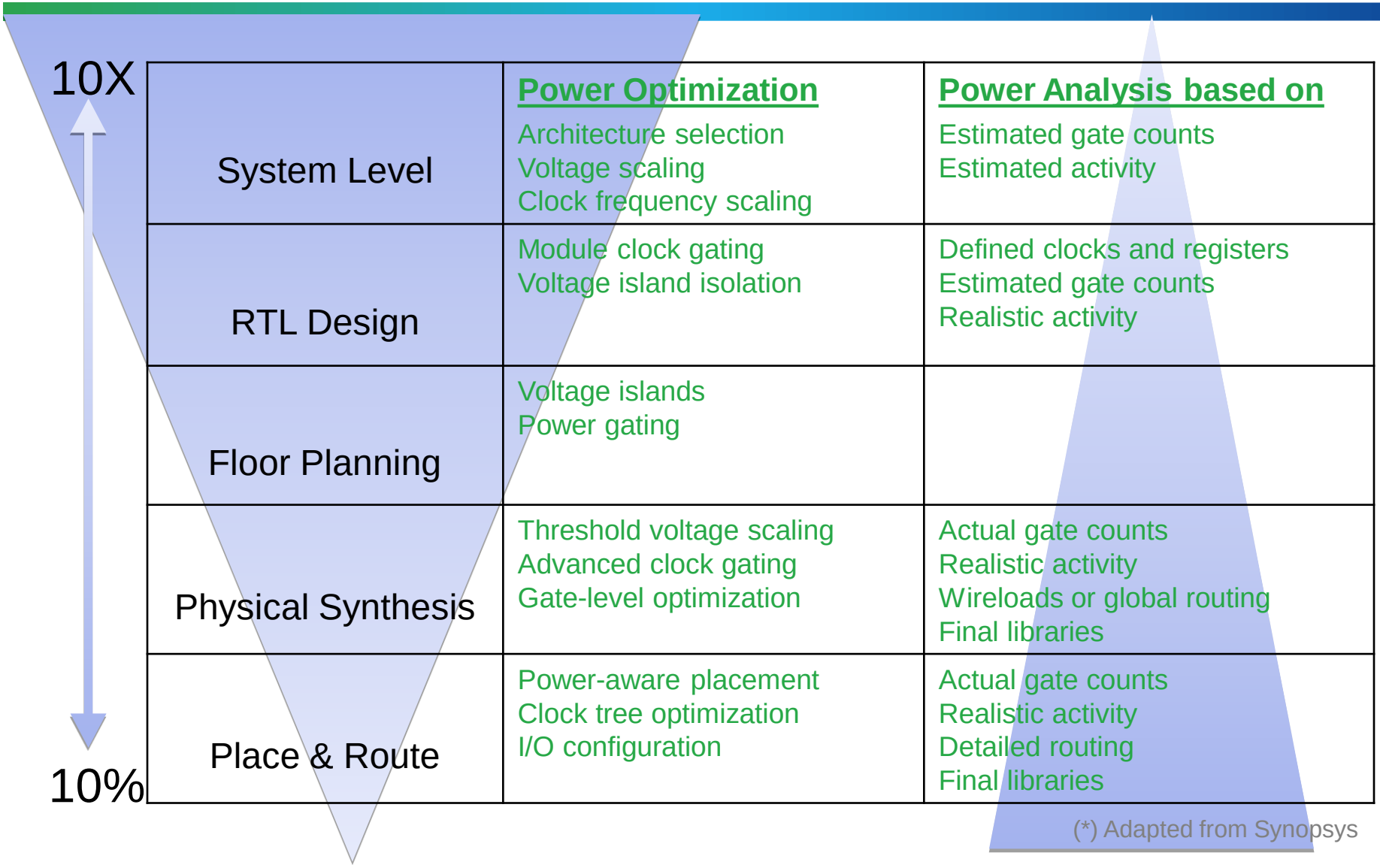


(*) source: Synopsys 2008

Why Power Management? (cont.)



Power Closure Challenges*



	Power Optimization	Power Analysis based on
System Level	Architecture selection Voltage scaling Clock frequency scaling	Estimated gate counts Estimated activity
RTL Design	Module clock gating Voltage island isolation	Defined clocks and registers Estimated gate counts Realistic activity
Floor Planning	Voltage islands Power gating	
Physical Synthesis	Threshold voltage scaling Advanced clock gating Gate-level optimization	Actual gate counts Realistic activity Wireloads or global routing Final libraries
Place & Route	Power-aware placement Clock tree optimization I/O configuration	Actual gate counts Realistic activity Detailed routing Final libraries

(*) Adapted from Synopsys

Power Closure Challenges

- Power metrics –what and when is it important?
- Power analysis accuracy and consistency
- Need for a combination of spatial and temporal information
- Good power vectors difficult to generate
- Power models complexity
  - IPs, operating modes, Temperature dependence of leakage
- Design & requirement complexity
  - Power, timing, area, cost, reliability

Power Optimization: System

Technique	Dynamic	Static
Clock optimization	√	
Parallelism / Pipelining	√ (2-3x)	X
Energy efficient SW & FW	√	√
Voltage & frequency scaling	√	√

- System level power breakdown
- Chip level power specification
- Hardware and firmware algorithm partition

Power Optimization: RTL

Technique	Dynamic	Static
Module clock gating	√	
Bus & State encoding	√	
Voltage & Frequency scaling	√	√
Retiming	√	
Power gating and sleep devices		
Voltage and power islands	√	√

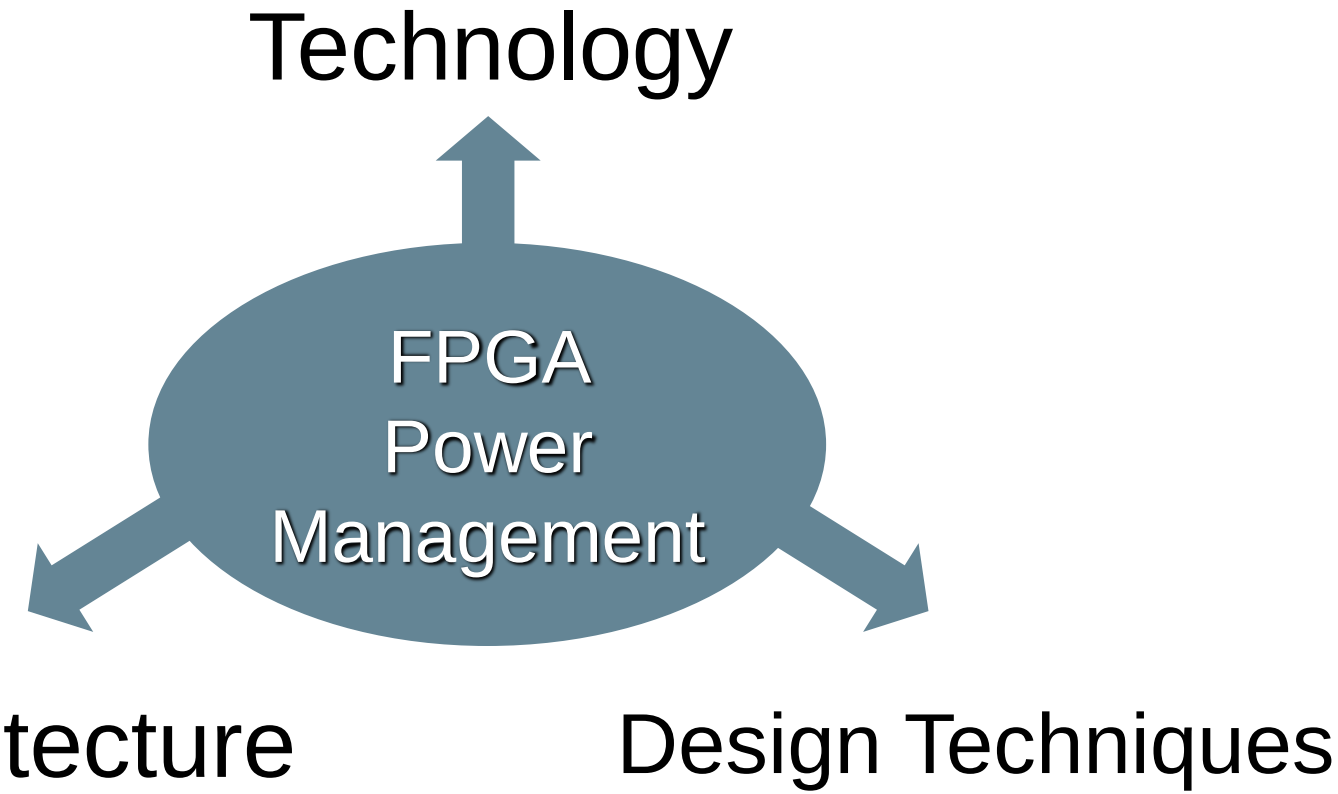
- Power constraints and per IP power specification
- RTL and IP power optimization
- power coverage

Power Optimization: Physical Design

Technique	Dynamic	Static
Clock optimization	√	
Activity aware P&R	√	
I/O optimization	√	
Voltage & Frequency scaling	√	√
Input state aware leakage		√

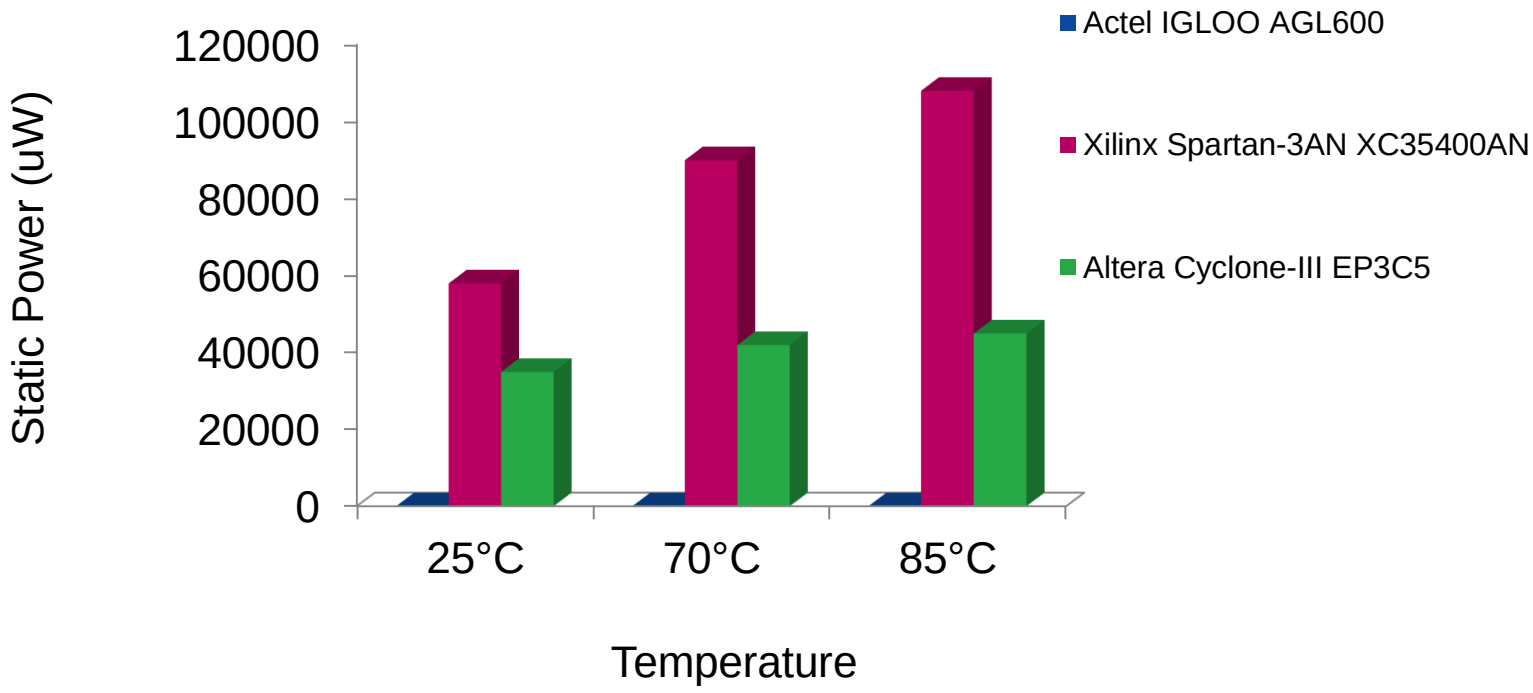
- power coverage

Microsemi Power Management



Microsemi Power Management: Technology

- Actel Flash FPGAs **> 1000 times** less static power
- Actel Flash FPGAs **competitive** for dynamic power
- RTAX-SL 50% lower standby current at 125 degree C

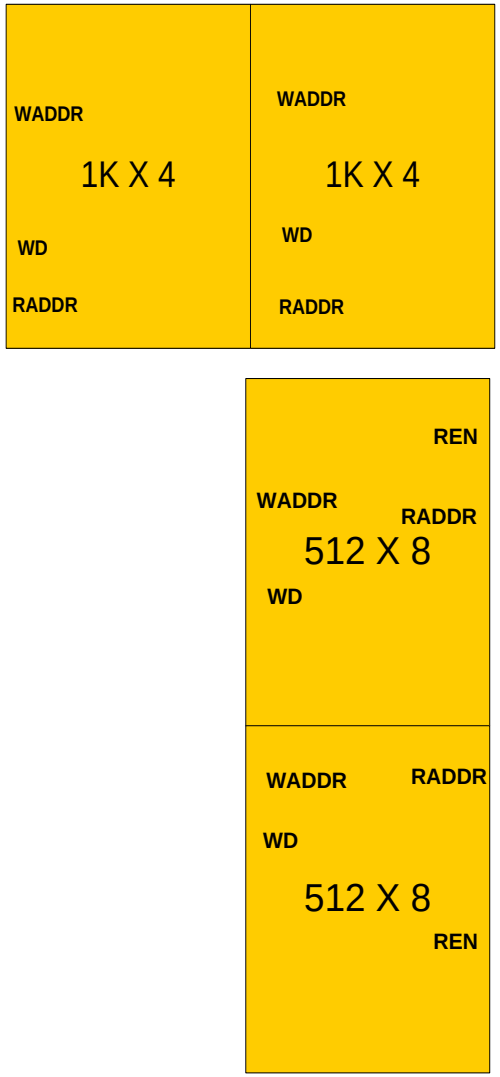


Microsemi Power Management: Architecture

- Low power macros
- Segmented clocks
- Low-power modes
- Multi-Voltage

Microsemi Power Management: Architecture

- Low power macros: memory
  - Width Cascading:
    No extra logic,
    Better performance
  - Depth Cascading:
    Extra decoding & muxing logic,
    Lower performance & higher area
- Synplicity uses depth
- > 50% power saving
- Low-power option in SmartGen

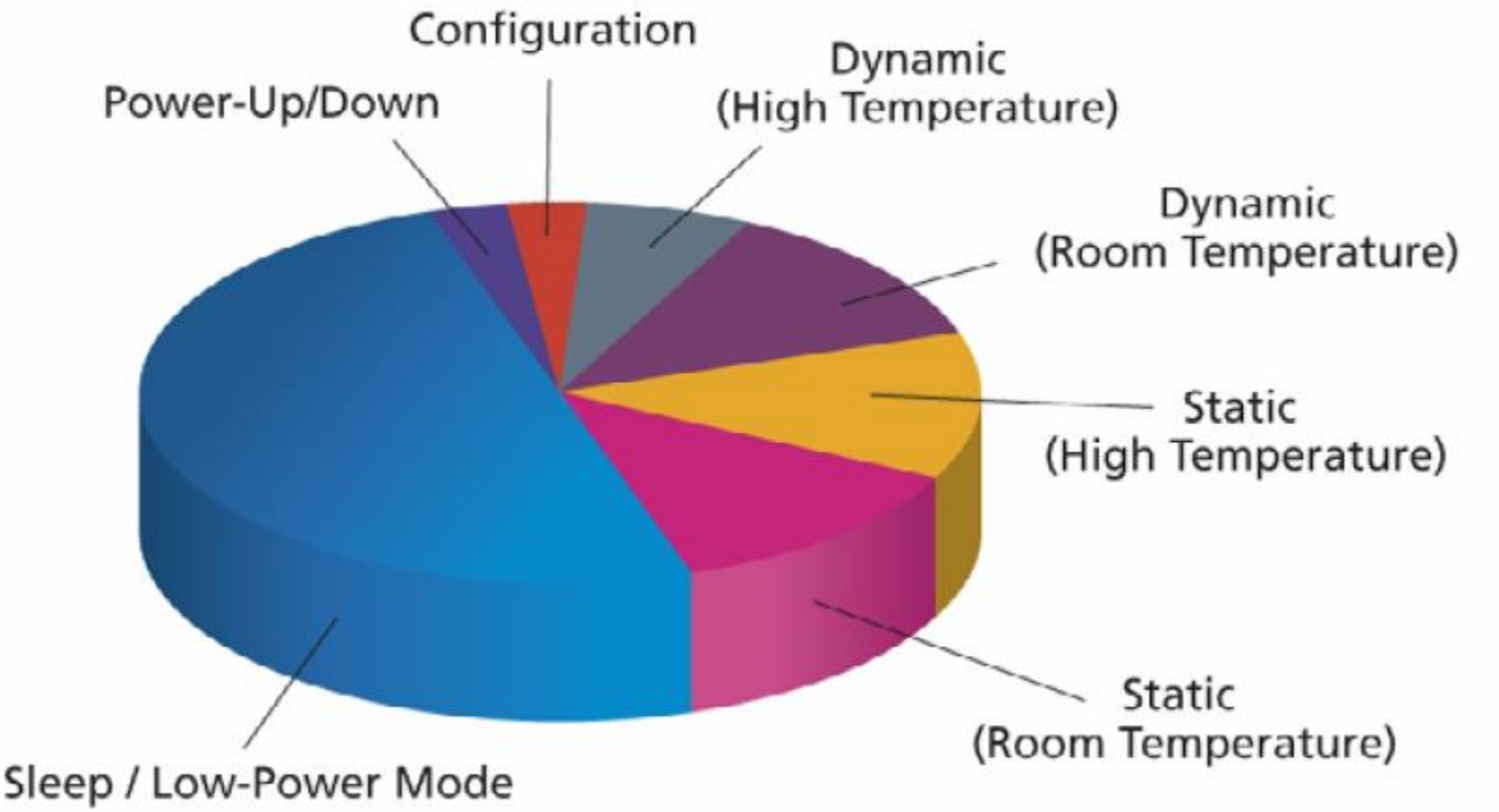


Microsemi Power Management: Architecture

- Low power macros: arithmetic
- SmartGen Ripple adder
  - Power **(-25%)** Performance **(-27%)**
- SmartGen Brent-Kung and Sklansky adders
  - Power **(-6 to -18%)** Comparable performance
- Experiment: Multipliers
  - Low-power multiplier with clock and signal gating
  - We could save up to **50%** in idle mode
  - Project on hold

Microsemi Power Management: Design Techniques

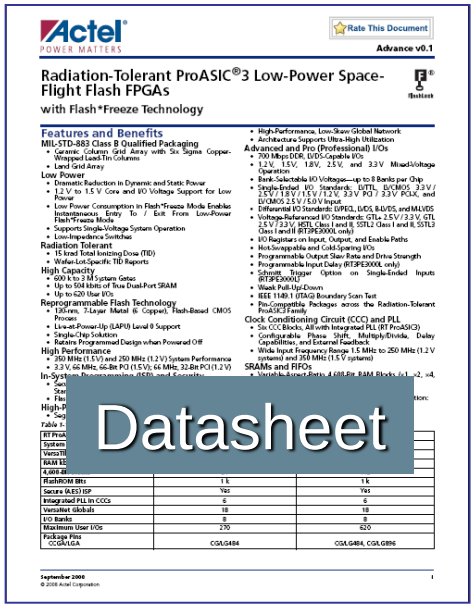
- Know your system power and temperature profile

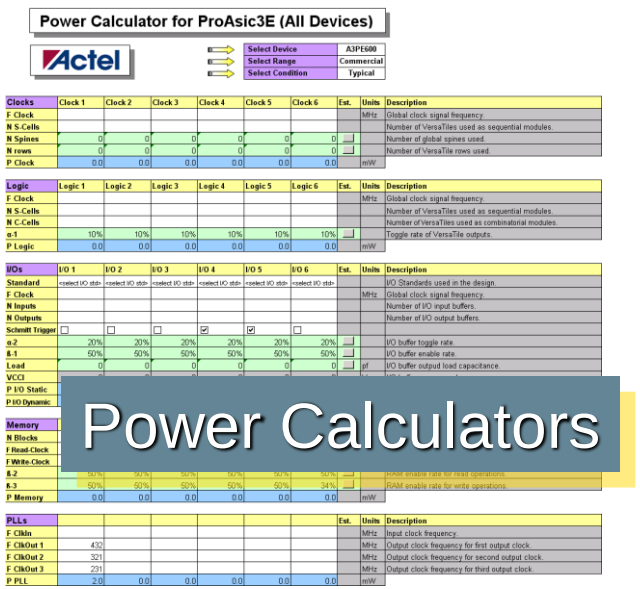


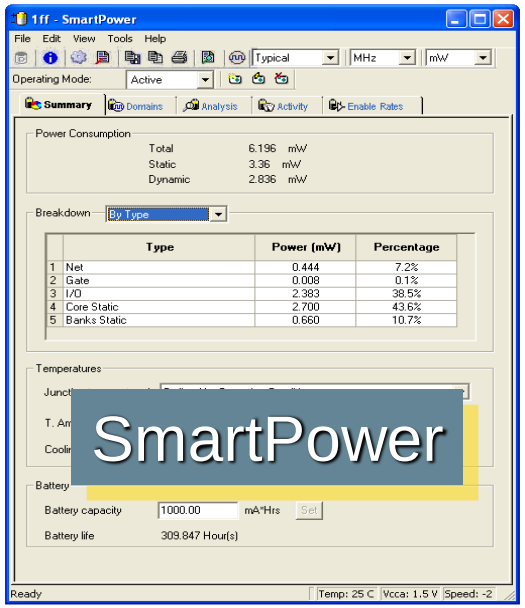
Microsemi Power Management: Analysis Tools

- Know your tools: Power Analysis

Design Flow



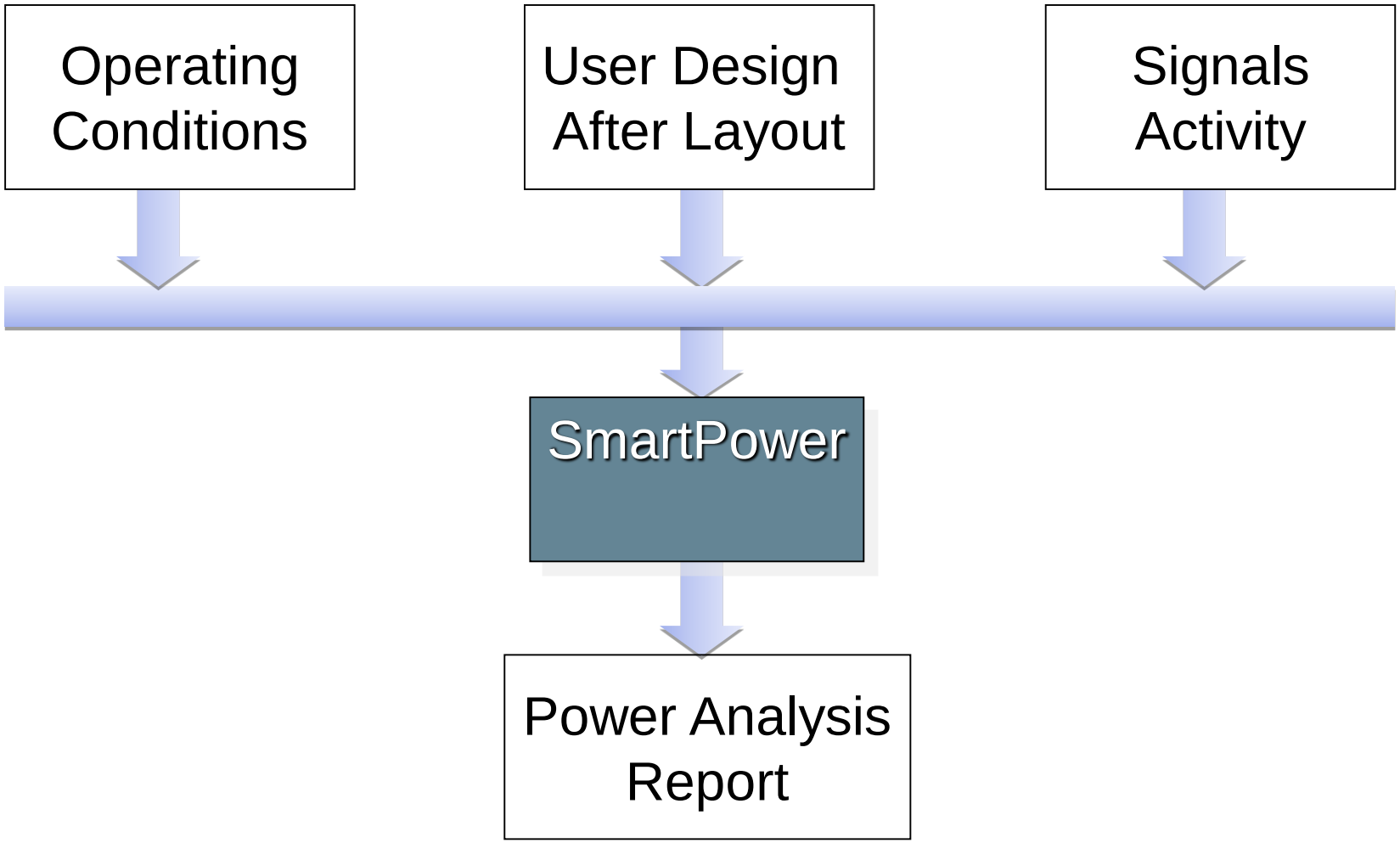




SmartPower: Power Analysis

- Average Analysis: Power budget for
  - Package Selection
  - Heat Dissipation
- Scenario Analysis: Capturing multi-functional power modes
- Glitch Analysis: Detecting power waste
- I/O Timing & Power Advisor
- Time Based Analysis: Peak power for
  - Power Supply Specification
  - Hot spot
  - Voltage drop

SmartPower: High Level Flow

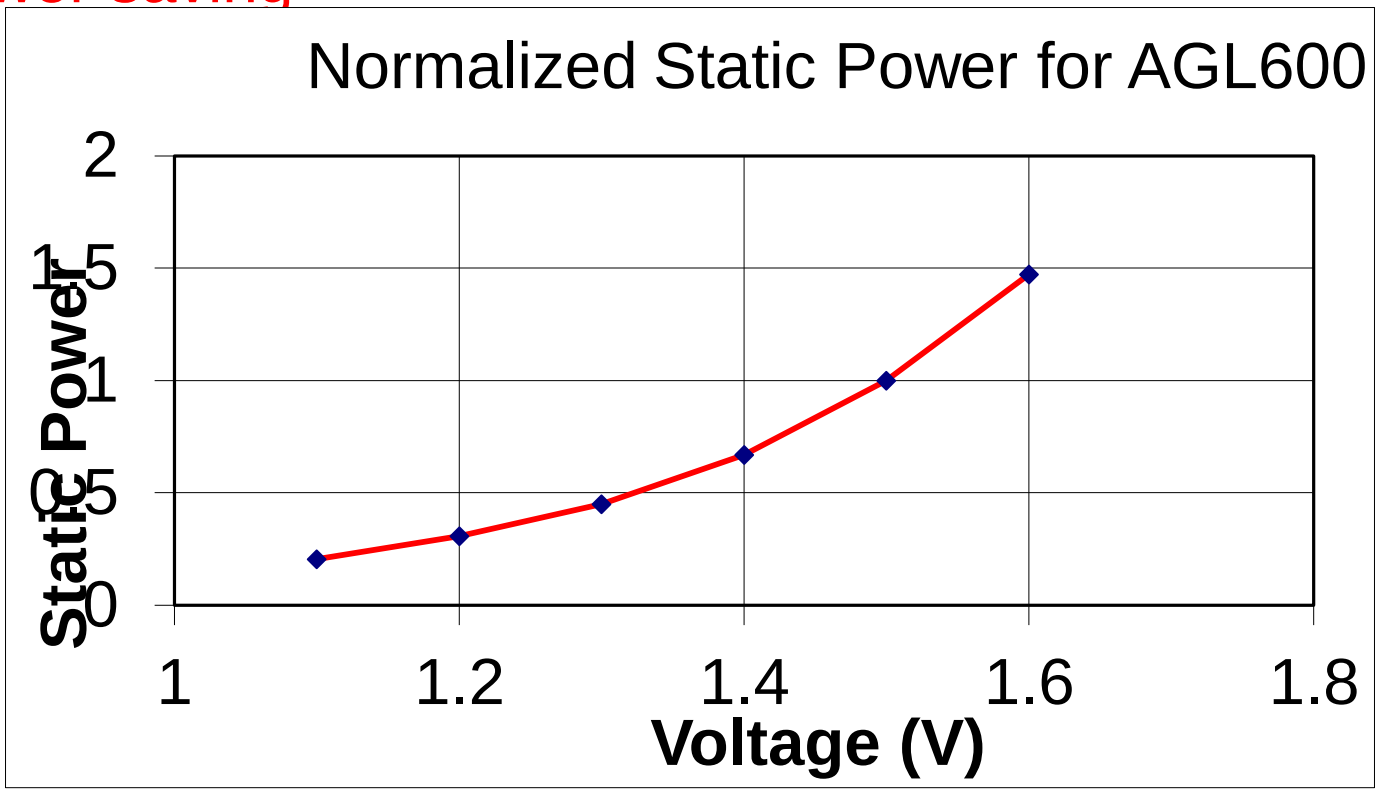


SmartPower: Operating Conditions

- Impact of Temperature:
  - High on Static, Limited on Dynamic Power
- Impact of Process:
  - High on Static, Moderate on Dynamic
- Impact of Voltage:
  - High on Static, High on Dynamic
- Impact of Radiation:
  - For RTAX-S, very small rise in ICC at 100Krad.
  - For RT-A3P, the TID reports show that this is also true for TID < 40Krad

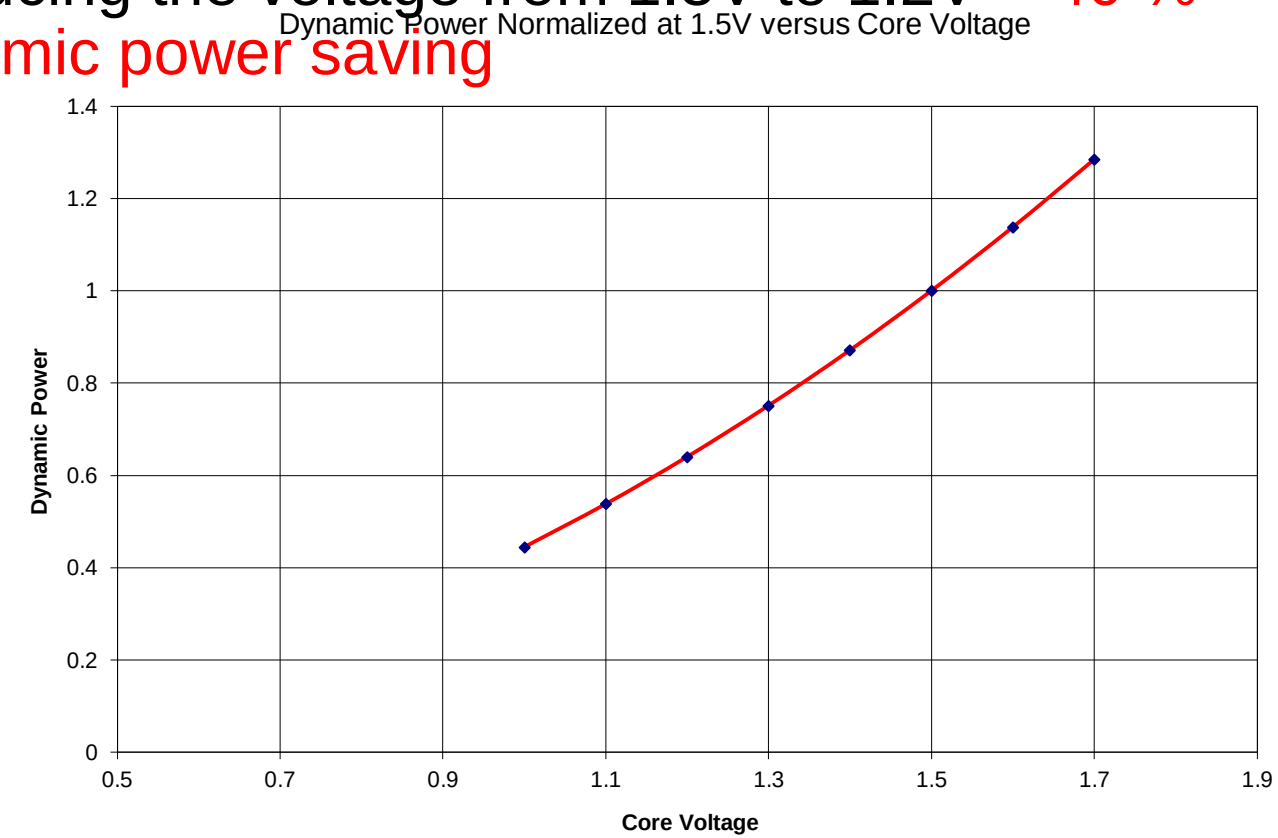
SmartPower: Operating Conditions (cont.)

- Impact of Voltage on Static Power
  - AGL600 Static ICC versus Voltage
  - Reducing the voltage from 1.5V to 1.2V = 70 % static power saving



SmartPower: Operating Conditions (cont.)

- Impact of Voltage on Dynamic Power
  - Power=C.V^2.Freq
  - Reducing the voltage from 1.5V to 1.2V = 40 % dynamic power saving

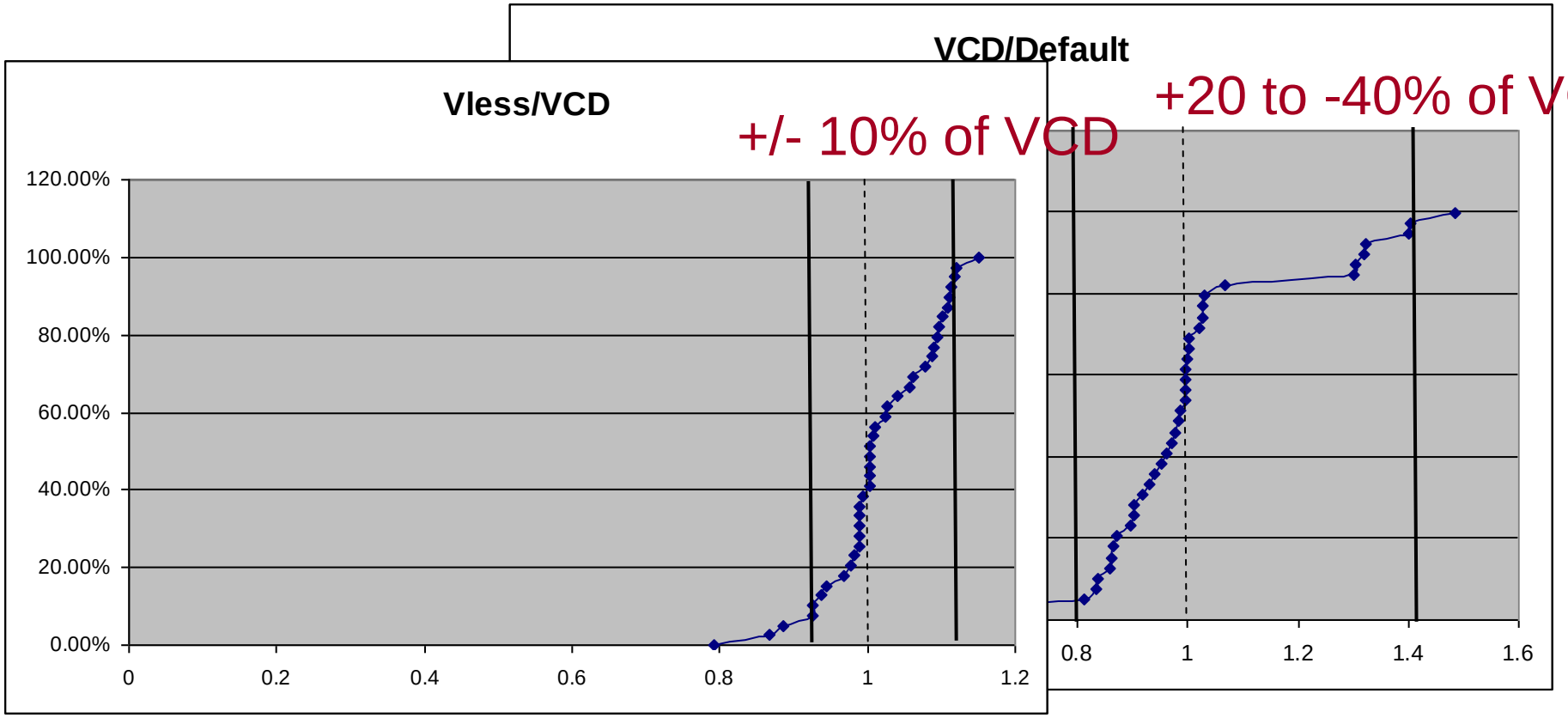


SmartPower: Signals Activity

- Through simulation data (VCD file)
  - Recommended flow
  - Simulation quality very important
- Actel's vectorless estimator
  - Clock constraints imported from SmartTime
- Default annotation for data and clocks
  - Two fixed values per clock domain: Clock frequency, data toggle rate
  - Convenient and fast - no simulation required – but inaccurate
- User specified net by net
  - Only useful to rectify specific nets activity

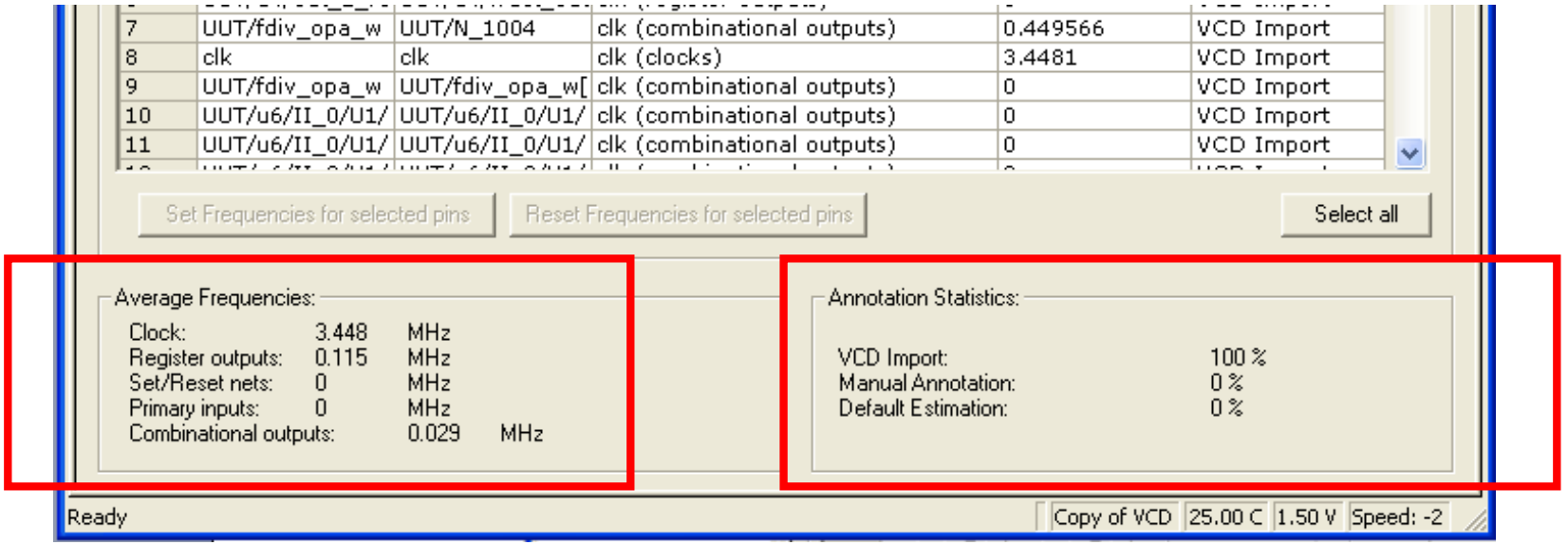
SmartPower: Signals Activity (cont.)

- Actel's vectorless estimator
  - Goal: Improve accuracy in the absence of simulation data (VCD)
  - Input: Probabilities and transition densities on primary inputs
  - Output: Switching activities on each pin



SmartPower: Signal Activities (cont.)

- Monitor the simulation coverage
  - If coverage < 95%, must revisit VCD flow



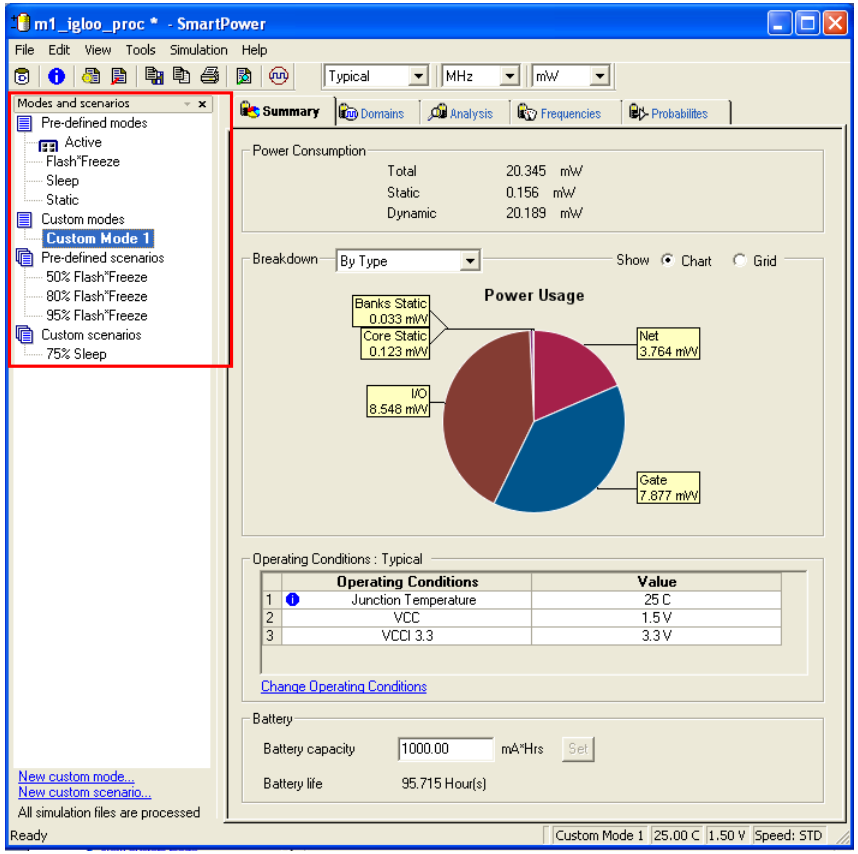
SmartPower: Modes and Scenarios

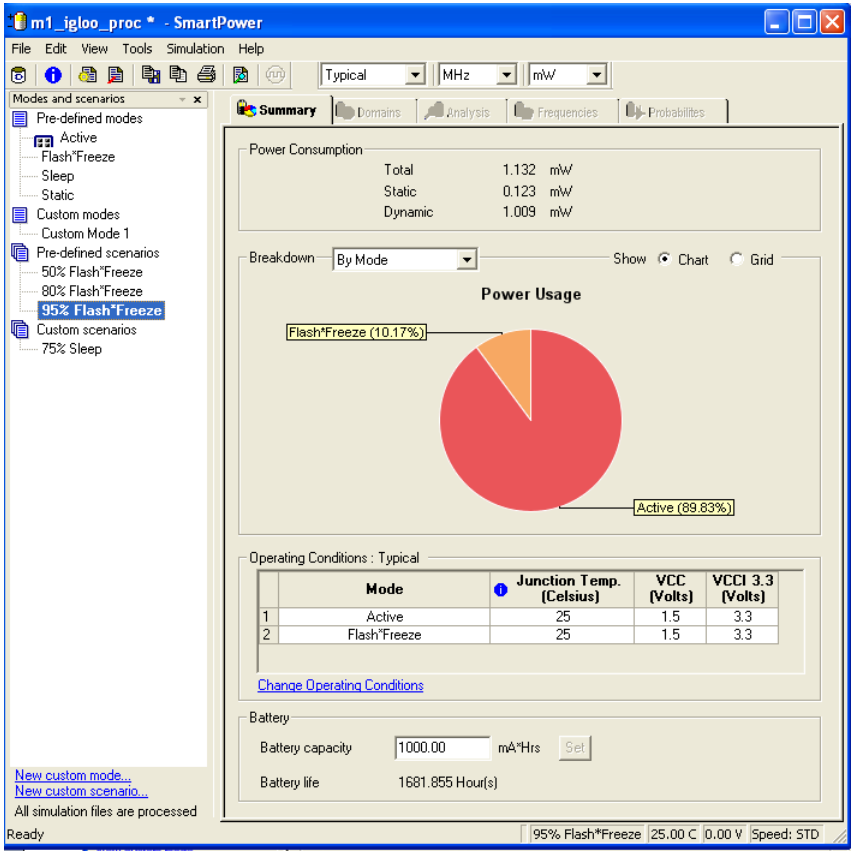
- Mode definition
  - To save a set of parameters defining the power of a design
  - To record specific operating conditions and activities
  - Predefined Modes: Flash*Freeze, Sleep, Stand-by (Fusion Only)

- Scenario definition
  - To combine different modes for power estimation
  - Predefined scenarios

SmartPower: Modes and Scenarios (cont.)

- SmartPower Modes and Scenarios Pane





SmartPower: Glitch Analysis

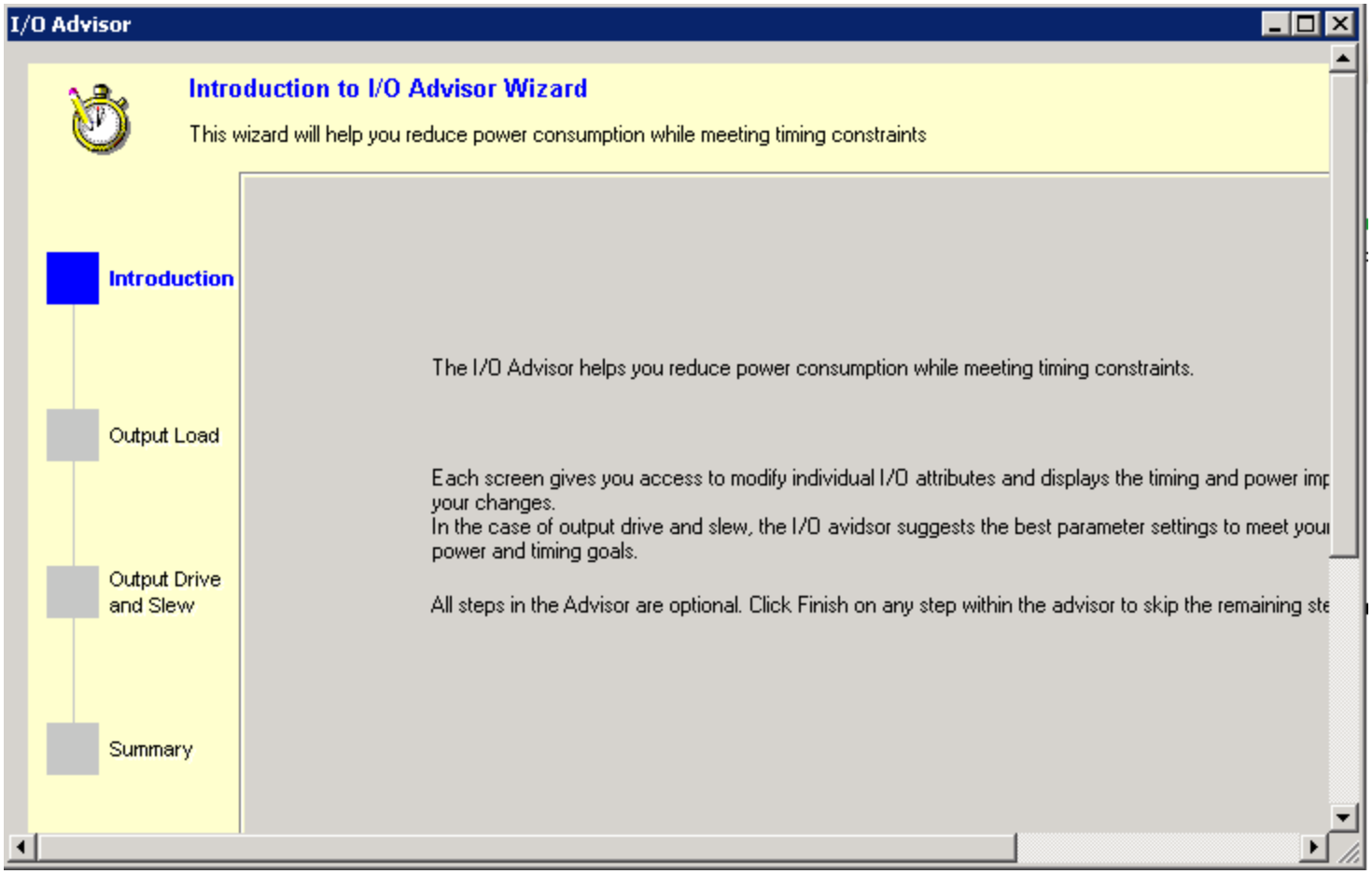
- Hazard or spurious transition definition: Due to delay mismatch among re-convergent paths.
- Wasted power represents 15%-20% of the global power
- Strongly dependent on circuit topology and test vectors
- Automatic glitch filtering when smaller than a given threshold, defined by family, and characterized by Spice
- A hazard report is accessible from SmartPower menu

SmartPower: I/O Advisor

- **Modification** of I/O attributes
  - Output load
  - Output drive and slew
- Algorithm to optimize power while meeting timing constraints – for output drive and slew
  - Positive slack – **selects** attributes with least power but still maintain positive slack
  - Negative slack – **selects** attributes to minimize negative slack
  - No slack – **selects** attributes with least power
- **Silicon family support**
  - G3 derivatives (8.6 release)

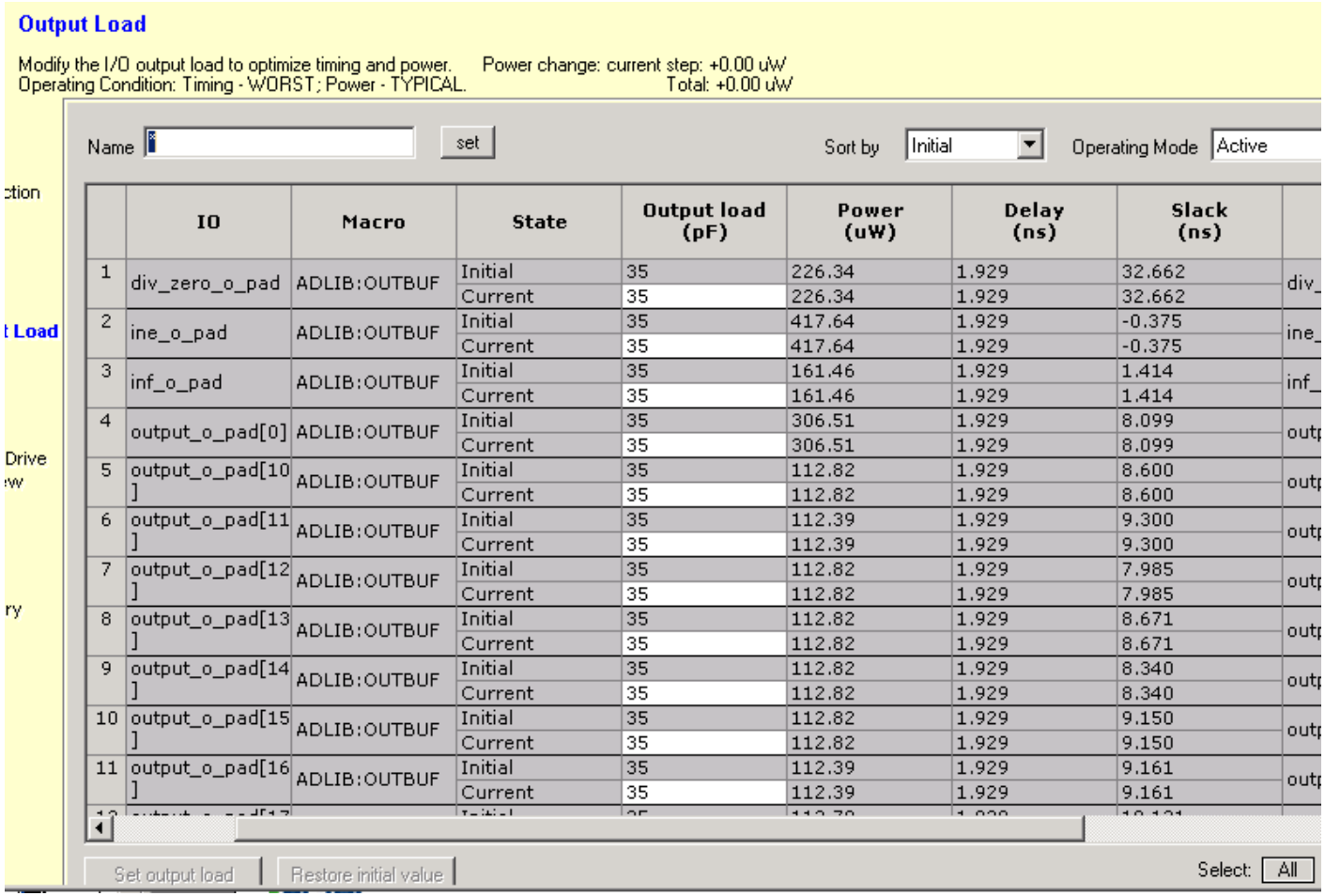
SmartPower: I/O Advisor (cont.)

- Introduction page
- Individual steps optional



SmartPower: I/O Advisor (cont.)

- Output load page
  - Change **current** output load to reduce power



Output Load

Modify the I/O output load to optimize timing and power. Power change: current step: +0.00 uW
Operating Condition: Timing - WORST; Power - TYPICAL. Total: +0.00 uW

Name: set | Sort by: Initial | Operating Mode: Active

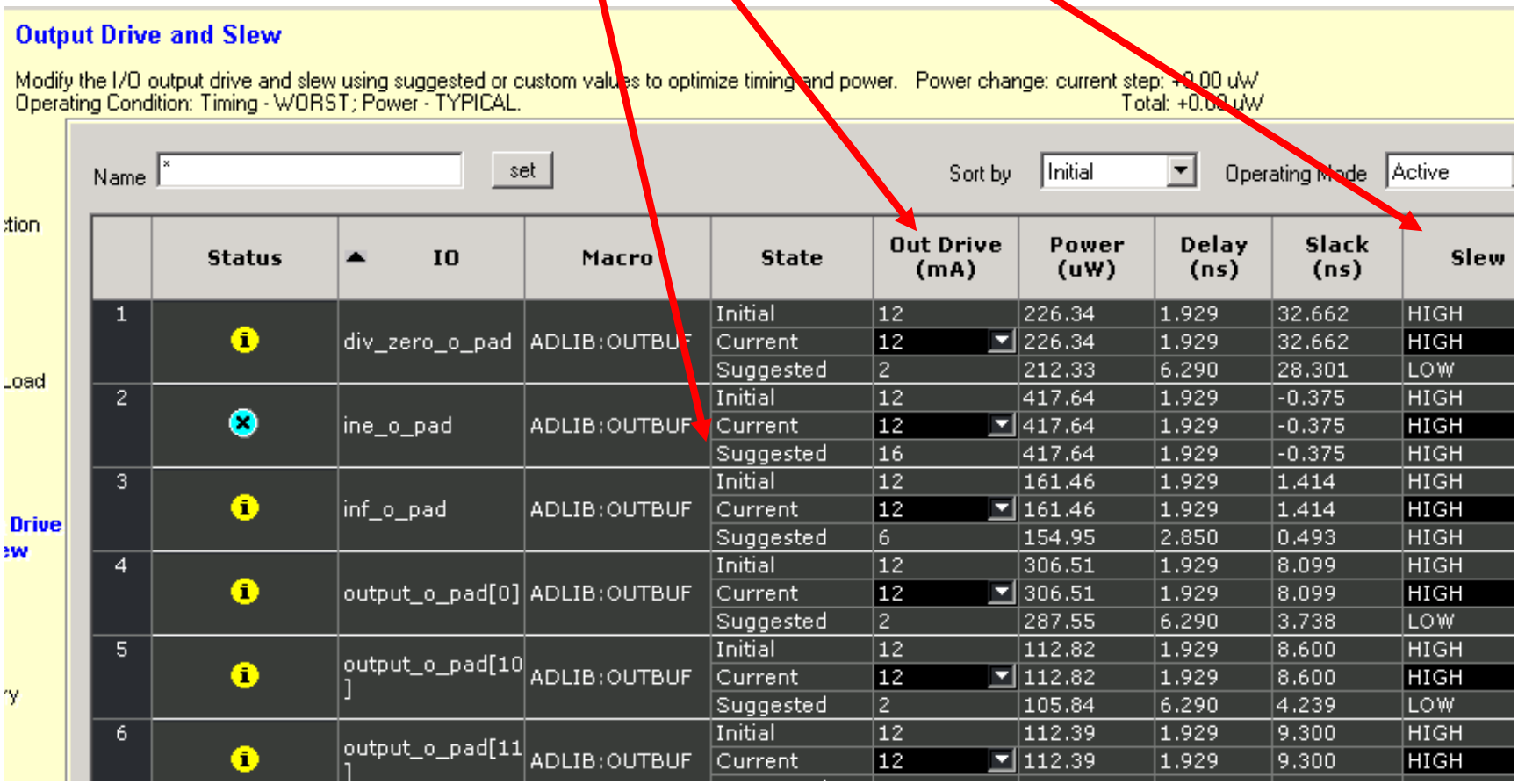
	IO	Macro	State	Output load (pF)	Power (uW)	Delay (ns)	Slack (ns)	
1	div_zero_o_pad	ADLIB:OUTBUF	Initial	35	226.34	1.929	32.662	div_
			Current	35	226.34	1.929	32.662	
2	ine_o_pad	ADLIB:OUTBUF	Initial	35	417.64	1.929	-0.375	ine_
			Current	35	417.64	1.929	-0.375	
3	inf_o_pad	ADLIB:OUTBUF	Initial	35	161.46	1.929	1.414	inf_
			Current	35	161.46	1.929	1.414	
4	output_o_pad[0]	ADLIB:OUTBUF	Initial	35	306.51	1.929	8.099	outp
			Current	35	306.51	1.929	8.099	
5	output_o_pad[10]	ADLIB:OUTBUF	Initial	35	112.82	1.929	8.600	outp
			Current	35	112.82	1.929	8.600	
6	output_o_pad[11]	ADLIB:OUTBUF	Initial	35	112.39	1.929	9.300	outp
			Current	35	112.39	1.929	9.300	
7	output_o_pad[12]	ADLIB:OUTBUF	Initial	35	112.82	1.929	7.985	outp
			Current	35	112.82	1.929	7.985	
8	output_o_pad[13]	ADLIB:OUTBUF	Initial	35	112.82	1.929	8.671	outp
			Current	35	112.82	1.929	8.671	
9	output_o_pad[14]	ADLIB:OUTBUF	Initial	35	112.82	1.929	8.340	outp
			Current	35	112.82	1.929	8.340	
10	output_o_pad[15]	ADLIB:OUTBUF	Initial	35	112.82	1.929	9.150	outp
			Current	35	112.82	1.929	9.150	
11	output_o_pad[16]	ADLIB:OUTBUF	Initial	35	112.39	1.929	9.161	outp
			Current	35	112.39	1.929	9.161	

Set output load | Restore initial value | Select: All

SmartPower: I/O Advisor (cont.)

Output drive and slew page

- User can change “current” output drive and slew
- **I/O Advisor provides** “suggestion” for better power consumption while meeting timing constraints



Output Drive and Slew

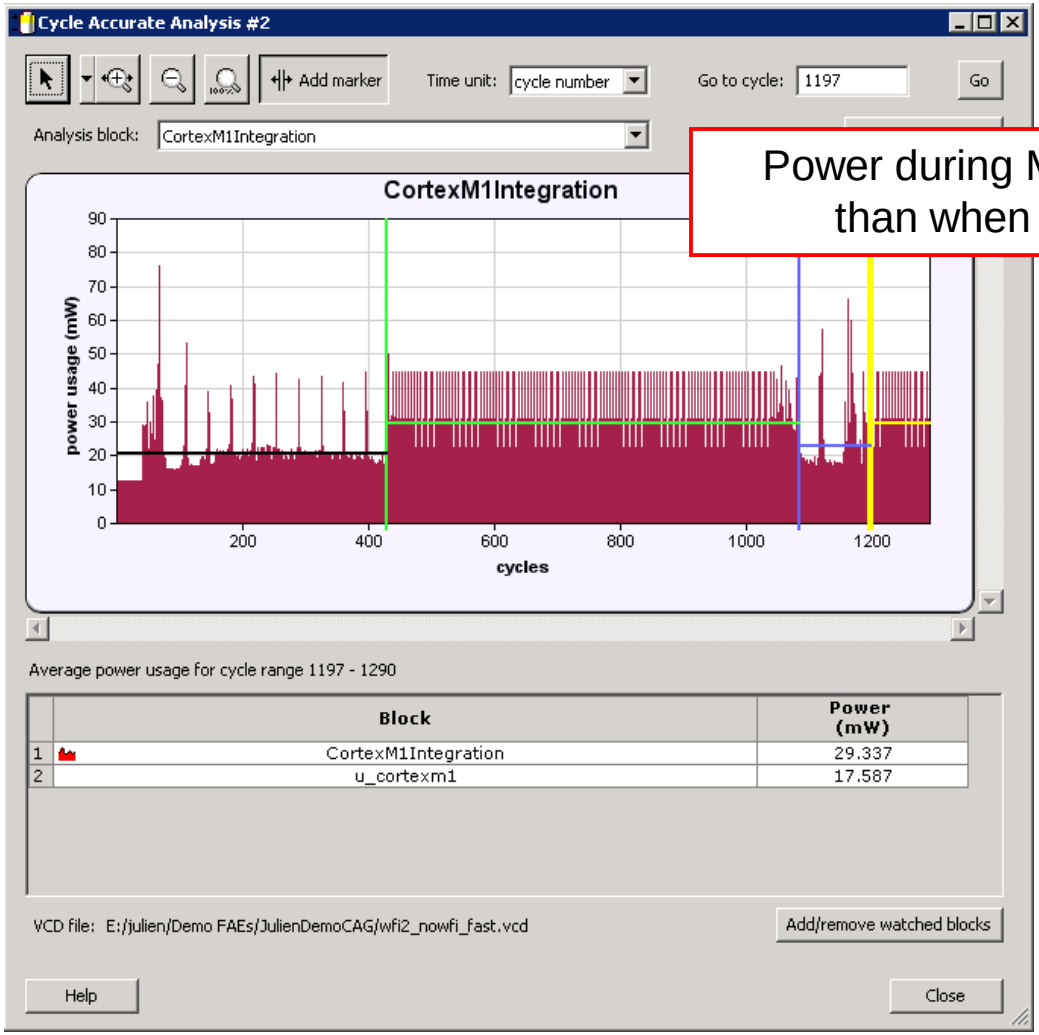
Modify the I/O output drive and slew using suggested or custom values to optimize timing and power. Power change: current step: +0.00 uW
Operating Condition: Timing - WORST; Power - TYPICAL. Total: +0.00 uW

Name * set Sort by Initial Operating Mode Active

	Status	IO	Macro	State	Out Drive (mA)	Power (uW)	Delay (ns)	Slack (ns)	Slew
1		div_zero_o_pad	ADLIB:OUTBUF	Initial	12	226.34	1.929	32.662	HIGH
				Current	12	226.34	1.929	32.662	HIGH
				Suggested	2	212.33	6.290	28.301	LOW
2		ine_o_pad	ADLIB:OUTBUF	Initial	12	417.64	1.929	-0.375	HIGH
				Current	12	417.64	1.929	-0.375	HIGH
				Suggested	16	417.64	1.929	-0.375	HIGH
3		inf_o_pad	ADLIB:OUTBUF	Initial	12	161.46	1.929	1.414	HIGH
				Current	12	161.46	1.929	1.414	HIGH
				Suggested	6	154.95	2.850	0.493	HIGH
4		output_o_pad[0]	ADLIB:OUTBUF	Initial	12	306.51	1.929	8.099	HIGH
				Current	12	306.51	1.929	8.099	HIGH
				Suggested	2	287.55	6.290	3.738	LOW
5		output_o_pad[10]	ADLIB:OUTBUF	Initial	12	112.82	1.929	8.600	HIGH
				Current	12	112.82	1.929	8.600	HIGH
				Suggested	2	105.84	6.290	4.239	LOW
6		output_o_pad[11]	ADLIB:OUTBUF	Initial	12	112.39	1.929	9.300	HIGH
				Current	12	112.39	1.929	9.300	HIGH

SmartPower: Time Based Analysis

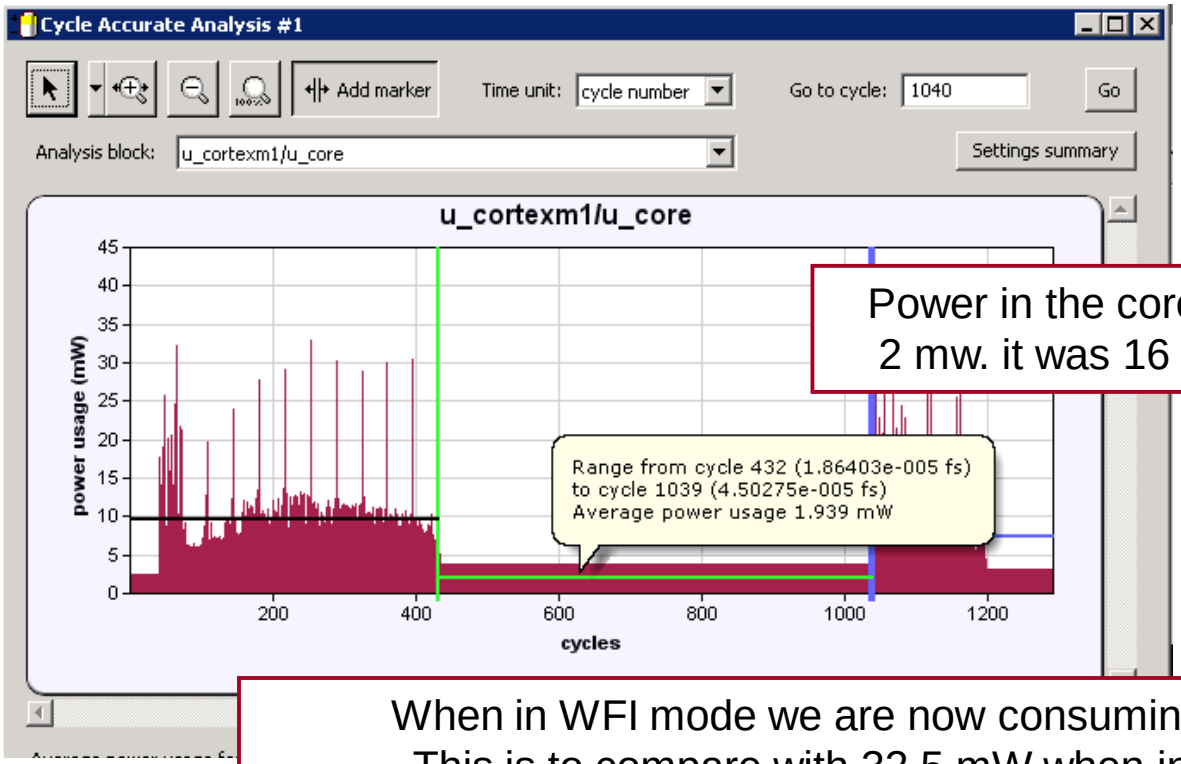
- Using Cycle Accurate Analysis to debug a simulation time window



Power during M1/WFI is higher than when not in WFI !!!

SmartPower: Time Based Analysis (cont.)

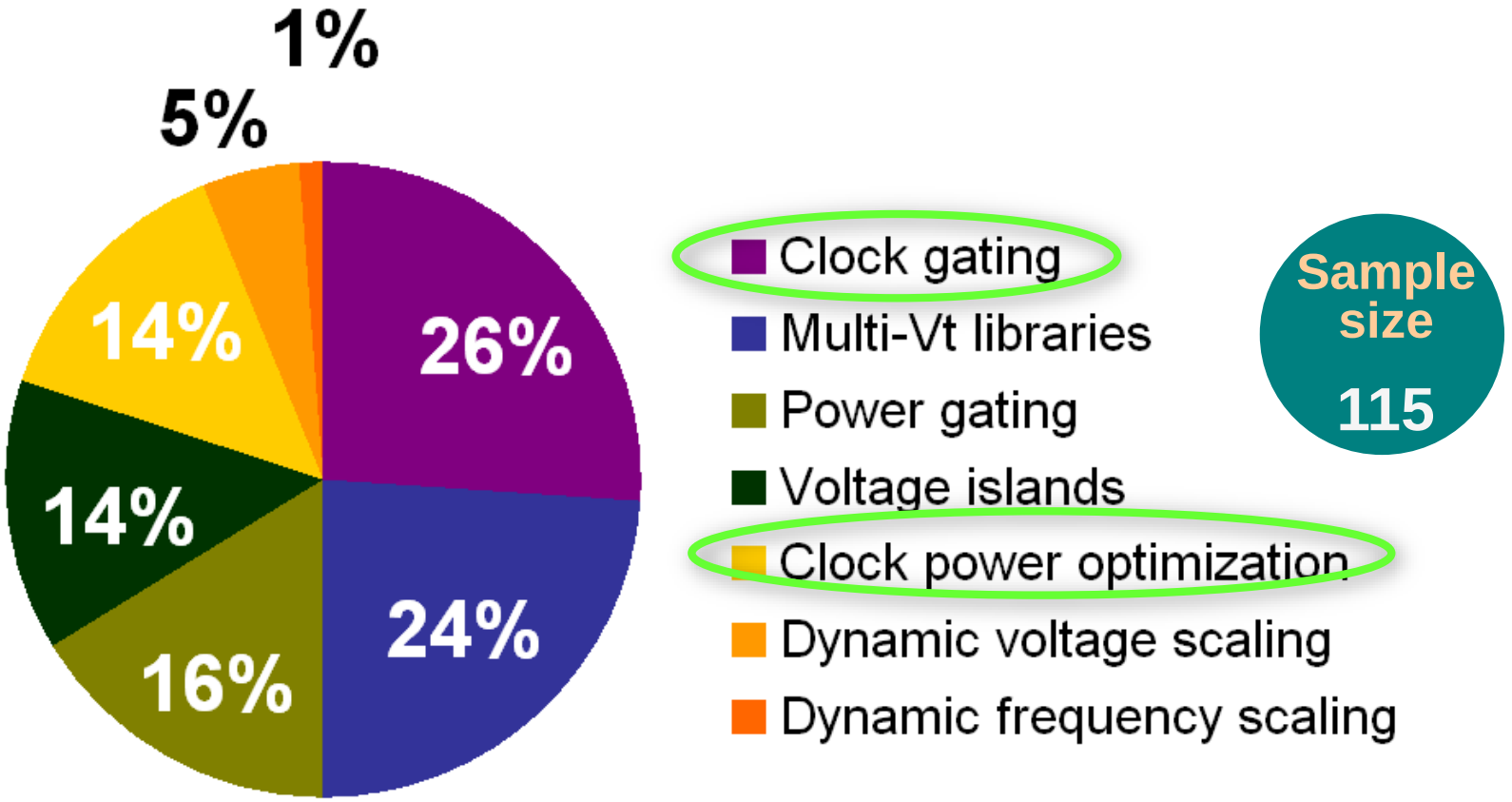
- Verifying that the Gated Clock solved the problem



Power in the core during WFI is less than 2 mw. it was 16 mW without gated clock

When in WFI mode we are now consuming only 13 mw. This is to compare with 32.5 mW when in Burst Mode

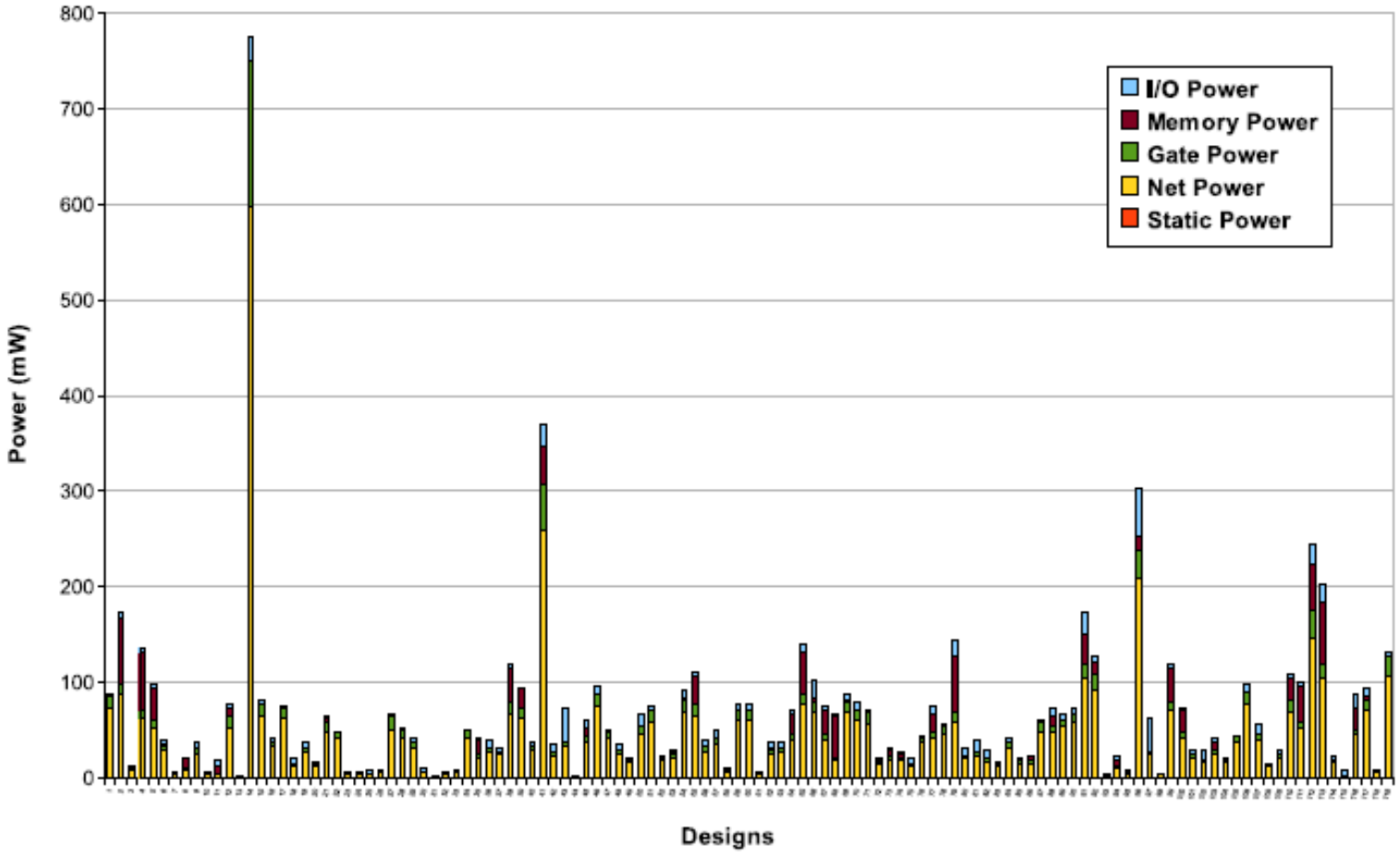
PDPR: Global Routing Optimization



Survey end users at DAC Suite 2006
Source Sequence Design inc.

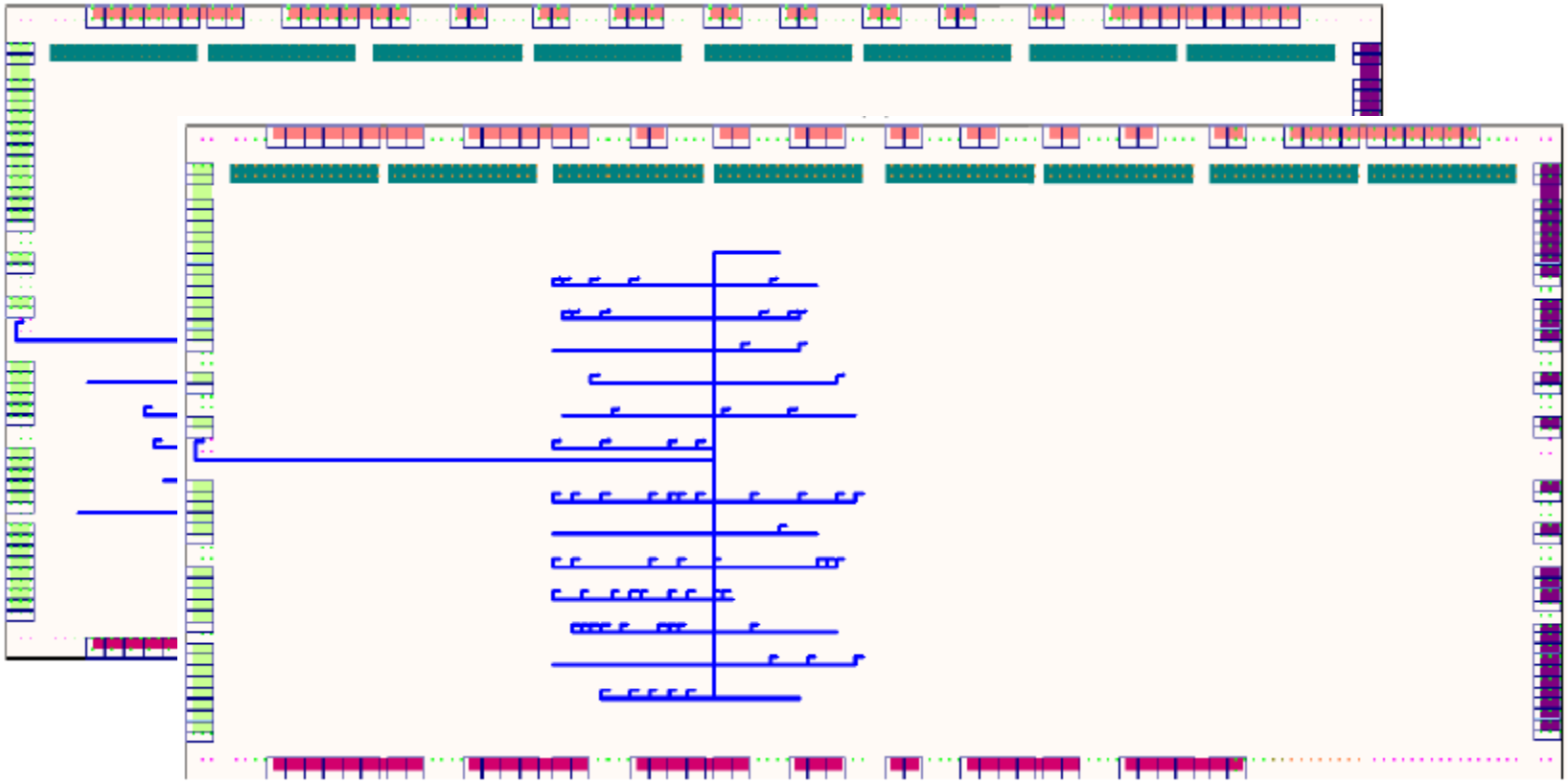
PDPR: Global Routing Optimization

- For designs without memories on IGLOO 1.2V devices
  - 65% of total power (Nets); 12% (Gates); 13% (I/Os)



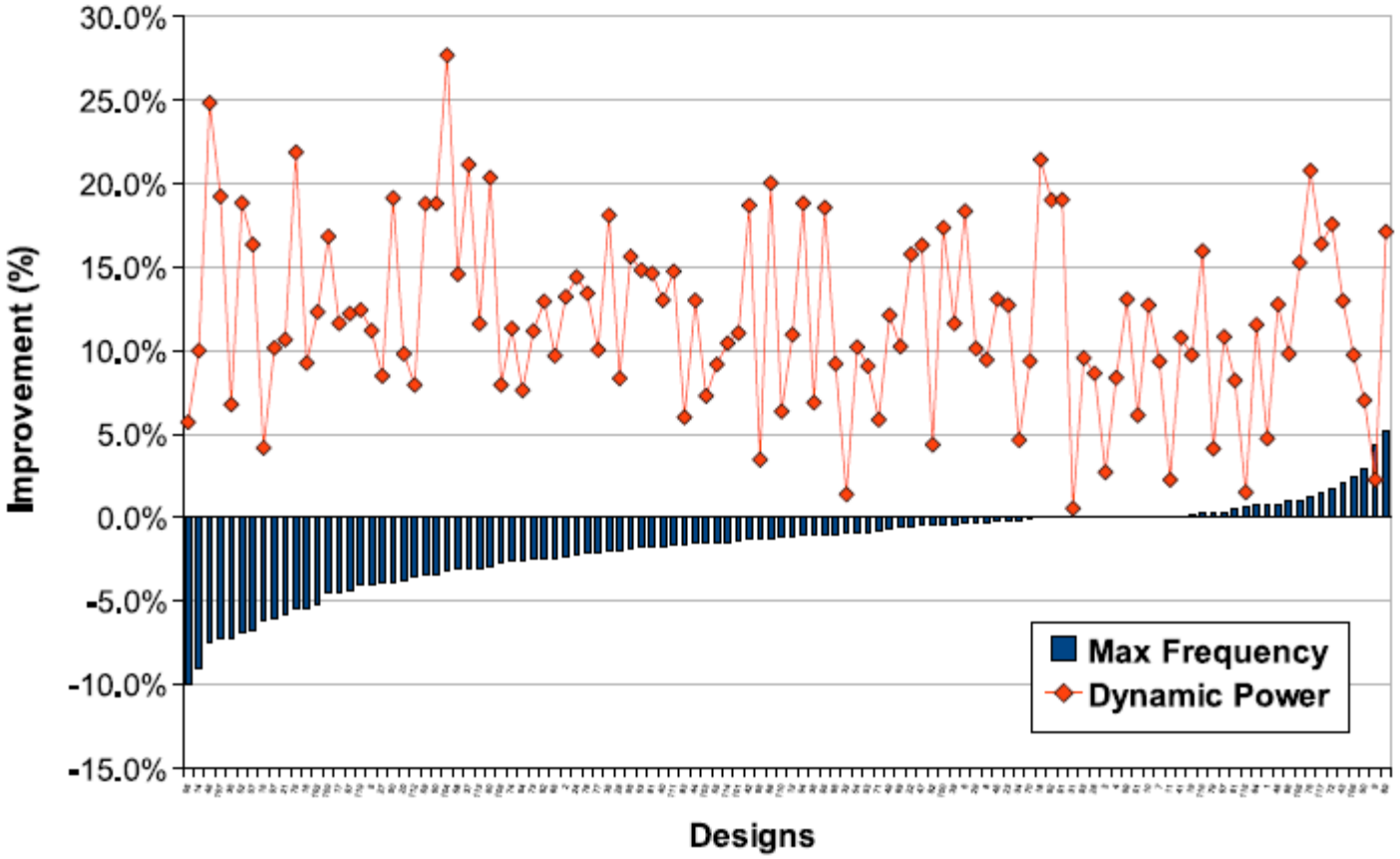
PDPR: Global Routing Optimization

- Reducing the global network segments (spines) during placement
- From 7 spines and 60 ribs to 2 spines and 26 ribs.



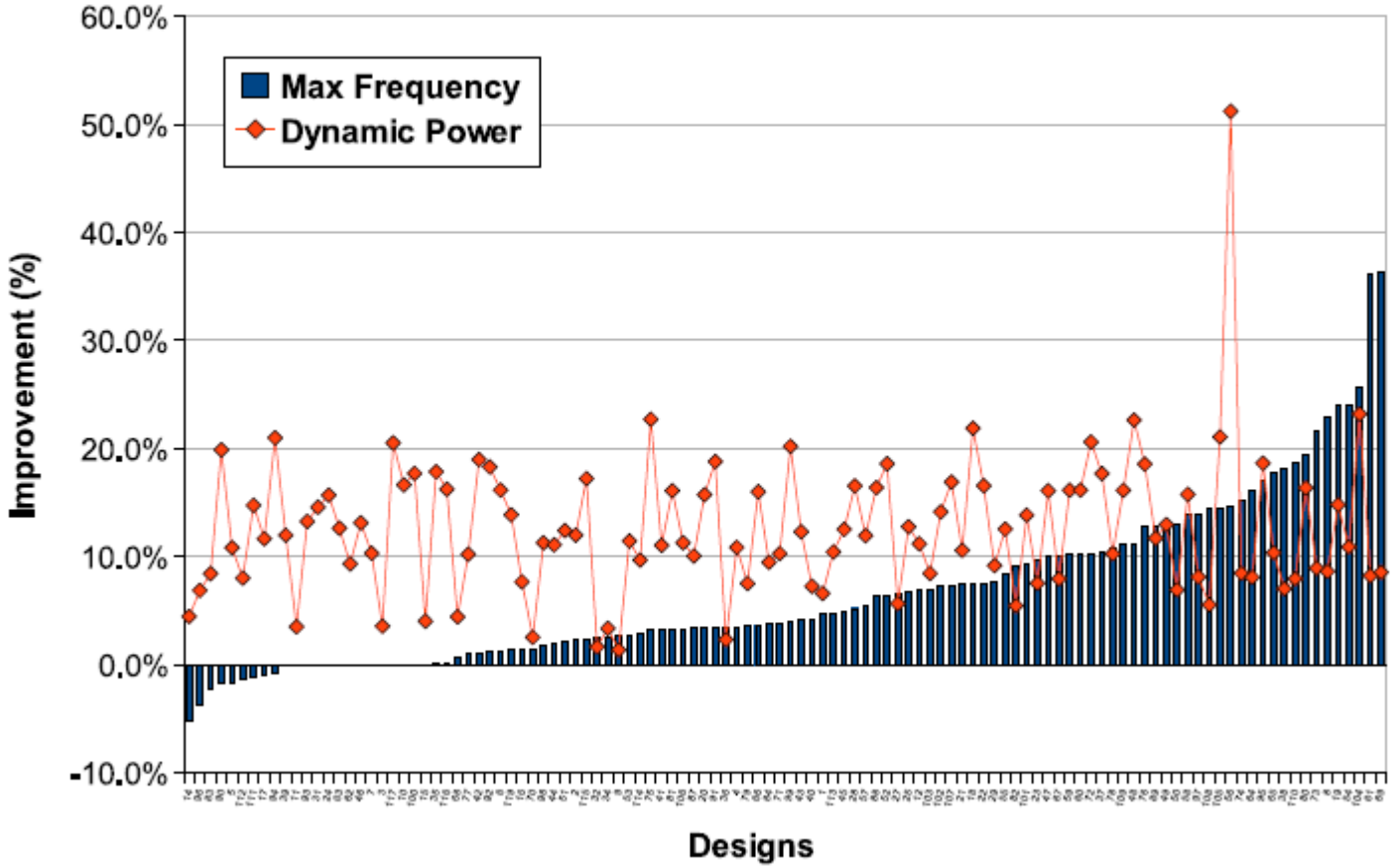
PDPR: Global Routing Optimization

- 12% average (28% max) reduction in overall dynamic power
- 18% average (37% max) reduction in net power
- ~1% performance loss



PDPR: Power-Driven Re-Synthesis

- Area minimization with sequential optimization
- 6.7% average timing improvement
- 13% average power saving



Conclusion: Best Design Practices

- Architectural exploration has great impact
- Write power friendly RTL
- Clock reduction scheme is very important
- Power-efficient memory selection is key
- Low-power arithmetic macros can be helpful
- Develop accurate power vectors
- Verify power early and often
- Run “power regressions” throughout – RTL to tapeout

Questions ?