DATE 16

DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

Modern Clocking Strategies

Organisers:

Jordi Cortadella, Universitat Politècnica de Catalunya
Luciano Lavagno, Politecnico di Torino
Alex Yakovlev, University of Newcastle

Speakers:

Koen van Eijk, Synopsys
Alex Yakovlev, University of Newcastle
David M. Zar, Blendics
Jordi Cortadella, Universitat Politècnica de Catalunya

1 Overview

Clock network design and timing analysis are among the most challenging tasks in integrated circuit design. The former also exhibits the broadest range of very different solutions, ranging from classical zero-skew clocking, to multiple independent clock islands, each operating at a different Dynamic Voltage and Frequency Scaling (DVFS) point, and to clocks that dynamically adapt to the timing characteristics of the underlying logic. Of course, every clocking strategy must be supported by a corresponding verification method on the static timing analysis side. This can be particularly tricky when the clocks become less and less synchronous, due to power management methods or to techniques that improve robustness with respect to variability.

This tutorial is aimed at covering the full range of synthesis and verification tasks for the clocks, starting from basic definitions and techniques, and then gradually expanding the horizon.

Each talk will be offered by a leading industrial or academic expert, and will enable designers to choose the best synchronization technique for the problem at hand.

Attendees are assumed to know about sequential logic design and basic single-phase zero-skew clocking. They will gain knowledge about the following topics: (1) clock synthesis and timing analysis, especially in conjunction with power management techniques such as clock gating, power gating and DVFS; (2) reliability analysis in the presence of meta-stability for clock domain crossing; (3) asynchronous synchronization techniques; (4) advanced adaptive clocking strategies, where the clock latency or period adapts to the operating conditions of the logic.

2 Lectures

The tutorial will be organized in four lectures, each one covering different aspects of clock synthesis and timing. Next, an abstract of each lecture is presented.

Clock Synthesis and Chip Variability (Koen van Eijk, Synopsys)

One of the main challenges for clock synthesis is dealing with chip variability. With the decreasing dimensions and lower voltages used by new generations of CMOS technology, global and local variations are becoming more and more important. Clock synthesis needs to take variation tolerance into account, to mitigate the impact of these variations.

Mainstream methods for clock tree synthesis typically work by building an initial clock structure in a bottom-up fashion, and then optimizing this structure to improve insertion delay, skew, power and area. These methods can consider variation tolerance by including techniques such as multi-corner optimization, common path sharing and balancing of cell and wire delays. For the global distribution of high-frequency clocks, a different approach is commonly used, which is based on using clock structures that are robust by construction, such as H-trees and clock meshes. Multisource clock synthesis combines these approaches, to support a range of clock structures that provide good trade-offs between robustness, power and area.

In this lecture we will first explain the basics of clock synthesis and timing closure for synchronous circuits, and then describe in more detail techniques and design styles for high-performance, variation tolerant clock structures.

Asynchronous timing (Alex Yakovlev, University of Newcastle)

Can we coordinate circuits in time without clock? The answer is yes, if we use asynchronous circuits. These circuits, also called self-timed circuits, do not rely on a global clock signal and operate using local synchronization mechanisms such as handshakes. This makes them very different from widely adopted synchronous circuits and promises many benefits, such as inherent resilience to process, voltage, temperature and aging variations, average rather than worst-case operation in time and power domains, and better modularity and compositionality. For example, asynchronous timing enables robust operation at near-threshold or sub-threshold voltages (NTV and STV), where the optimum point for energy-per-operation lies for many types of logic and memory. This capability allows asynchronous timing to fit ideally for systems powered by energy harvesting, e.g. Internet-of-Things nodes.

Despite these benefits, asynchronous timing is not yet widely adopted by industry, mainly because of the difficulties of integrating it into the standard EDA tool flows.

In this lecture we will provide a brief overview of the state-of-the-art of asynchronous timing. We will focus on its two main design paradigms, bundled data and delay-insensitive circuits, and compare their gains and penalties. We will demonstrate existing asynchronous tool support for both paradigms.

We will also highlight recent success stories, in particular, industrially adopted design flow for little digital hardware components asynchronous circuits providing flexible timing for analog/mixed-signal circuits such a power converters and AD converters.

Finally, we will discuss how these tools can be used for the design of elastic data-flow pipelines, as well as fully self-timed SRAMs, which allows creating systems where processors and memory can seamlessly operate at NVT/SVT. We will conclude by posing future research and development challenges that are currently on the agenda of the asynchronous community.

Metastability and Clock Domain Crossing (David M. Zar, Blendics)

Multiple-clock system-on-chip (SOC) designs require synchronization when transferring signals and data among clock domains and when receiving asynchronous inputs. Such synchronizations are often susceptible to metastability effects that may propagate into the receiving circuit and may cause malfunctions. To mitigate the nondeterministic effects associated with metastability, latches and flip-flops are often used to synchronize the data. Common structures for this purpose include pipelined flip-flops and FIFOs. There is, however, a probability that these circuits will not resolve from a metastable state within the allowed time. The probabilities are becoming a concern as technology nodes get smaller and as the intrinsic parameters of the devices become increasingly variable and problematic; scaling does not help us, anymore!

For multiple-clock SOC designs it is important to understand how synchronizer circuits may fail, and to be able to design reliability as measured by a particular allowable probability of failure, or level of failures in time (FIT). This lecture will present some common synchronization structures, where they may be used and how to evaluate their reliability for both an individual synchronizer and for a system with many synchronizers. Parameters that govern synchronizer reliability are contained in the process transistor-model and in the application of the synchronizer. These two different sources of parameters often involve two different designers and who may work in two different companies. Methods to unite these sources will be discussed.

Some of the latest research in this area will be presented along with models, examples of good synchronizer circuits and a discussion of why data flip-flops make terrible synchronizing devices.

Advanced Clocking (Jordi Cortadella, Universitat Politècnica de Catalunya)

Clock frequency is one of the most important parameters in system design and typically is a pre-defined target before synthesis. The time uncertainties in nanoelectronics circuits due to process, voltage, temperature and aging (PVTA) variations demand safe guardband margins that result in conservative clock frequencies. These margins imply a high cost in energy and performance.

In the last few years, several techniques for adaptive clocking have emerged with the aim of dynamically adapting the frequency of the clock to the dynamic variations (VTA) of the system. These techniques may contribute to reduce energy consumption up to 40%. Among all the sources of variations, the most challenging problem is the safe adaptation to voltage droops. This lecture will review some of the most recent advances in adaptive clocking.

Few companies have proposed different schemes based on anticipating voltage droops and quickly adapting the clock frequency to the delays of the system while the droop is active. Various approaches around this idea will be presented and discussed.

Techniques based on resilient circuits proposed by ARM and Intel will be also covered. These techniques are based on pushing clock frequency to the limits in a way that timing errors may be detected and corrected at runtime. These techniques require sophisticated mechanisms for error detection/correction not always available in conventional systems.

Finally, a technique based on substituting the PLL by a ring oscillator will be presented. The design and power/performance benefits of this technique will be analyzed.

3 Biographies

Jordi Cortadella is Professor and Head of the Computer Science Department at the Universitat Politcnica de Catalunya. He is a Fellow of the IEEE and member of the Academia Europaea. He holds a M.S. and a Ph.D. degree in Computer Science (Universitat Politcnica de Catalunya, 1985 and 1987). In 1988, he was a Visiting Scholar at the University of California, Berkeley. His research interests include formal methods and computer-aided design of VLSI systems with special emphasis on asynchronous circuits, concurrent systems and logic synthesis. He has co-authored numerous research papers and has been invited to present tutorials at various conferences.

Prof. Cortadella has served on the technical committees of several international conferences in the field of Design Automation and Concurrent Systems and is associate editor of the IEEE Transactions on CAD of Integrated Circuits and Systems. He received best paper awards at DAC 2004, ASYNC 2004 and ACSD 2009. In 2003, he was the recipient of a Distinction for the Promotion of the University Research by the Generalitat de Catalunya.

Luciano Lavagno received his Ph.D. in electrical engineering and computer science from the University of California at Berkeley, CA, USA, in 1992. He has been the architect of the POLIS project, developing a complete hardware/software co-design environment for control-dominated embedded systems, and an architect of the CtoSilicon high-level synthesis system from Cadence Design Systems. He is a co-author of two books on asynchronous circuit design, of a book on hardware/software co-design of embedded systems, and has published over 200 journal and conference papers.

He is currently a full Professor with the Department of Electronics and Telecommunication Engineering of Politecnico di Torino, Italy. He has been an Associate Editor of IEEE TCAS and ACM TECS. His research interests include the synthesis of asynchronous and low-power circuits, the concurrent design of mixed hardware and software embedded systems, and the high-level synthesis of hardware modules from algorithmic specifications.

Koen van Eijk is a Scientist in the Design Group at Synopsys, Inc, where he works on clock tree synthesis. Prior to joining Synopsys in 2012, he was a Chief Technologist at Magma Design Automation, where he worked on placement, physical optimization, floorplanning, and hierarchical design implementation. Dr. van Eijk studied at the Electrical Engineering Department of the Eindhoven University of Technology, from which he graduated with honors in 1992 and obtained a Ph.D. degree in 1997, on the topic of equivalence checking for digital circuits.

Alex Yakovlev was a Dream Fellow of Engineering and Physical Sciences Research Council (EPSRC), United Kingdom, to investigate different aspects of energy-modulated computing during 2012-2013.

He received D.Sc. from Newcastle University in 2006, and M.Sc. and Ph.D. from St. Petersburg Electrical Engineering Institute in 1979 and 1982 respectively, where he worked in the area of asynchronous and concurrent systems since 1980, and in the period between 1982 and 1990 held positions of assistant and associate professor at the Computing Science department. Since 1991 he has been at the Newcastle University, where he is a professor and head of the MicroSystems research group at the School of Electrical and Electronic Engineering. His current interests and publications are in the field of modelling and design of asynchronous, concurrent, real-time and dependable systems on a chip. He has published 8 edited and co-authored monographs and more than 300 papers in academic journals and conferences, and has managed over 30 research contracts. He is a Senior Member of the IEEE and Fellow of IET.

He has chaired program committees of several international conferences, including the IEEE Int. Symposium on Asynchronous Circuits and Systems (ASYNC), Petri nets (ICATPN), Applications of Concurrency to Systems Design (ACSD), and he has been Chairman of the Steering committee of the Conference on Application of Concurrency to System Design since 2001. In April 2008 he was General Chair of the 14th ASYNC Symposium and 2nd Int. Symposium on Networks on Chip, 22nd PATMOS, and Tutorial Chair at Design Automation and Test in Europe (DATE) in 2009. He has served as a Steering committee member for ASYNC, NOCS, ICATPN, PATMOS. He was recently an invited speaker at DDECS 2010, KTN event on Power Management in 2010, DATE 2011, DCIS 2014, where he spoke on Asynchronous Systems and Energy-Modulated Computing. He was a tutorial organiser and speaker at DATE 2013, gave lectures and courses on asynchronous design at ARM, IMEC and Dialog Seminconductor in 2015.

David M. Zar is a Co-Founder of Blendics where he is also a Senior Engineer working on metastability-related tools and analysis. Along with several of the other Co-Founders of Blendics, Mr. Zar came from Washington University in St. Louis where he was involved in asynchronous circuit design as well as metastability work on and off for over 19 years. He is the principal developer of MetaACE, the first commercially available tool for the analysis of metastability in synchronizers. In addition, Mr. Zar has worked in the areas of medical devices, networking and cybersecurity for the past 25 years.

DATE 16

DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

Clock Synthesis and Chip Variability

Koen van Eijk

Synopsys

DATE 16

DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

Modern Clocking Strategies: Clock Synthesis and Chip Variability

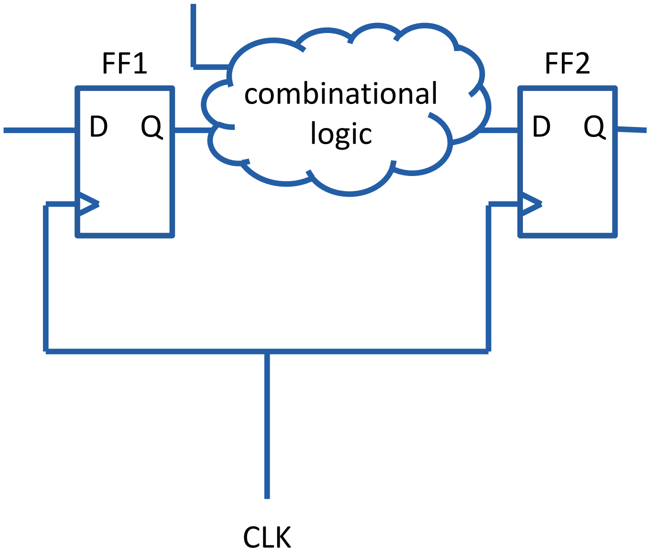
Koen van Eijk
Synopsys

SYNOPSYS®
Silicon to Software™

Overview

1. Clock Synthesis Fundamentals
2. Chip Variability and Timing Sign-Off
3. Variation Tolerant Clock Structures
4. Design Flow Aspects and Automation

Synchronous Circuits



Setup timing check

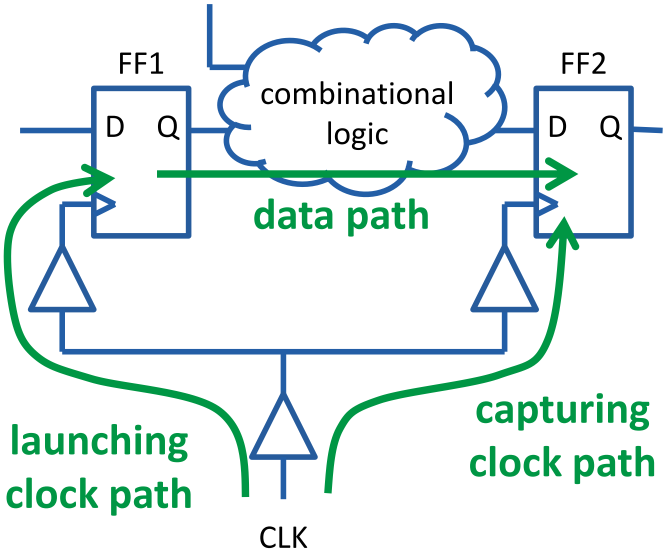
- The data must arrive at the D input before the arrival of the next clock edge

Hold timing check

- The data must remain valid long enough after the arrival of the clock edge

Multiple clock domains, clock gating, latches, path exceptions, …

Synchronous Circuits



Setup timing check

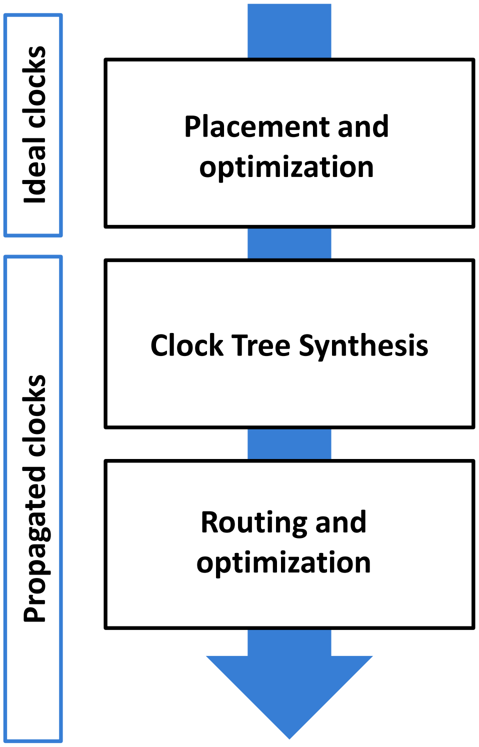
- The data must arrive at the D input before the arrival of the next clock edge

Hold timing check

- The data must remain valid long enough after the arrival of the clock edge

Multiple clock domains, clock gating, latches, path exceptions, …

Clock Tree Synthesis


Ideal clocks

Propagated clocks

Placement and optimization

Clock Tree Synthesis

Routing and optimization

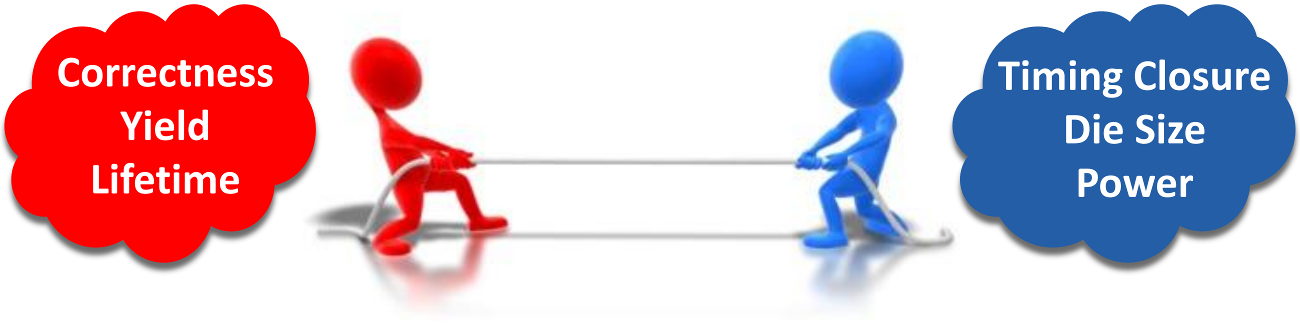
- Clock tree synthesis creates the network that connects the system clock to the sequential elements of the chip
- Main goals
  - Balance insertion delays
  - Minimize clock area and power
- The later flow stages include clock optimization

Clock Balancing

- Skew: Difference in arrival times at clock inputs
  - Global: Between the clock inputs of any two sequential elements in the same clock domain
  - Local: Between two clock inputs that launch and capture a timing path
    - Positive skew: Capture comes later than launch
    - Negative skew: Capture comes earlier than launch
- Useful skew: Use skew to meet signal timing
  - Create positive skew on timing-critical paths
  - Accept negative skew on non-timing-critical paths

Timing Sign-Off

- Use static timing analysis to check that all timing constraints are satisfied
- Determine the worst possible conditions for each type of timing check
- How to account for chip variability?



Correctness
Yield
Lifetime

Timing Closure
Die Size
Power

Sources of Variation

Process

- Variation in critical dimensions
- Random dopant fluctuation
- Variation of the gate oxide thickness

Voltage

- Offset in the voltage regulator
- Power noise (IR drop, di/dt noise)

Temperature

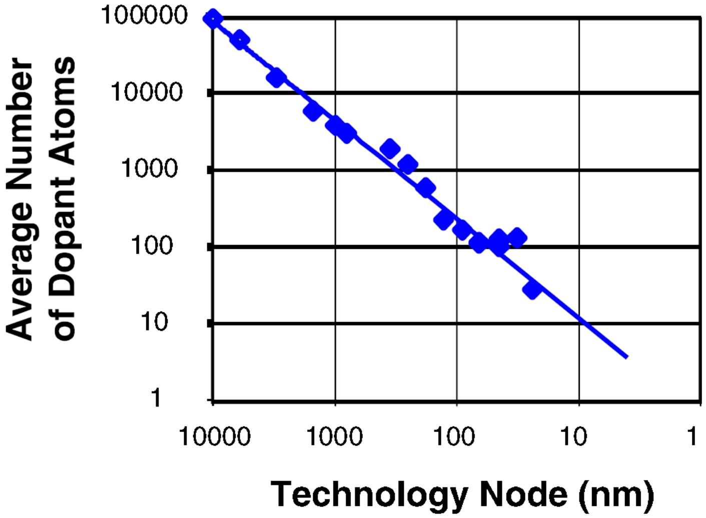
- Ambient temperature changes
- Global variations and local fluctuations due to power dissipation

Aging

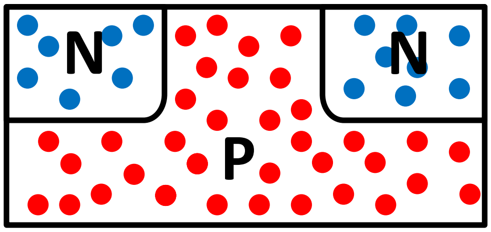
- Hot carrier injection
- Bias temperature instability

Reference: M. Wirnshofer, *Variation-Aware Voltage Scaling for Digital CMOS Circuits*, Springer Series in Advanced Microelectronics 41

Example: Random Dopant Fluctuation



Source: K. Kuhn et al., Managing Process Variation in Intel's 45nm CMOS Technology, Intel Technology Journal, Volume 12, Issue 2, 2008



- Main origin of random variability in conventional bulk transistors
  - Responsible for ~60% of device-to-device V_T variation at 45nm
- For SOI and 3D multi-gate transistors, the channel/body doping can be eliminated to mitigate RDF effects
  - Still other sources of random variability

Classifying Variation

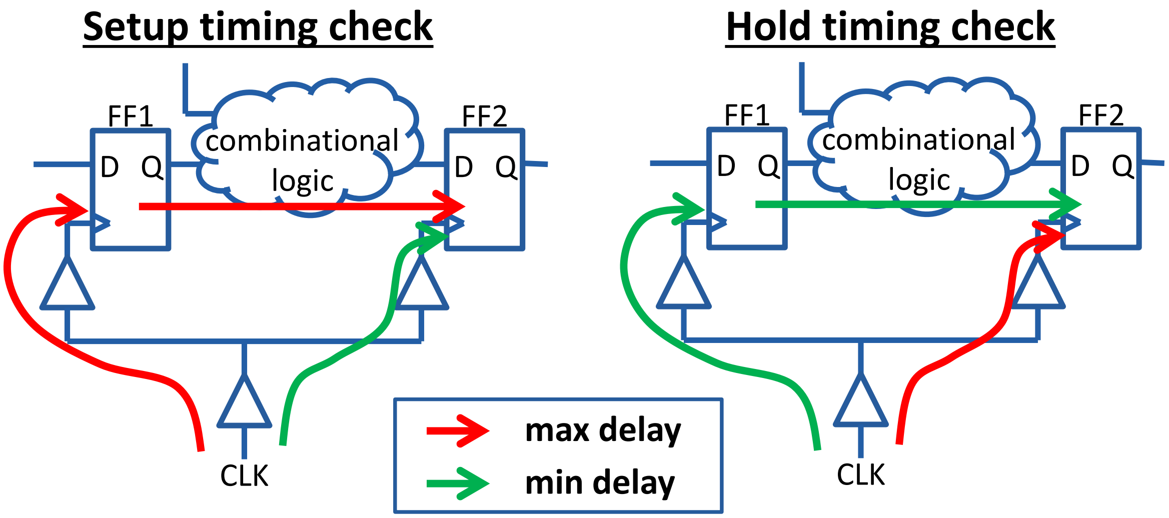
- Systematic versus random
  - Systematic variations are deterministic in nature, can be attributed to layout or manufacturing equipment related effects, and generally show spatial correlation behavior
- Global versus local
  - Global variations affect similar components on the same die in the same way
- Static versus dynamic
  - Dynamic variation is time-dependent, with time constants ranging from small (power noise) to large (aging)
  - Process variation is static

Modeling variation

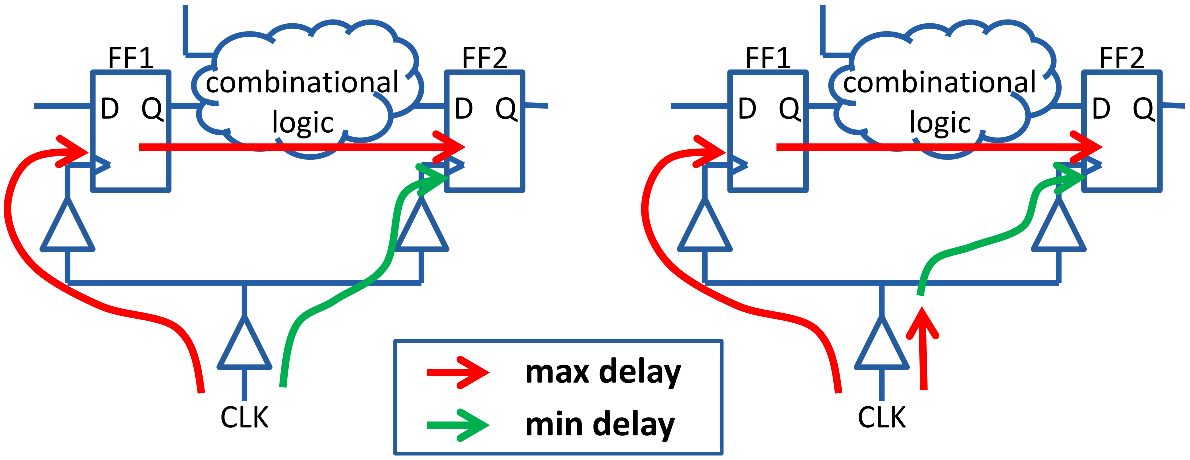
- Multicorner analysis
  - Model chip-to-chip variations
  - Examples: Ambient temperature, metal dimensions
- Basic OCV modeling
  - Model variations with min/max delays
  - Examples: Power noise, temperature fluctuations
- Advanced and parametric OCV
  - Model correlations to reduce pessimism
  - Examples: Random dopant fluctuation, effective gate lengths

Basic OCV Modeling

- Perform conservative checks by applying maximum and minimum delays
  - Example: Define min/max by -10%/+10% derating



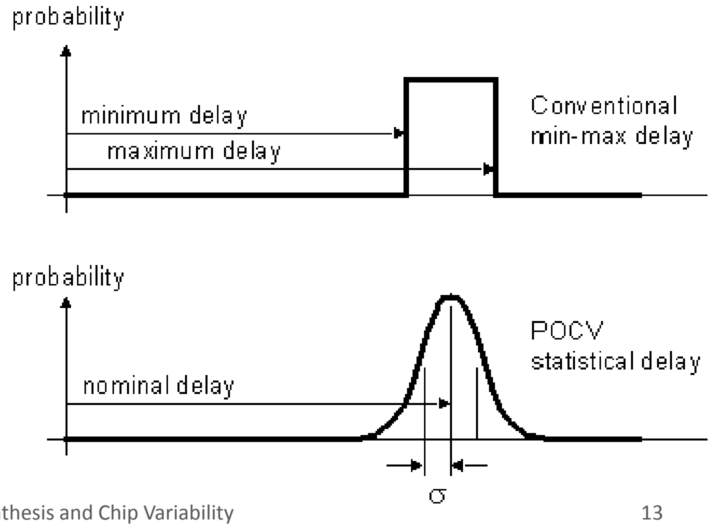
Clock Reconvergence Pessimism Removal



- It is pessimistic to assume both minimum and maximum delays for the common part
- CRPR removes this pessimism by only assuming maximum delays for the common part

Advanced and Parametric OCV

- Advanced OCV
  - Derate values as a function of logic depth (for random variations) and location (for systematic variations)
- Parametric OCV
  - Computes arrival times, required times and slacks as statistical distributions



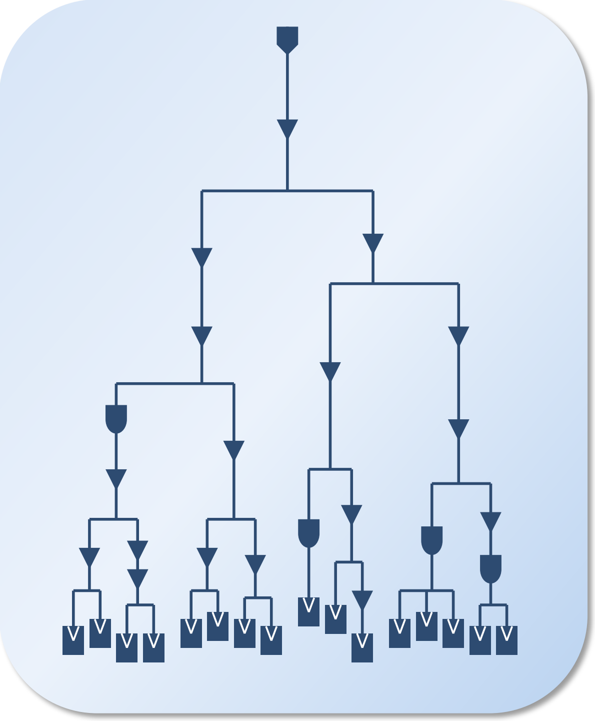
Creating Robust Clock Structures

Synthesis algorithm

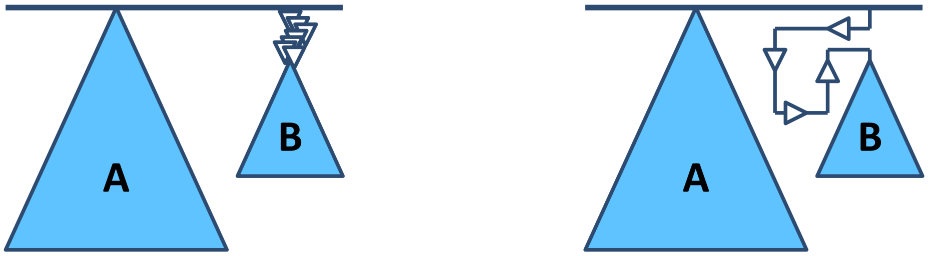
- Bottom-up tree construction using clustering, buffering, clock gate merging, splitting, sizing and relocation

Techniques for improving variation tolerance

- Multi-corner optimization
- Gate/wire delay balancing
- Common path sharing

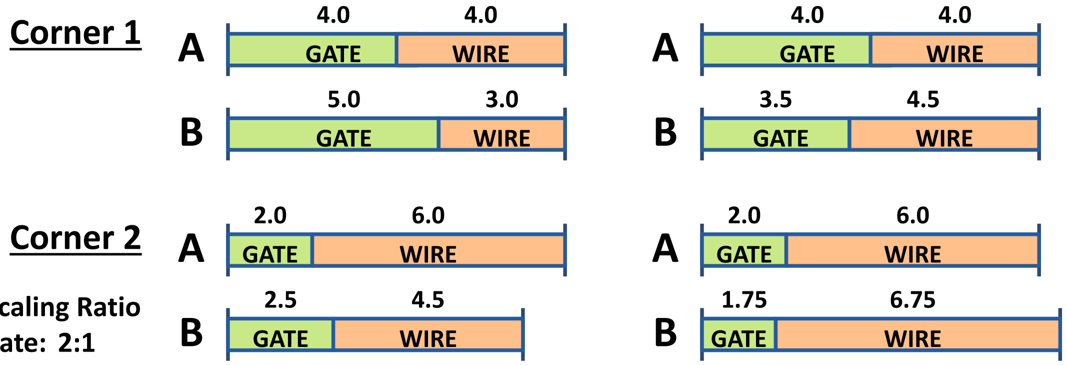


Gate/Wire Delay Balancing



Balancing with Cell Delay Only

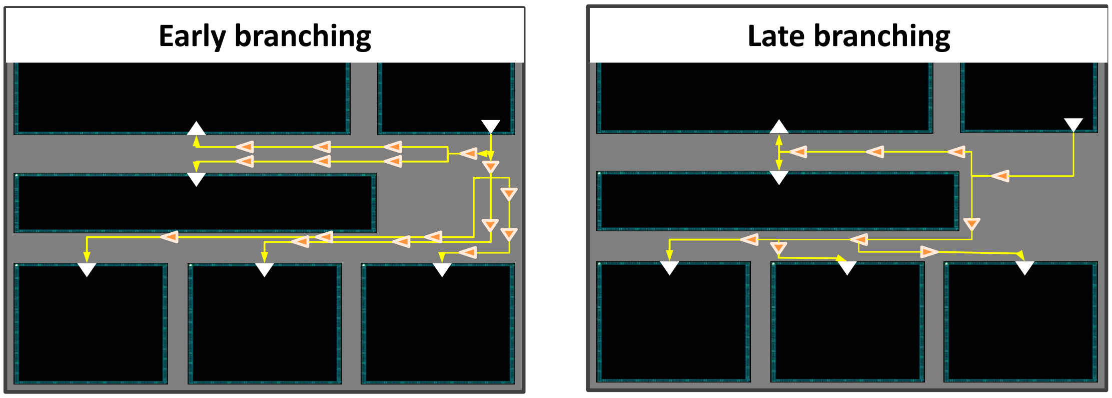
Balancing with Cell & Wire Delay



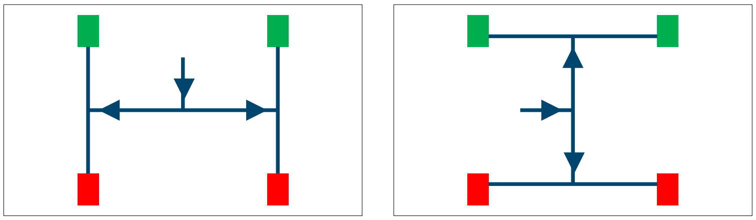
Corner Scaling Ratio
For gate: 2:1
For Wire: 1:1.5

Common Path Sharing

- **Prefer late branching**



- **Timing-aware clustering**



Robust by Construction

Global clock tree (H-tree)

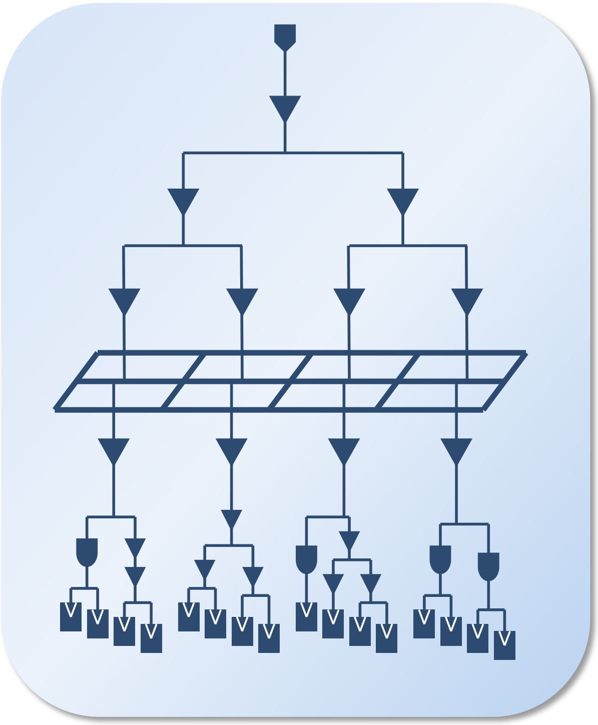
- Structural balancing
- Strong repeaters and high layers to achieve low latency
  - Less impact from OCV derating
  - Less sensitive to variations

Clock mesh

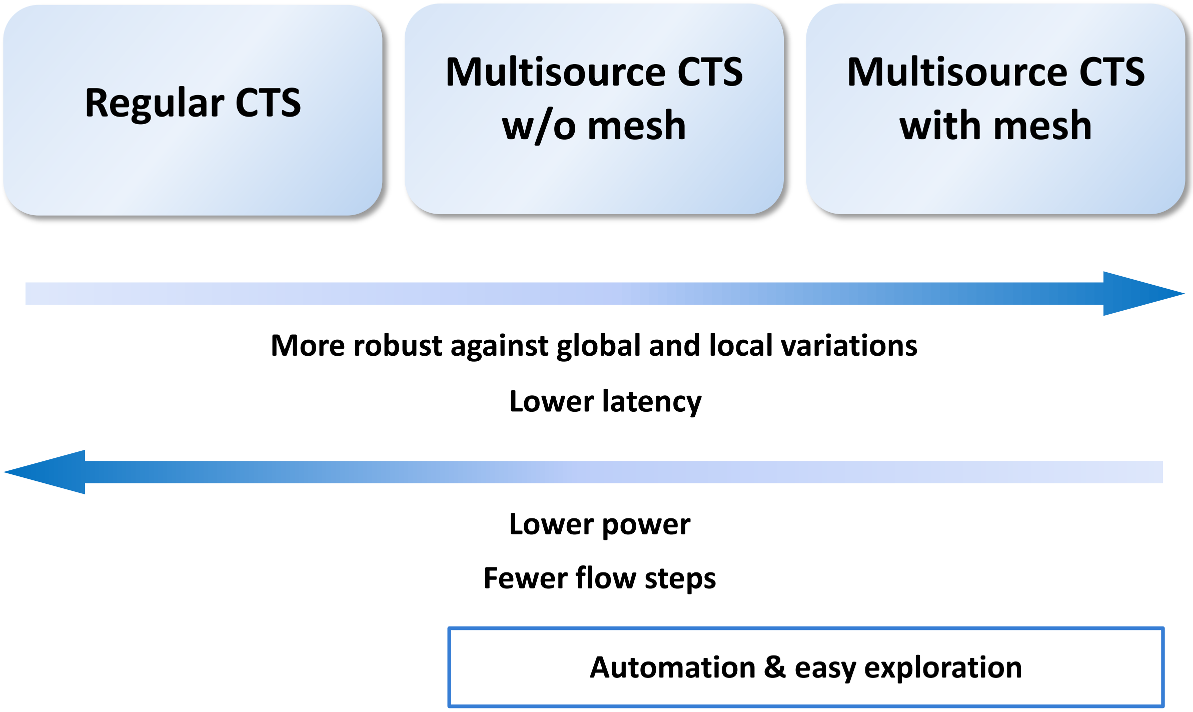
- OCV tolerance

Clock subtrees

- Flexibility for fine-grain clock gating and useful skew



Trade-offs for Multisource CTS



Regular CTS

Multisource CTS w/o mesh

Multisource CTS with mesh

More robust against global and local variations

Lower latency

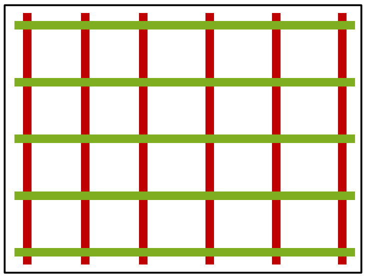
Lower power

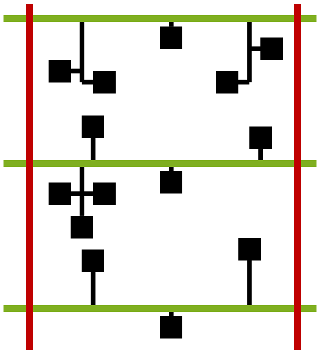
Fewer flow steps

Automation & easy exploration

Mesh Creation

1. Create mesh routes
   - Non-default rule, shielding
2. Insert mesh drivers
   - Placed in regular pattern
3. Hook up pins to mesh
   - Restrict topology to avoid daisy chaining
4. Analyze and annotate delays
   - Integrated circuit simulation

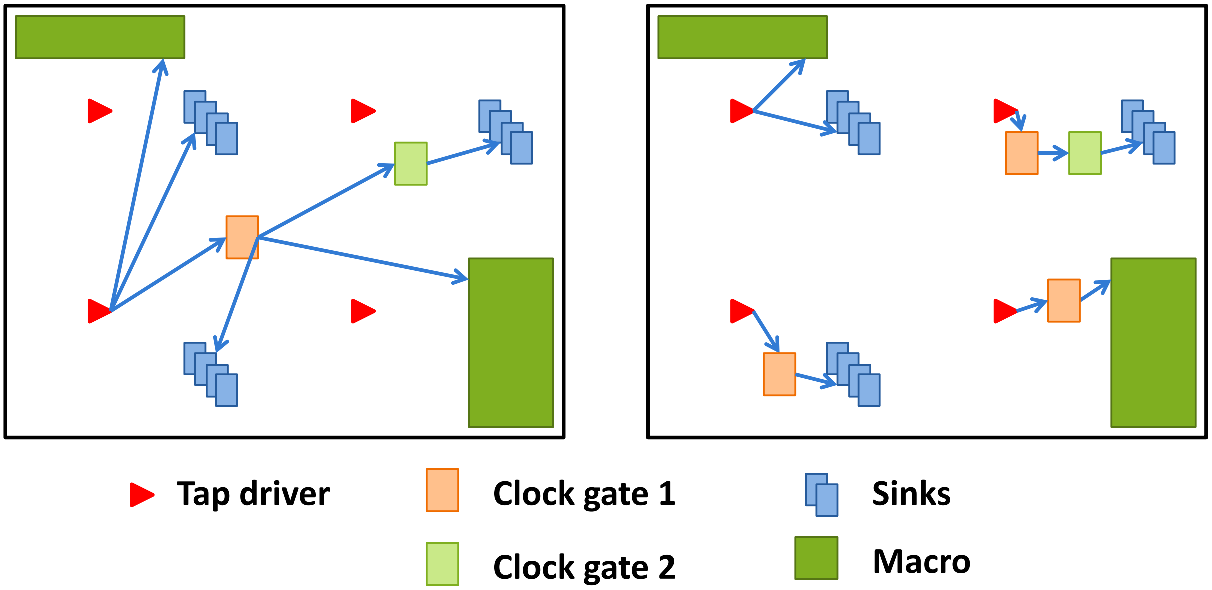




Tap Assignment

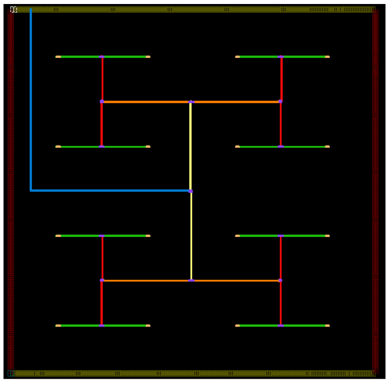
Assigning clock sinks to tap drivers

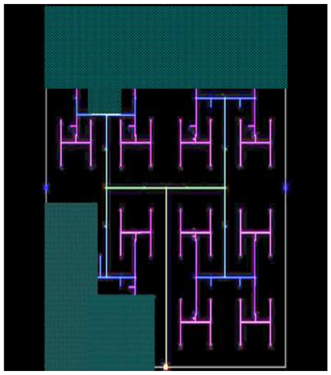
- Involves clock gate splitting



Global Distribution with H-tree

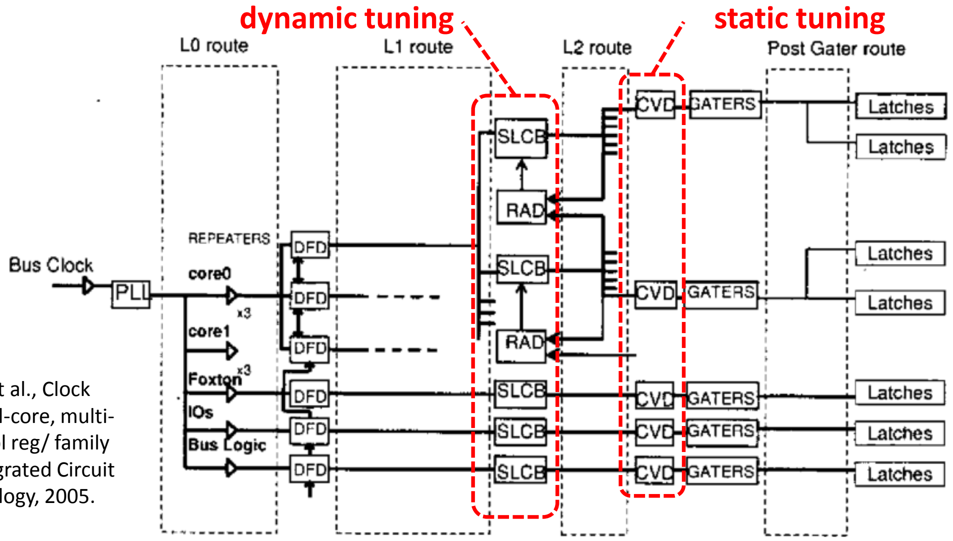
- **H-tree synthesis**
  - **Simultaneous buffering and routing**
  - **Balanced routing per level**
  - **Layers that are primarily used for power distribution, with few tracks available**
- **Handle fragmented floorplans**
  - **Obstructed areas**
  - **Areas with less regular placement**
  - **Dummy branches for balancing**





Post-Silicon Clock Tuning

- Insert cells with tunable delays and tune the delay to minimize skew
  - Typically part of the global clock distribution

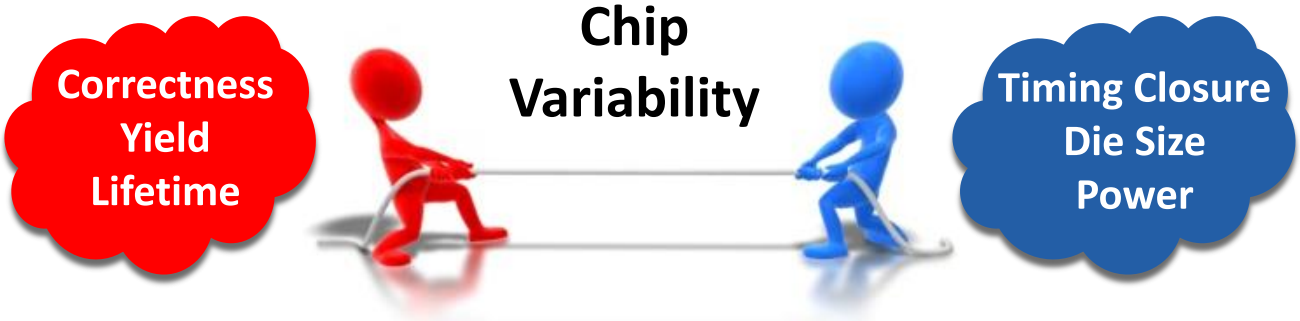


dynamic tuning

static tuning

Source: B. Doyle et al., Clock distribution on a dual-core, multi-threaded Itanium/spl reg/ family microprocessor. Integrated Circuit Design and Technology, 2005.

Summary



Correctness
Yield
Lifetime

Chip
Variability

Timing Closure
Die Size
Power

- Variation tolerant clock structures
  - Improving balancing and path sharing in bottom-up tree construction
  - Global distribution with H-tree
  - Clock mesh for very high performance designs
- Complete flow support for multisource CTS
  - Tap synthesis, mesh creation and analysis, global tree synthesis

DATE 16

DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

Asynchronous Timing

Alex Yakovlev

University of Newcastle

DATE 16

DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

Modern Clocking Strategies: Asynchronous Timing

Jordi Cortadella
UPC, Barcelona

Alex Yakovlev

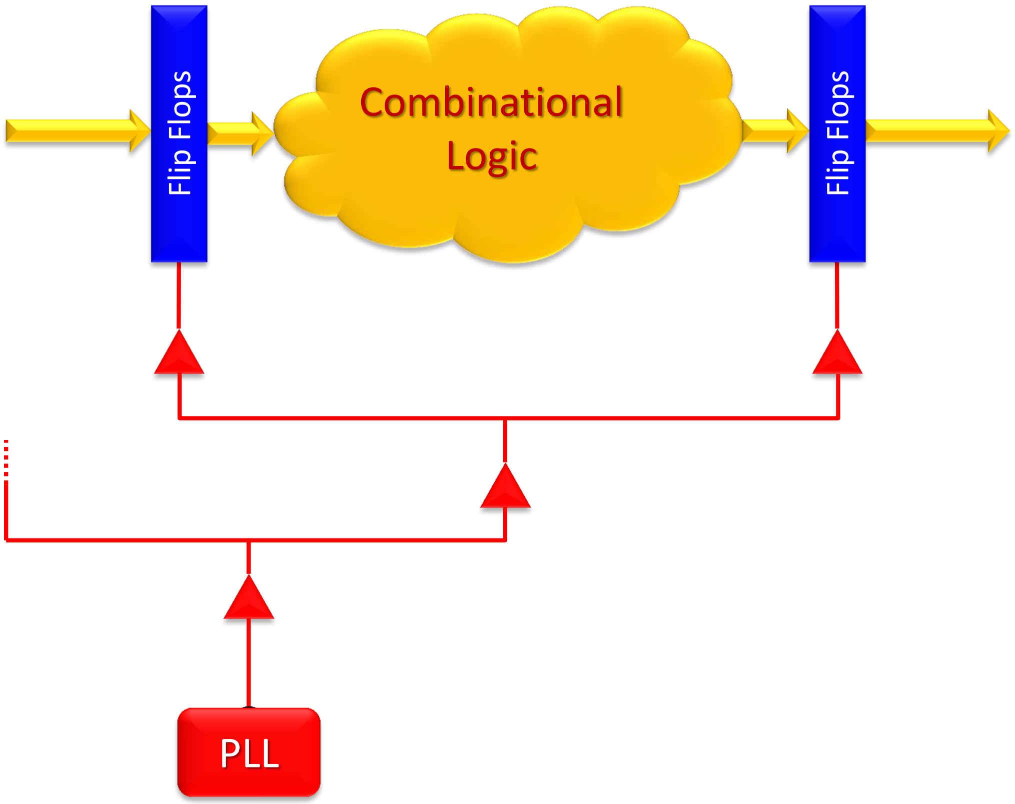
Newcastle University, UK

Overview

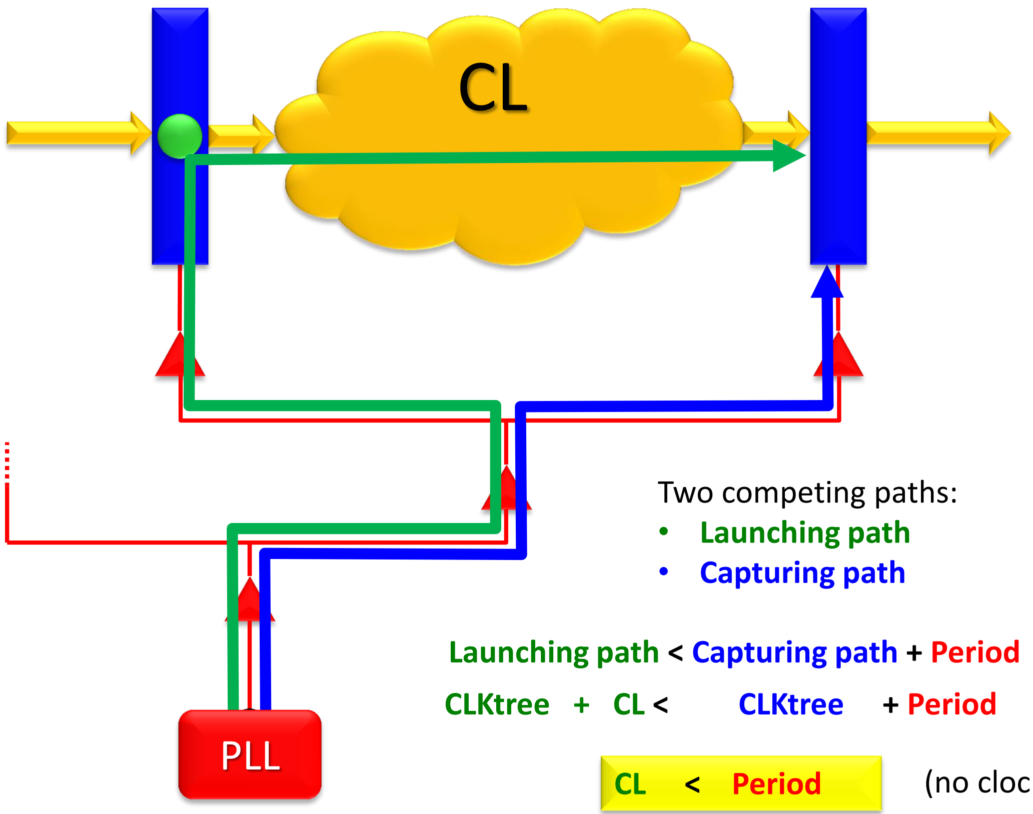
- Starting from Synchronous
- Completion Detection
- Handshaking
- Why going asynchronous?
- Towards near and sub-threshold
- GALS – Globally Asynchronous, Locally Synchronous
- Design Automation

Synchronous circuits

Synchronous circuit



Synchronous circuit



Two competing paths:

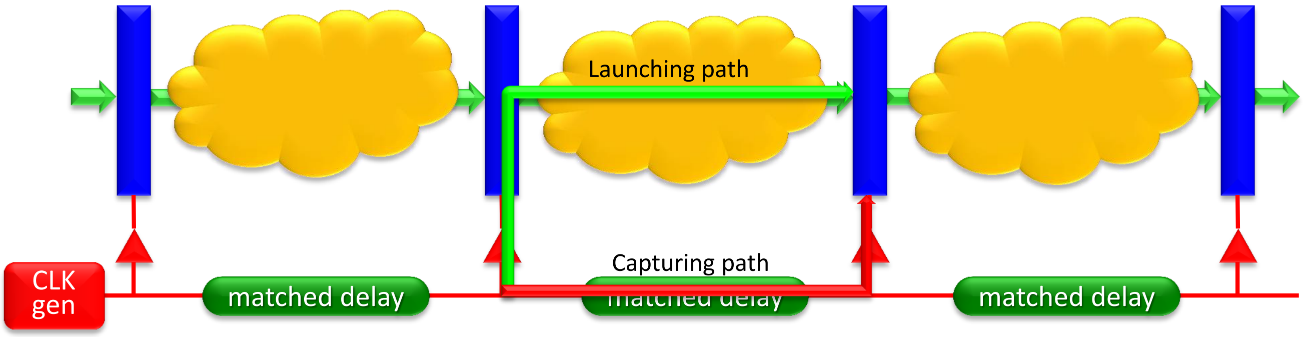
- Launching path
- Capturing path

Launching path < Capturing path + Period

CLKtree + CL < CLKtree + Period

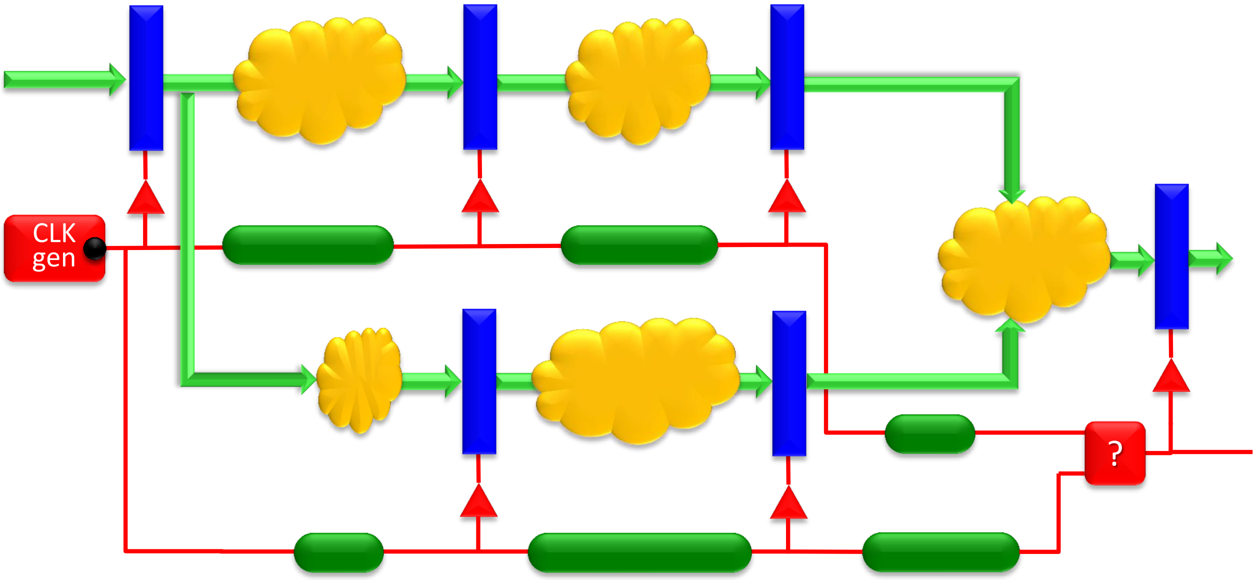
CL < Period (no clock skew)

Source-synchronous



- No global clock required
- More tolerance to PVT variations
- Period > longest combinational path
- Good for acyclic pipelines

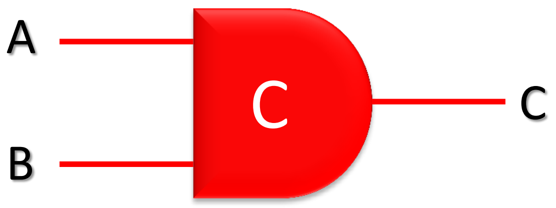
Source-synchronous with forks and joins



How to synchronize incoming events?

Completion detection

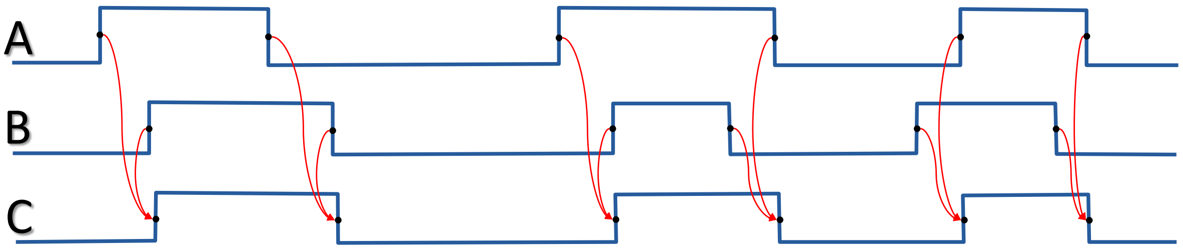
C element (Muller 1959)



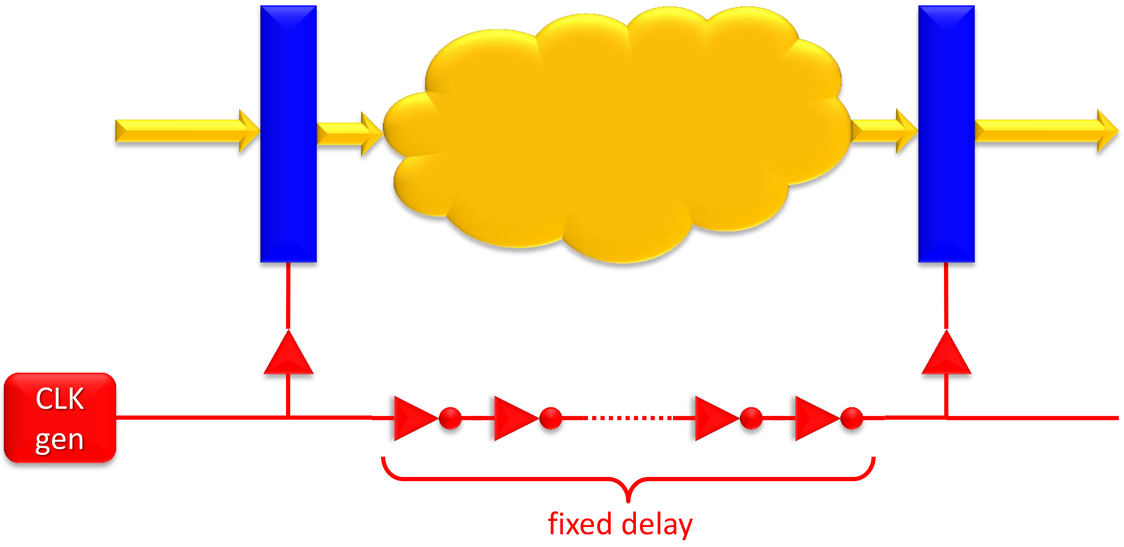
A	B	C
0	0	0
0	1	C
1	0	C
1	1	1

C= AB + C(A+B)

(many implementations exist)



Completion detection



The fixed delay must be longer than the worst-case logic delay (plus variability)

Q: could we detect when a computation has completed ASAP ?

Delay-insensitive codes: Dual Rail

- **Dual rail: every bit encoded with two signals**

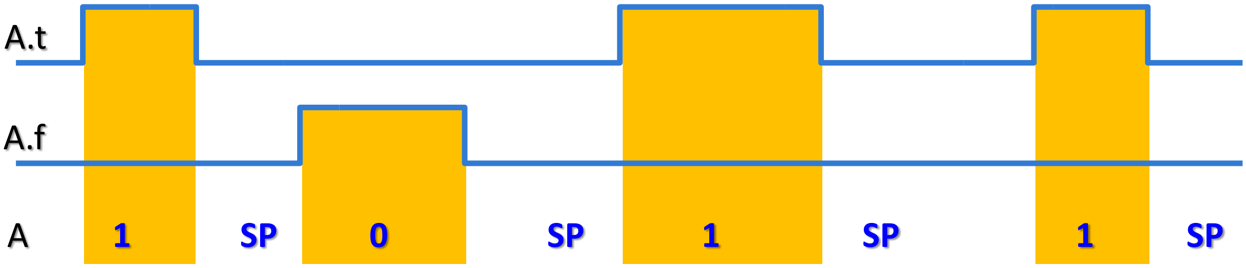
A.t	A.f	A
0	0	Spacer
0	1	0
1	0	1
1	1	Not used

Plus:

- Systematic code
- Easy to extract value
- Easy to detect completion

Minus:

- More area
- More power

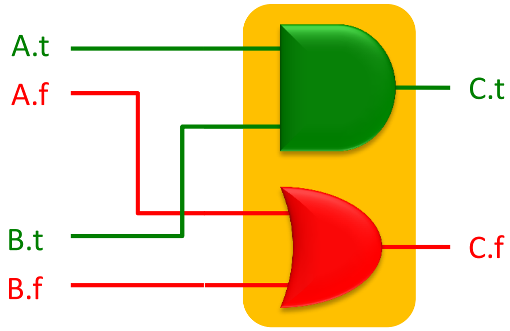


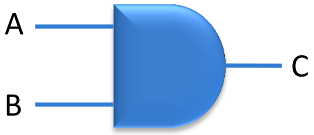
DI codes (1-of-n and m-of-n)

- **1-of-4:**
  - **0001=> 00, 0010=>01, 0100=>10, 1000=>11**
- **2-of-4:**
  - **1100, 1010, 1001, 0110, 0101, 0011 – total 6 combinations (cf. 2-bit dual-rail – 4 comb.)**
- **3-of-6:**
  - **111000, 110100, …, 000111 – total 20 combinations (can encode 4 bits + 4 control tokens)**
- **2-of-7:**
  - **1100000, 1010000, …, 0000011 – total 21 combinations (4 bits + 5 control tokens)**

Dual-Rail AND gate

A	B	C
SP	SP	SP
0	-	0
-	0	0
SP	1	SP
1	SP	SP
1	1	1





Single rail data vs. dual rail

Some back-of-the-envelope estimations:

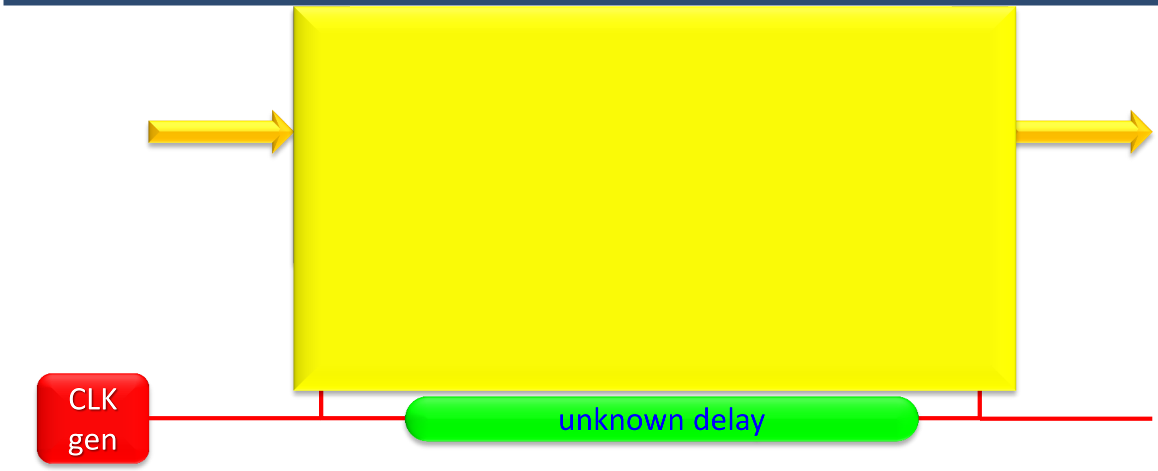
	Single rail	Dual Rail
Area	1	2
Delay	1	<< 1
Static power	1	2
Dynamic power	< 0.2	2

Dual rail:

- Good for speed
- Large area
- High power consumption

Handshaking

Handshaking

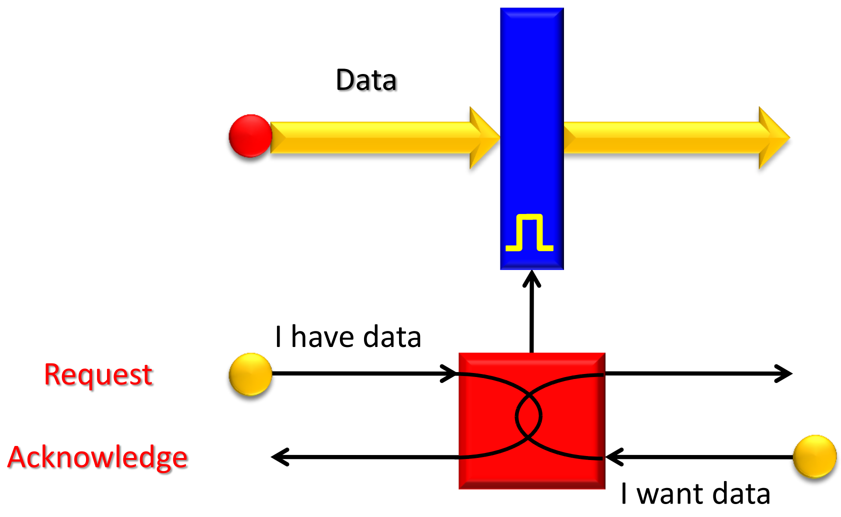


Assume that the source module can provide data at any rate:

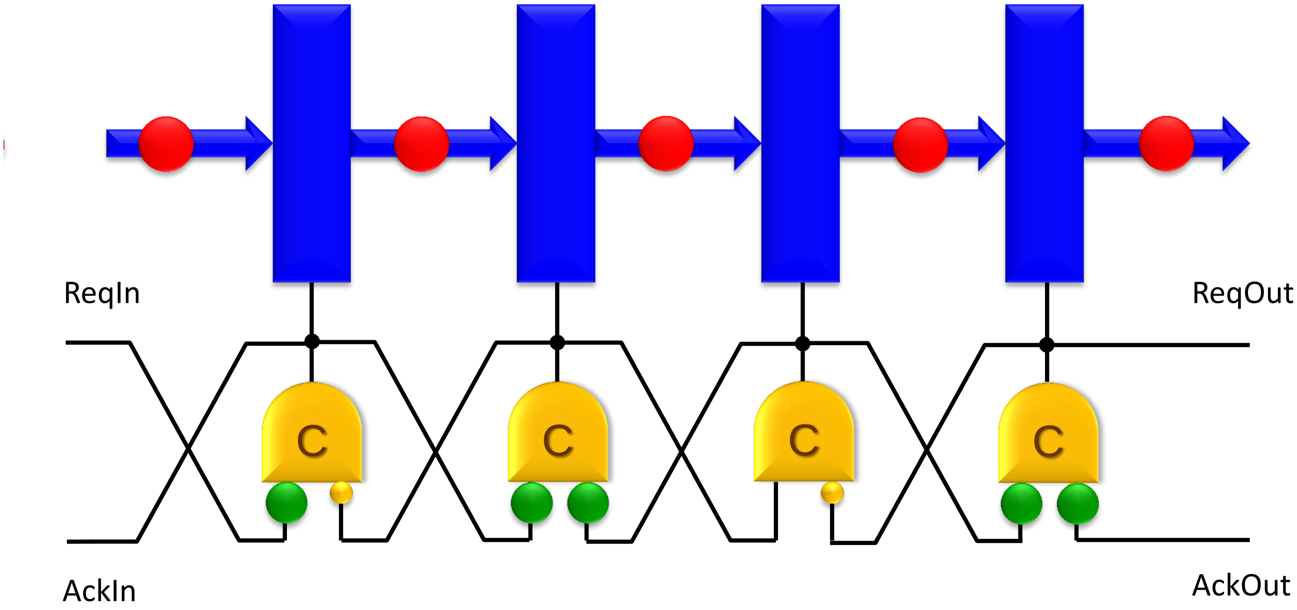
- When should the CLK generator send an event if the internal delays of the circuit are unknown?

Solution: handshaking

Handshaking

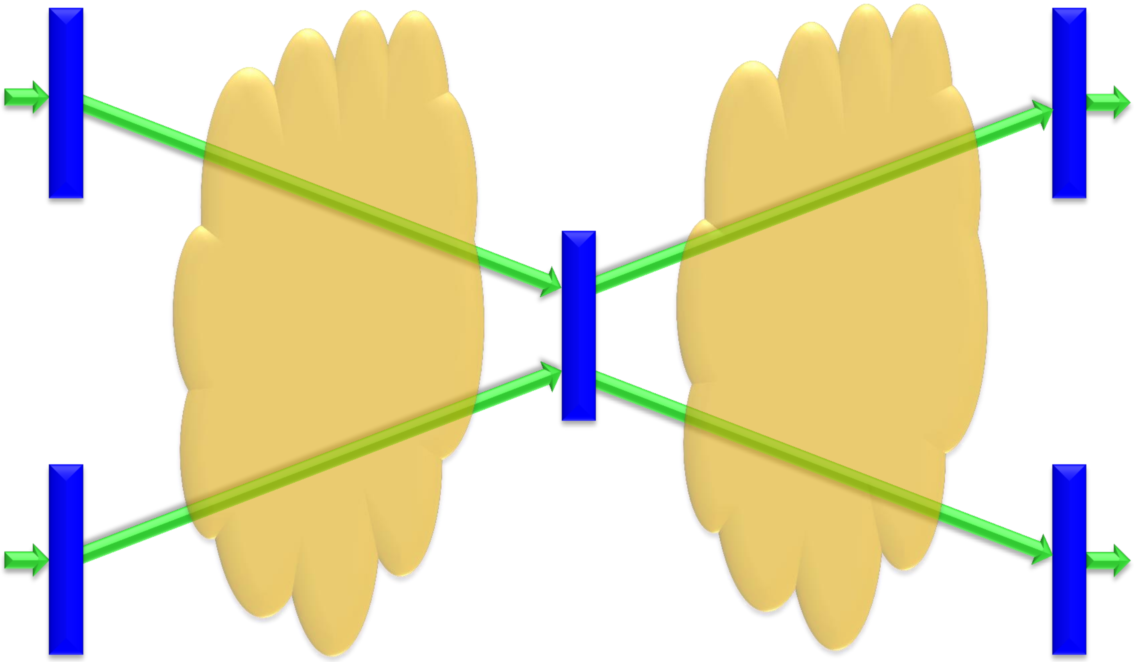


Asynchronous elastic pipeline

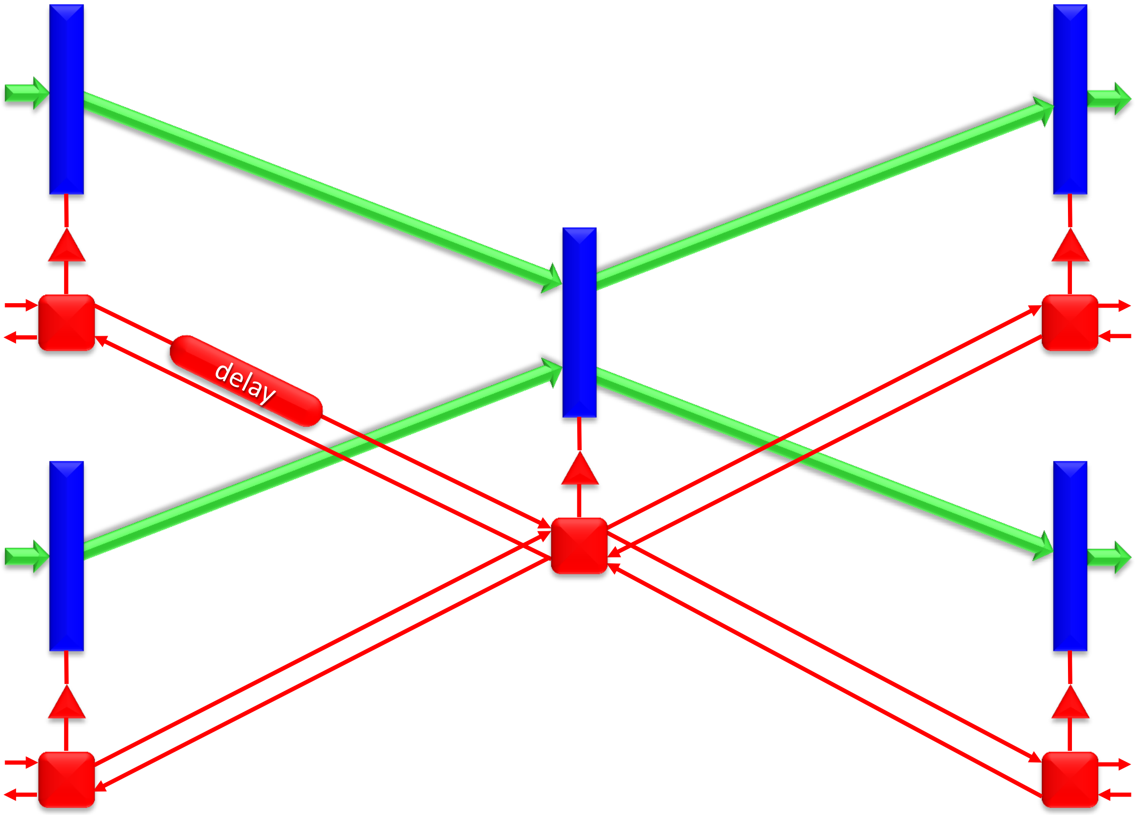


- David Muller's pipeline (late 50's)
- Sutherland's Micropipelines (Turing award, 1989)

Multiple inputs and outputs

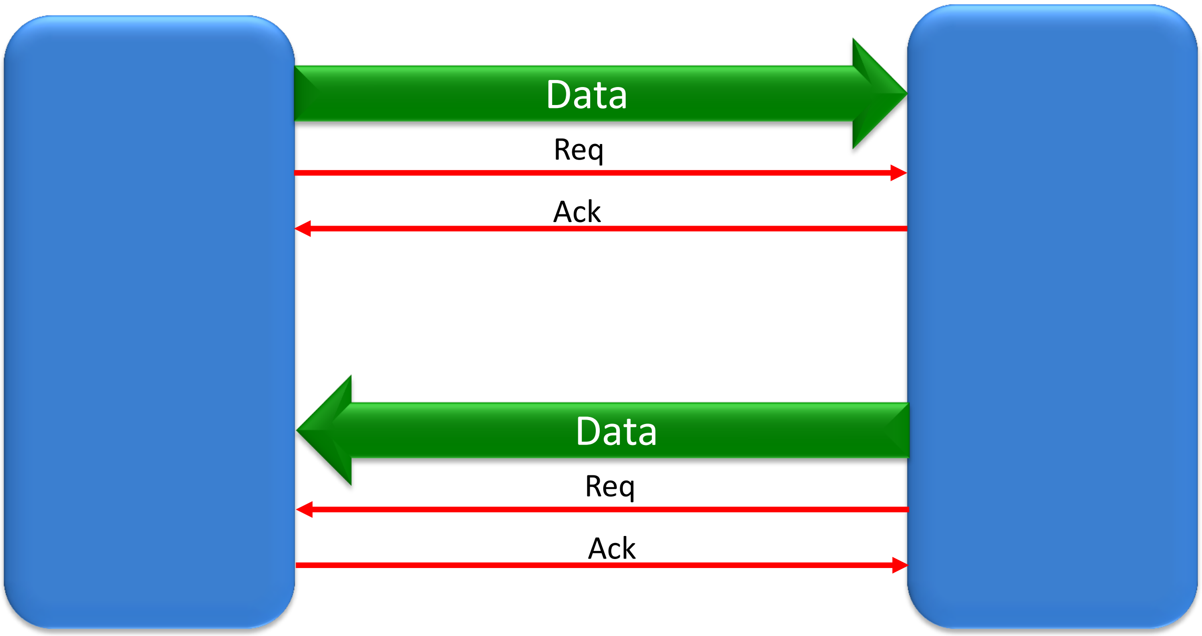


Multiple inputs and outputs

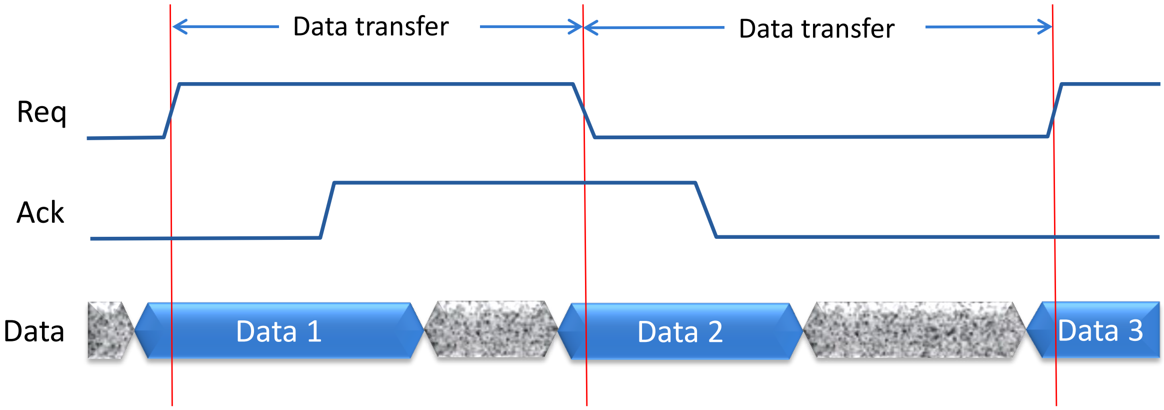


Channel-based communication

- A channel contains data and handshake wires

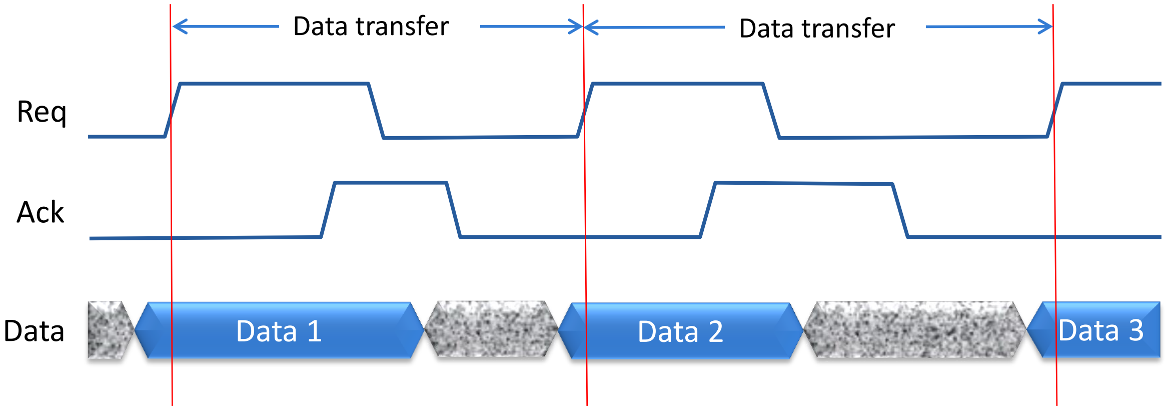


Two-phase protocol



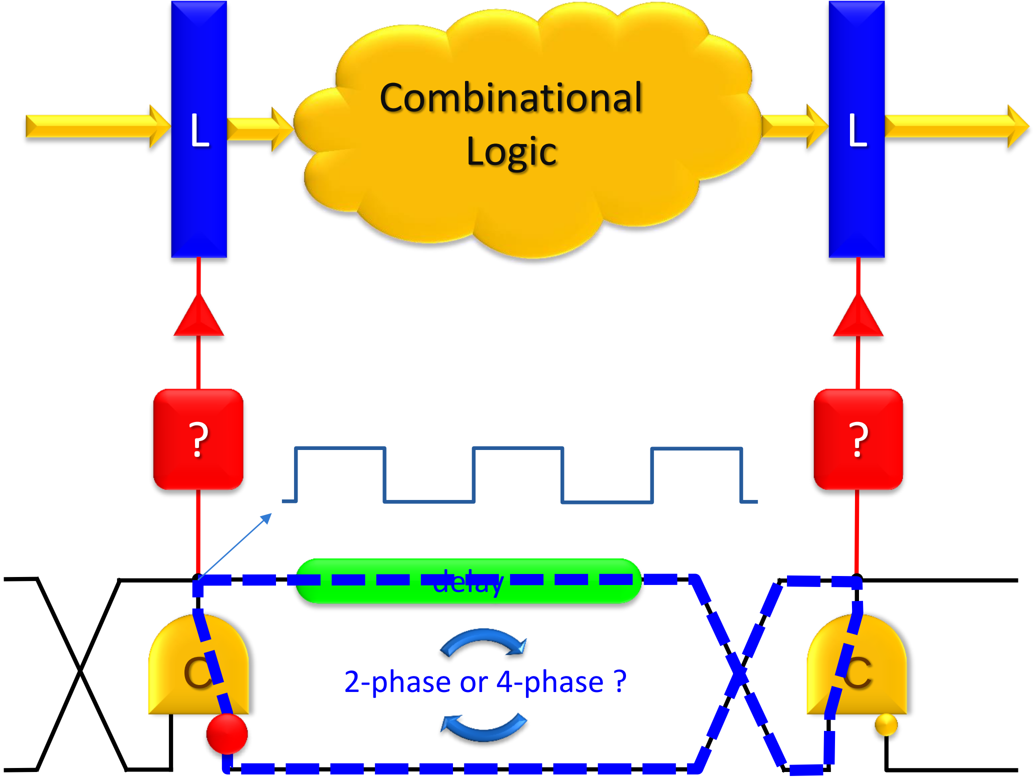
- Every edge is *active*
- It may require double-edge triggered flip-flops or pulse generators

Four-phase protocol

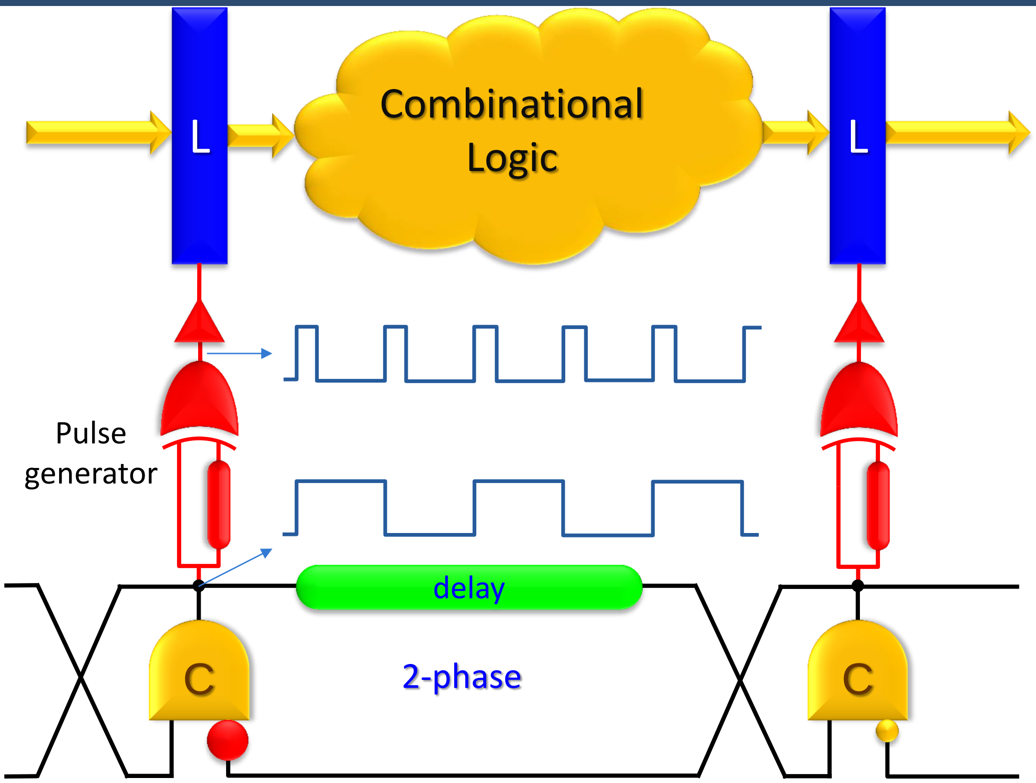


- Valid data on the *active* edge of Req
- Req/Ack must ***return to zero*** before the next transfer
- Different variations of the 4-phase protocol exist

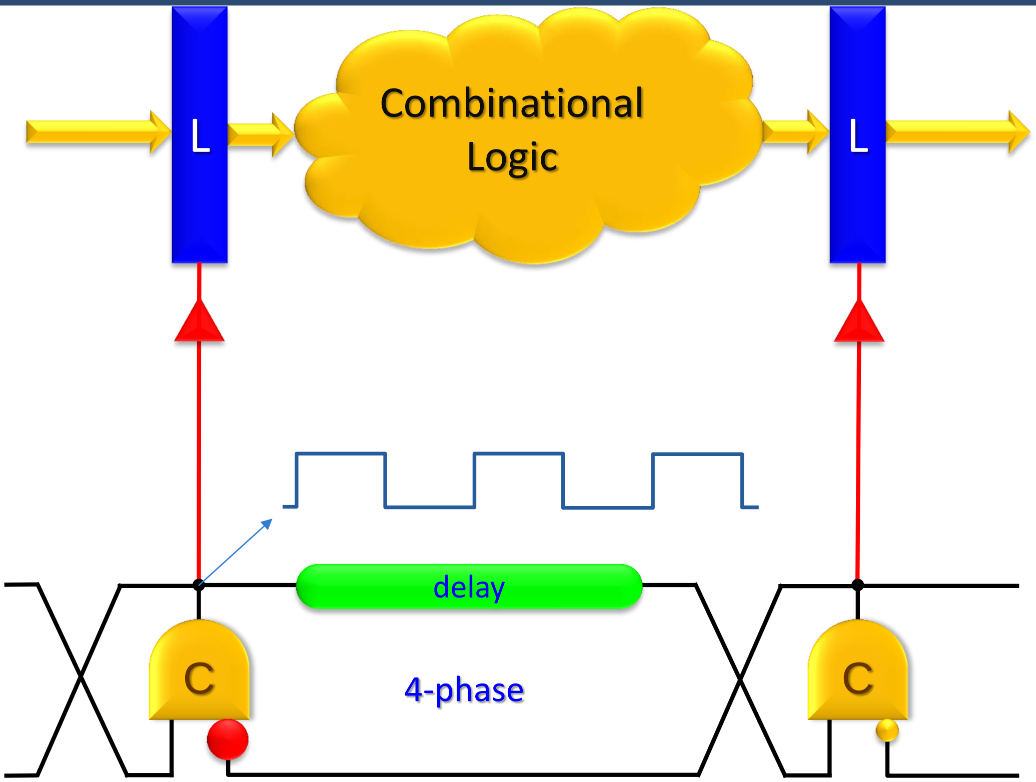
How to memorize?



How to memorize?



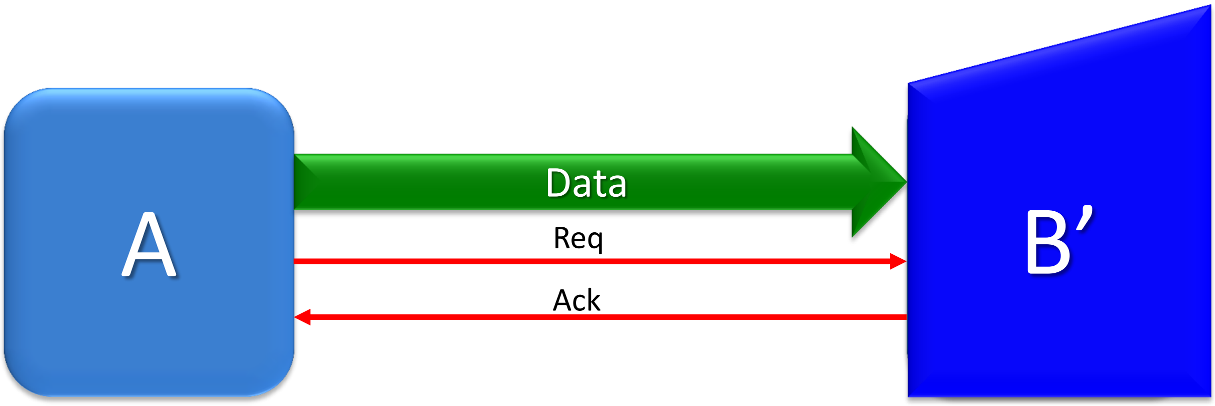
How to memorize?



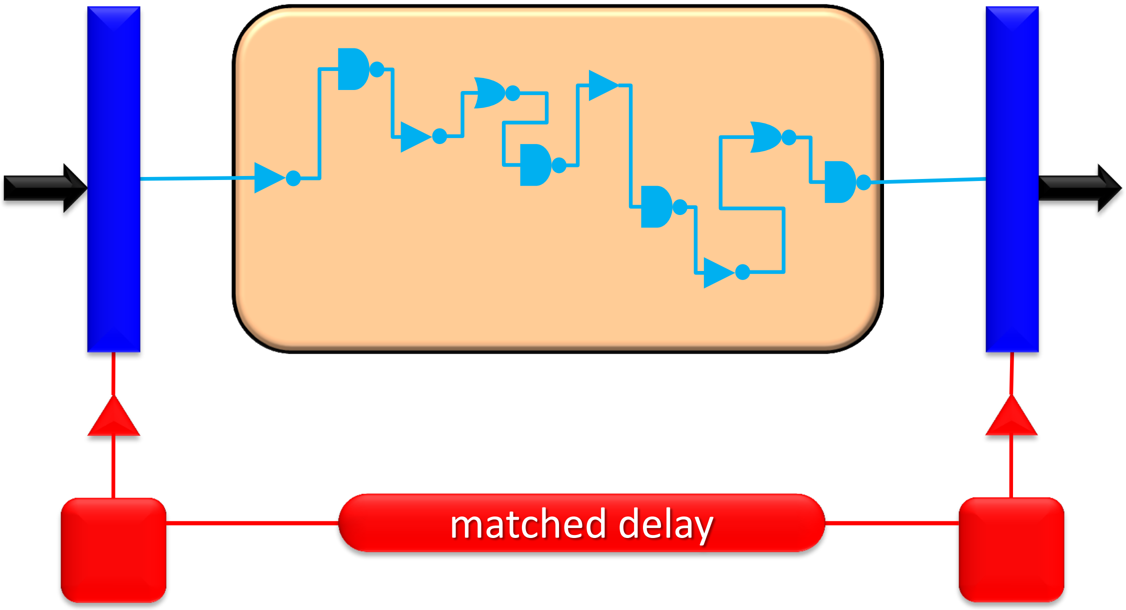
Why going asynchronous?

Modularity

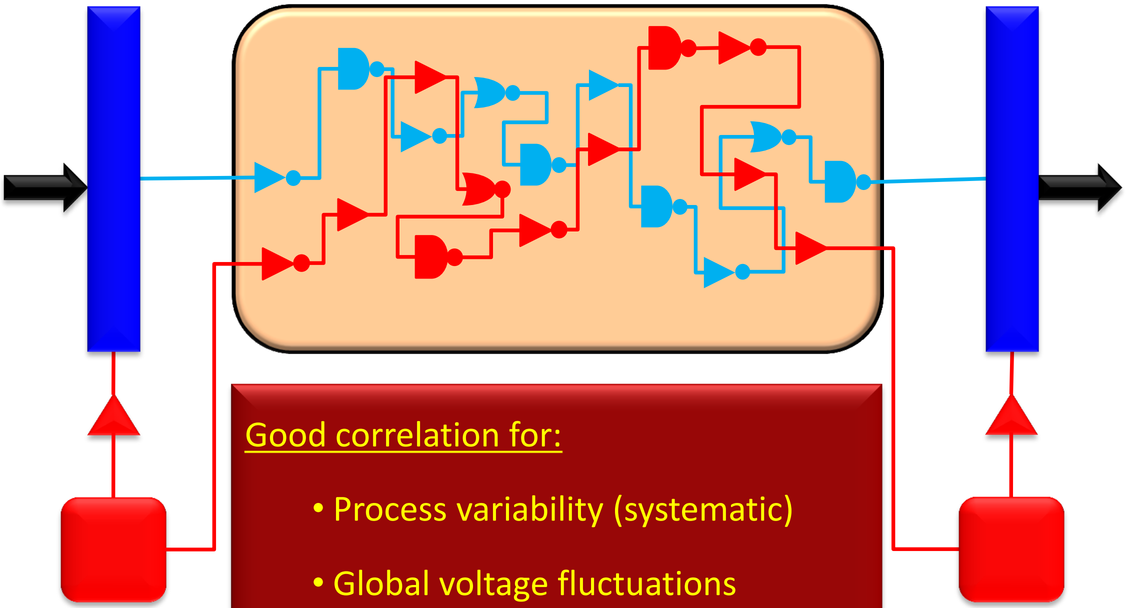
- Time-independent functional composability
  - Performance may be affected (but not functionality)



Tracking variability



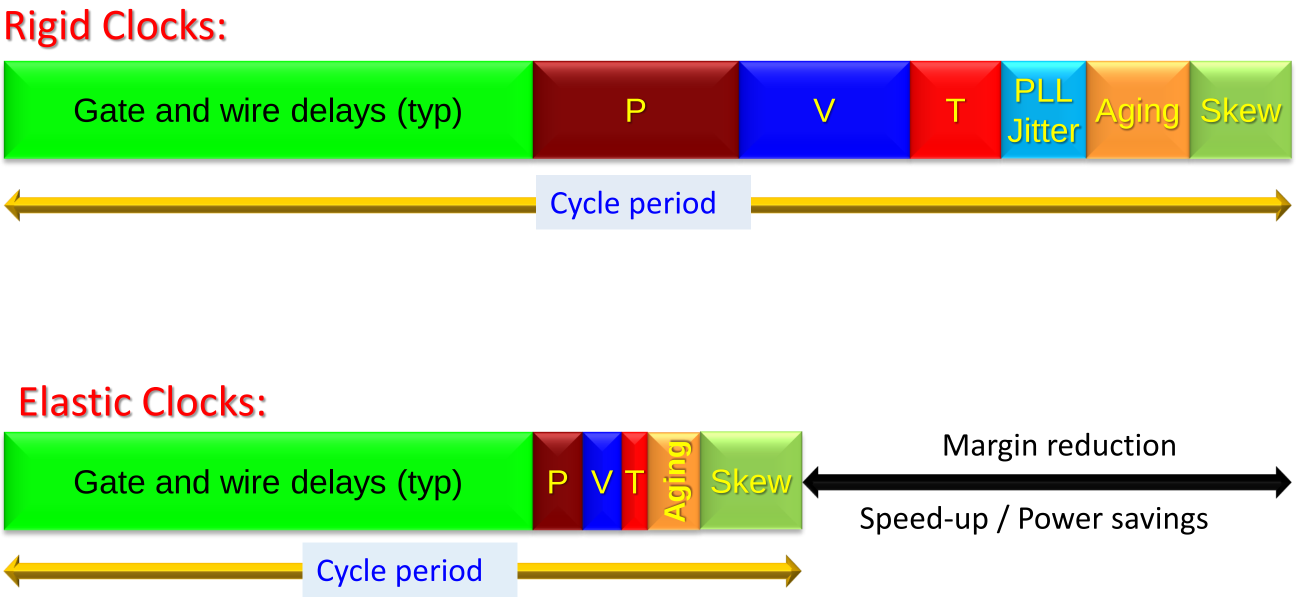
Tracking variability



Good correlation for:

- Process variability (systematic)
- Global voltage fluctuations
- Temperature
- Aging (partially)

Margins



Rigid Clocks:

Gate and wire delays (typ) | P | V | T | PLL Jitter | Aging | Skew

Cycle period

Elastic Clocks:

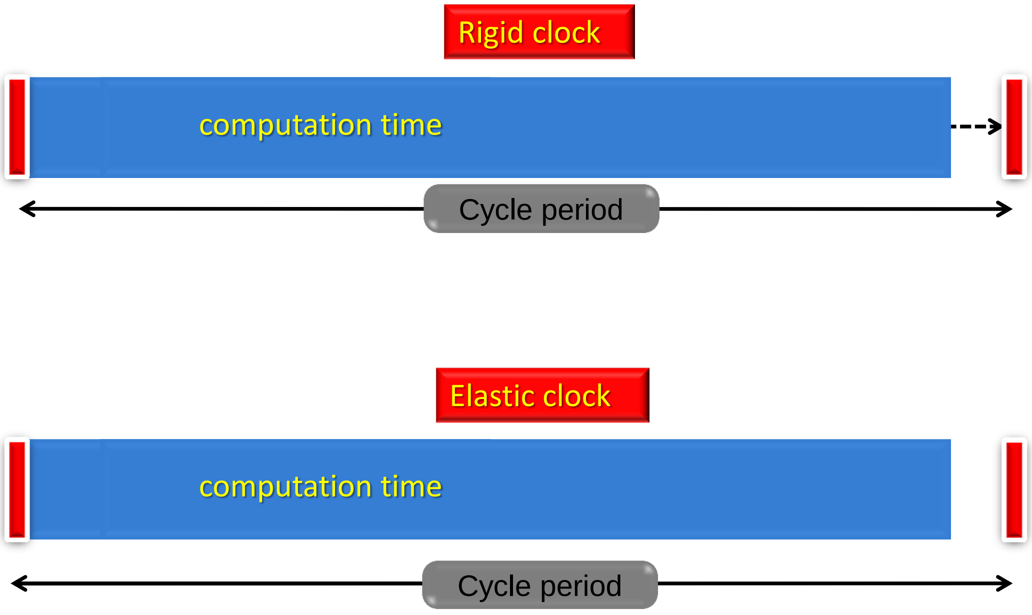
Gate and wire delays (typ) | P | V | T | Aging | Skew

Margin reduction

Speed-up / Power savings

Cycle period

Clock elasticity



Rigid clock

computation time

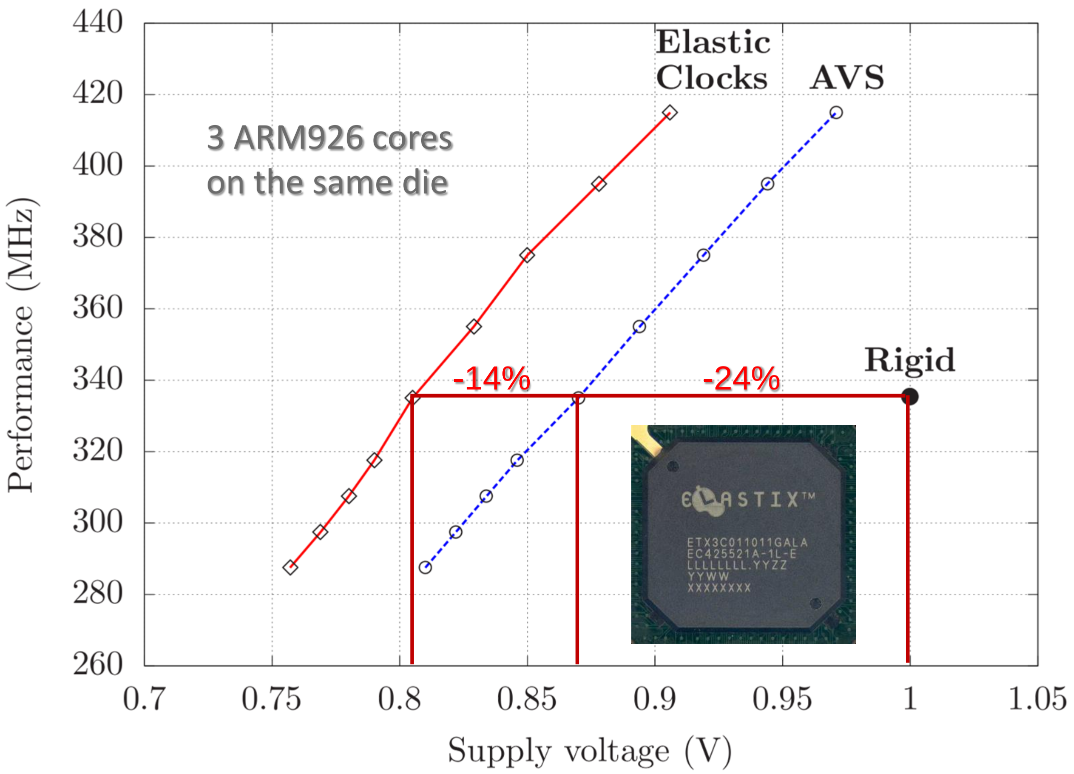
Cycle period

Elastic clock

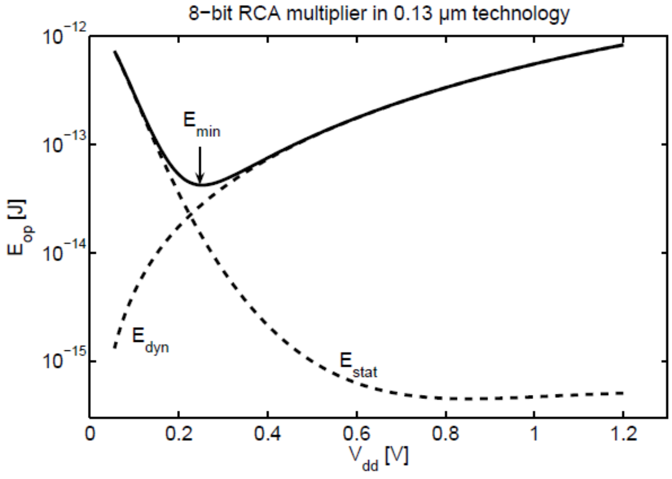
computation time

Cycle period

Voltage scaling and power savings



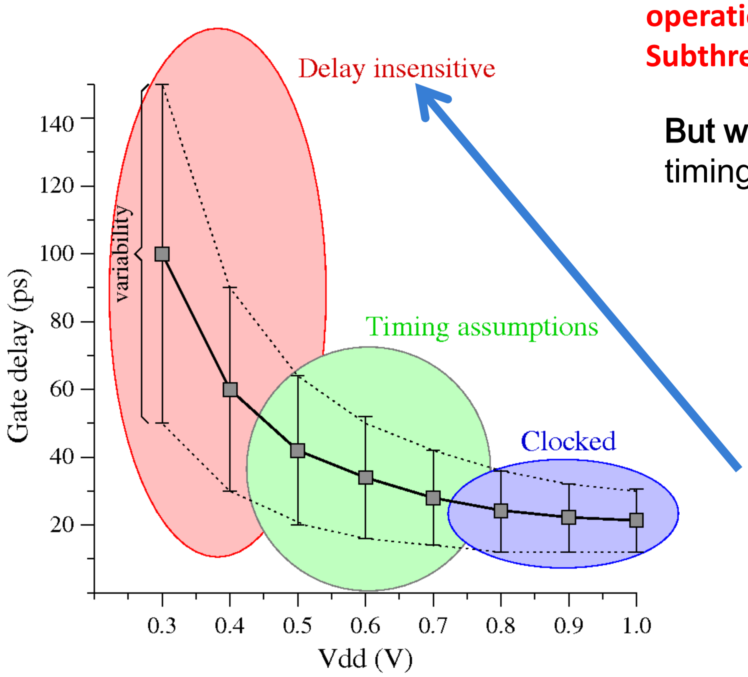
Scaling Vdd to near or sub-threshold



Optimising Vdd for energy per operation shifts us towards Subthreshold operation

Source: David Bol, Pushing Ultra-lLow-Power Digital Circuits into the Nanometer Era, PhD thesis,

Relationship with timing variability



Optimising Vdd for energy per operation shifts us towards Subthreshold operation

But we need to be more timing robust!

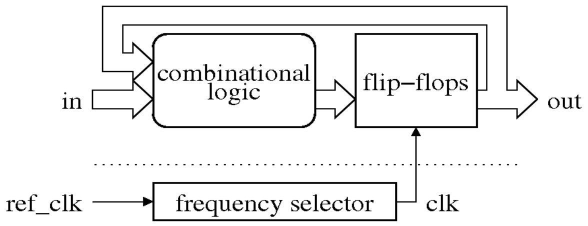
Technology node:
90nm

Source of variability analysis:
Yu Cao, Clark, L.T., 2007

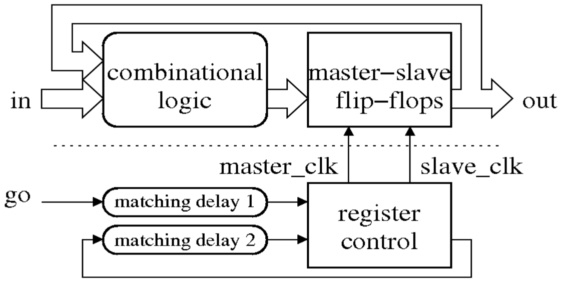
Example: 8-bit Booth’s Multiplier

- **Synchronous**
  - **Rigid 1GHz clock**
- **Frequency scaling**
  - **Tuned for 1GHz, 500MHz and 250MHz**
- **Asynchronous, bundled data**
  - **Extra control logic and delay lines**
- **Asynchronous, dual-rail**
  - **Double comb. logic and FF size (more leakage)**
  - **Extra completion detection and single-rail to dual-rail converters**
  - **Double switching activity (spacer/code-word)**

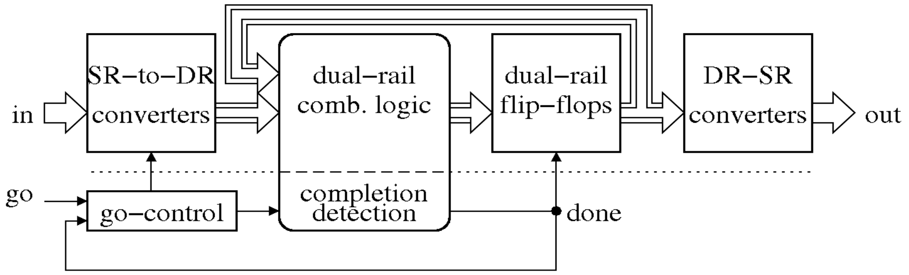
Benchmark Architectures



Adaptive frequency scaling

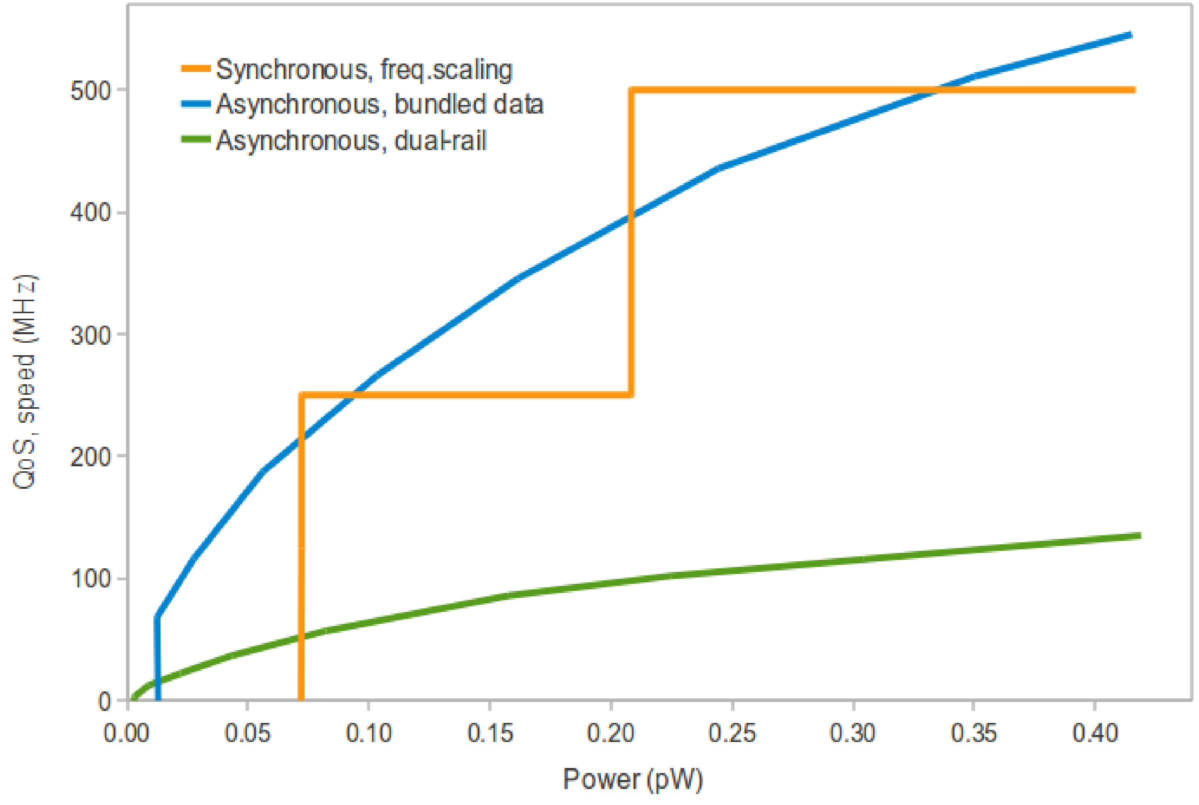


Bundled data

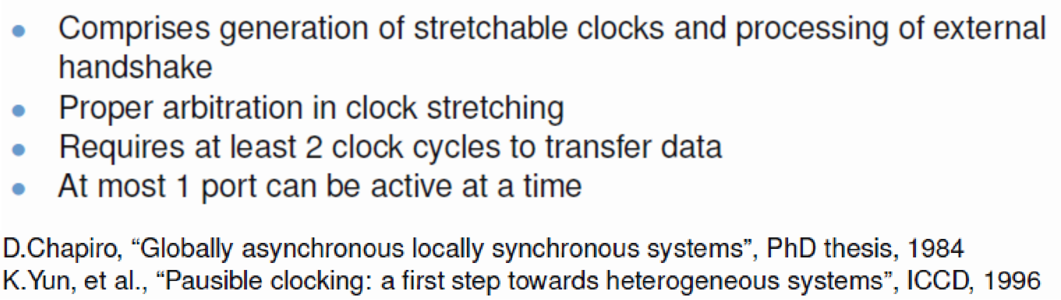


Dual-rail

Multiplier: Power-speed scaling



Globally Asynchronous Locally Synchronous (GALS)



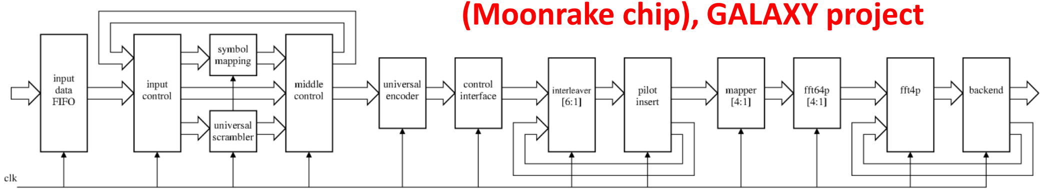
- Comprises generation of stretchable clocks and processing of external handshake
- Proper arbitration in clock stretching
- Requires at least 2 clock cycles to transfer data
- At most 1 port can be active at a time

D.Chapiro, “Globally asynchronous locally synchronous systems”, PhD thesis, 1984
K.Yun, et al., “Pausible clocking: a first step towards heterogeneous systems”, ICCD, 1996

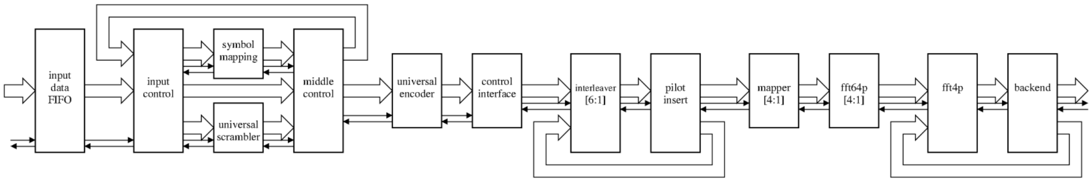
Clocking and interfaces: Towards GALS

Example from IHP Baseband processor design (Moonrake chip), GALAXY project

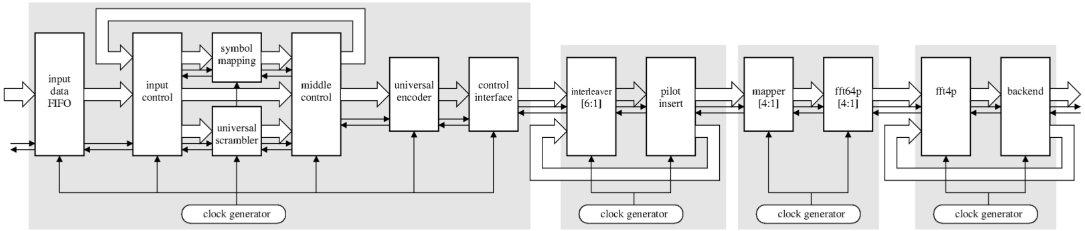
Synchronous



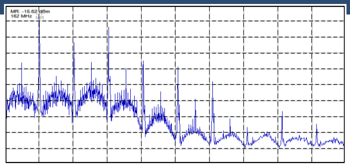
Asynchronous



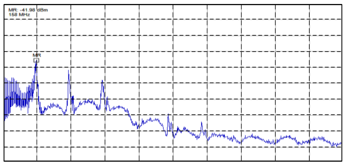
GALS



Measurements of Moonrake Chip



Amplitude of on-chip core VDD from SYNC TX



Amplitude of on-chip core VDD from GALS TX

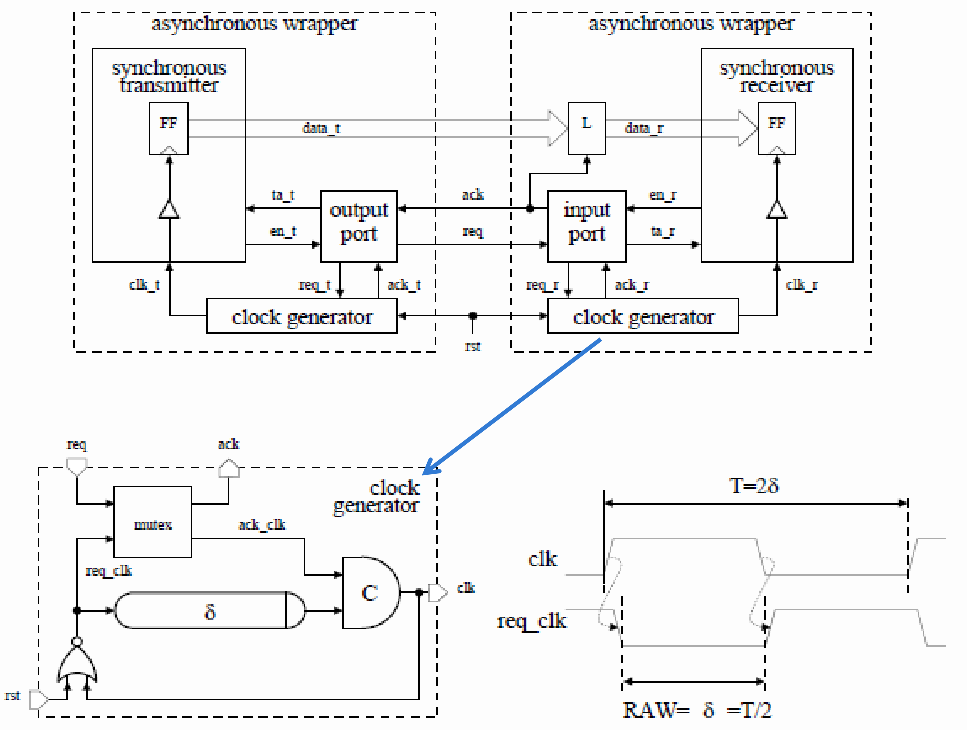
GALS has shown:

- **much better EMI profile**
- **improved power consumption and**
- **reduced area!**

	Area (mm^2)	Power Dissipation (mW)	Spectral amplitude of Core VDD (dBm)		
			1st peak	2nd peak	3rd peak
SYNC TX	2.33 (43.2%)	258	-15	-32	-23
GALS TX	2.22 (41.0%)	237	-41	-48	-53
Difference	**+4.7%**	**+8.2%**	**26dB**	**16dB**	**30dB**

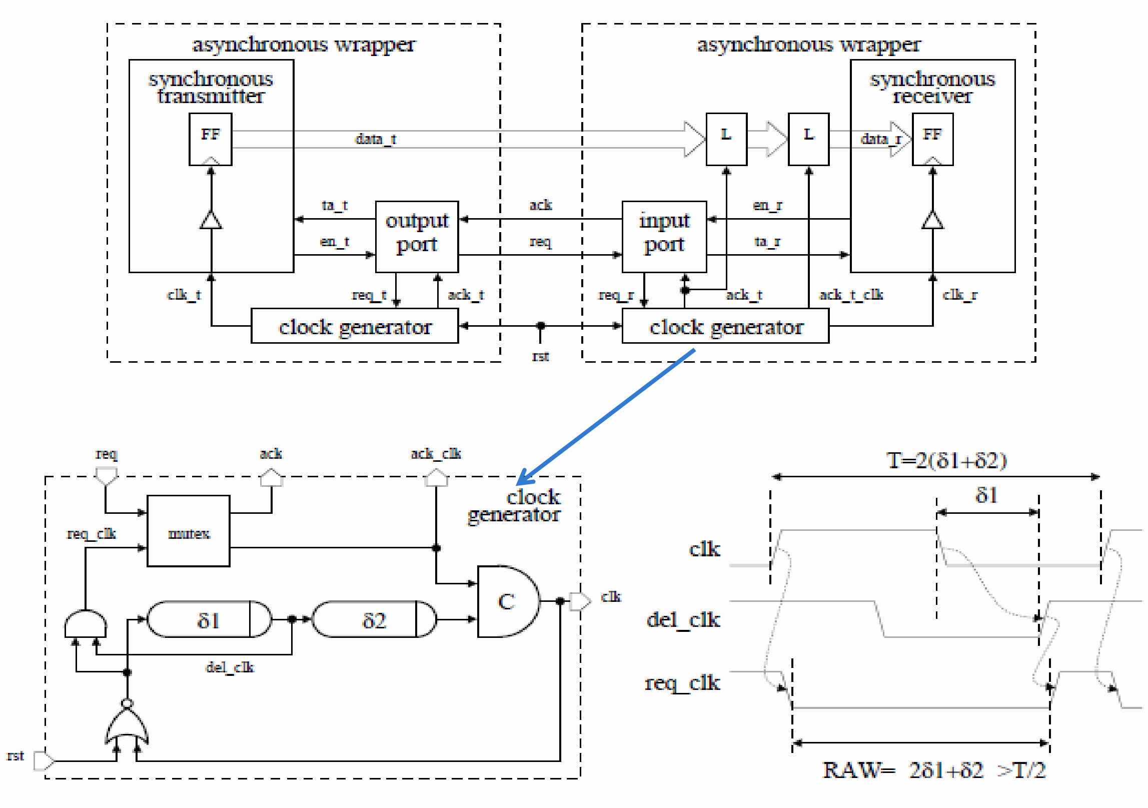
Source: M. Krstic et al. Evaluation of GALS Methods in Scaled CMOS Technology: Moonrake Chip Experience, IJERTCS, 3(4), 2012

GALS: asynchronous wrapper



Request-Acknowledge Window

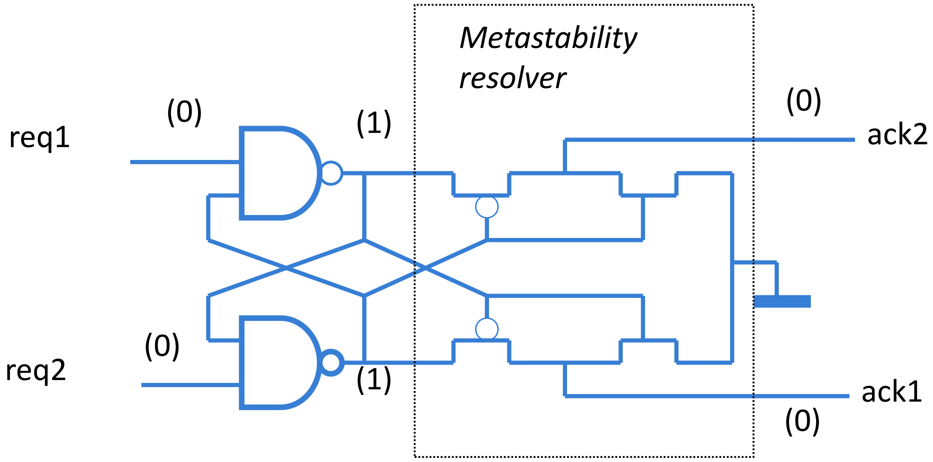
Optimization of async wrapper



Xin Fan, Milos Krstic, and Eckhard Grass, “Analysis and optimization of pausible clocking based GALS design,” ICCD 2009

Two-way arbiter (Mutual exclusion element)

Basic arbitration element: Mutex (due to Seitz, 1979)



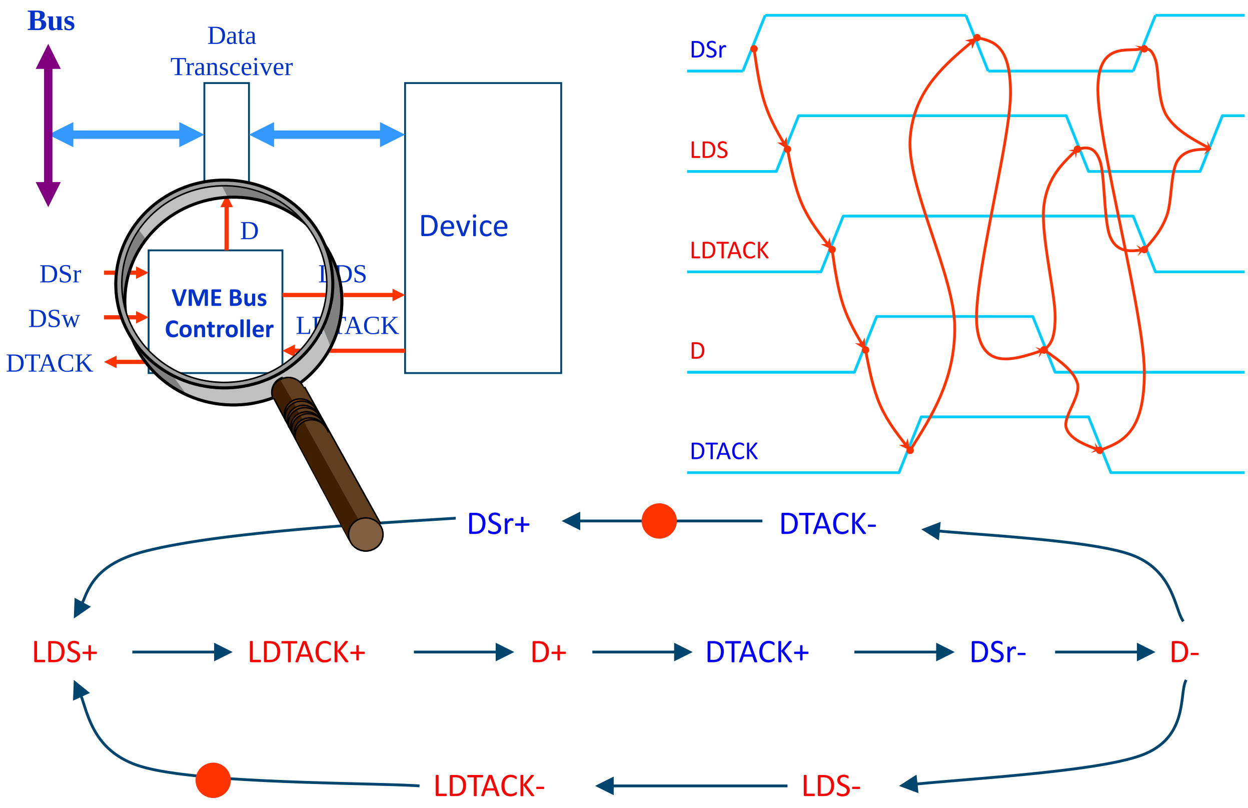
An asynchronous data latch with metastability resolver can be built similarly

Design Automation

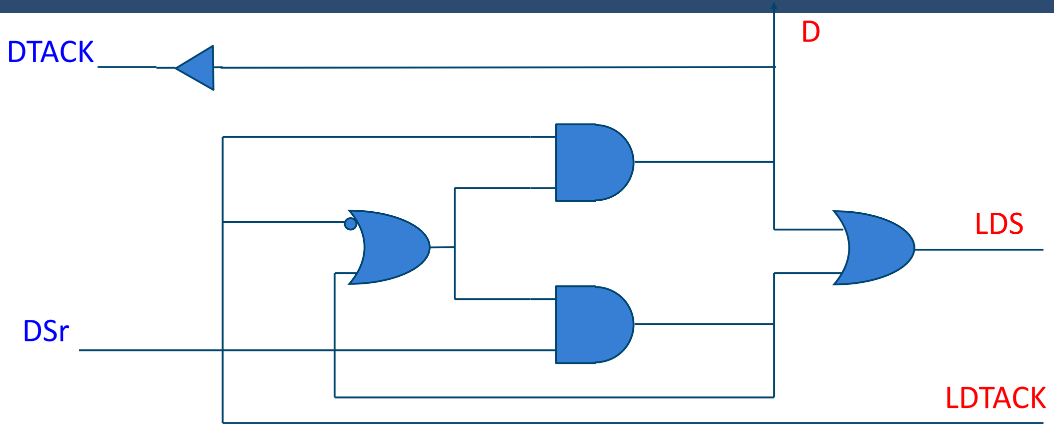
Design automation paradigms

- Synthesis of asynchronous controllers
  - Logic synthesis from Petri nets or asynchronous FSMs
- Syntax-directed translation
  - Correct-by-construction composition of handshake components
- De-synchronization
  - Automatic transformation from synchronous to asynchronous

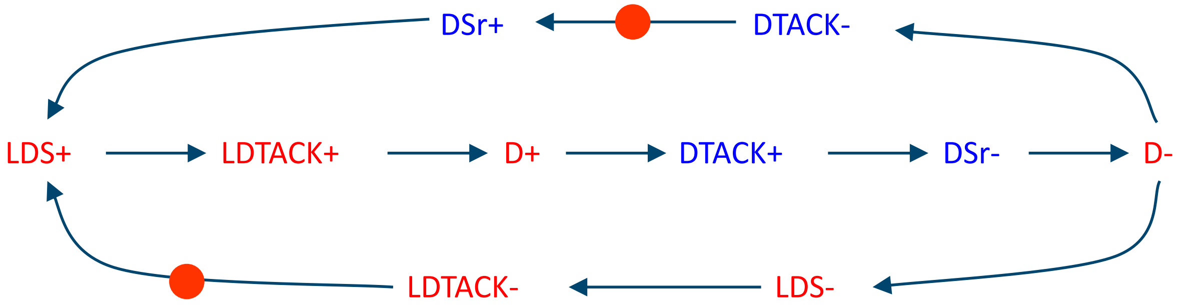
Synthesis of asynchronous controllers



Synthesis of asynchronous controllers



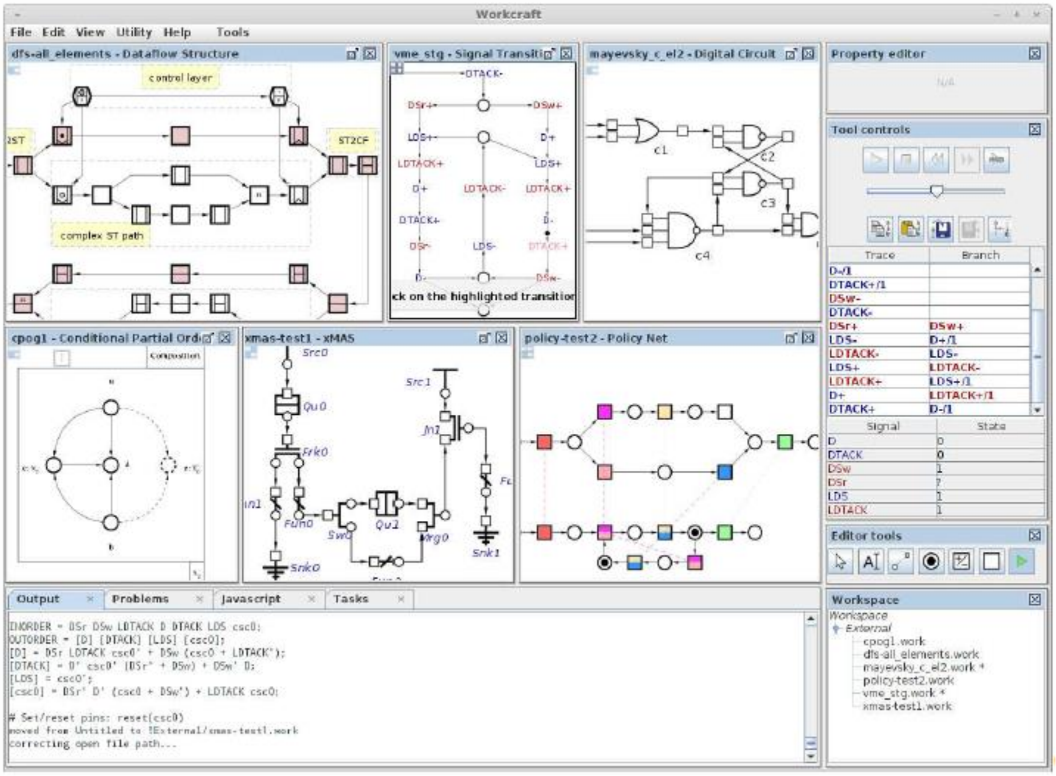
Example: Petrify



Workcraft tool

- Workcraft is a software package for graphical edit, analysis, synthesis and visualisation of asynchronous circuit behaviour
- Petrify, MPSAT, plus a few other tools are part of it as plug-ins
- It is based in Java tools
- Can be downloaded from http://workcraft.org/
- And installed in 10 minutes
- There is a simple to use tutorial for that

Workcraft


Workcraft
File Edit View Utility Help Tools
dfs-all_elements - Dataflow Structure
control layer
ST2CF
complex ST path
vme_stg - Signal Transitio
ck on the highlighted transitior
mayevsky_c_el2 - Digital Circuit
c1
c2
c3
c4
Property editor
N/A
Tool controls
cpog1 - Conditional Partial Ord
xmas-test1 - xMAS
Src0
Src1
Qu0
Frk0
Jn1
Fun0
Sw0
Qu1
Mrg0
Snk0
Snk1
policy-test2 - Policy Net
Editor tools
Output
Problems
Javascript
Tasks
Workspace

```
INORDER = DSr DSw LDTACK D DTACK LDS csc0;
OUTORDER = [D] [DTACK] [LDS] [csc0];
[D] = DSr LDTACK csc0' + DSw (csc0 + LDTACK');
[DTACK] = D' csc0' (DSr' + DSw) + DSw' D;
[LDS] = csc0';
[csc0] = DSr' D' (csc0 + DSw') + LDTACK csc0;

# Set/reset pins: reset(csc0)
moved from Untitled to !External/xmas-test1.work
correcting open file path...
```

Trace	Branch
D-/1	
DTACK+/1	
DSw-	
DTACK-	
DSr+	DSw+
LDS-	D+/1
LDTACK-	LDS-
LDS+	LDTACK-
LDTACK+	LDS+/1
D+	LDTACK+/1
DTACK+	D-/1

Signal	State
D	0
DTACK	0
DSw	1
DSr	?
LDS	1
LDTACK	1

Workspace
- External
  - cpog1.work
  - dfs-all_elements.work
  - mayevsky_c_el2.work *
  - policy-test2.work
  - vme_stg.work *
  - xmas-test1.work

Conclusions

- **Asynchrony offers flexibility in time**
  - **Modularity**
  - **Dynamic adaptability**
  - **Tolerance to variability**
- **Better optimization of power/performance**
  - **Facilitates near and sub-threshold modes**
  - **Facilitates easier power-gating**
- **Why isn't it an important trend in circuit design?**
  - **Lack of commercial EDA support (timing sign-off)**
  - **Designers do not feel comfortable with "unpredictable" timing**
  - **Other aspects: testing, verification, …**
- **De-synchronization might be a viable solution**

Some references

- **General Async Design:** J. Sparsø and S.B. Furber, editors. *Principles of Asynchronous Circuit Design*, Kluwer Academic Publishers, 2001. (electronic version of a tutorial based on this book can be found on: http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/855/pdf/imm855.pdf
- **Async Control Synthesis:** J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. *Logic Synthesis of Asynchronous Controllers and Interfaces*. Springer-Verlag, 2002. (Petrify software can be downloaded from: http://www.lsi.upc.edu/~jordicf/petrify/)
- **Arbiters and Synchronizers:** D.J. Kinniment, Synchronization and Arbitration in Digital Systems, Wiley and Sons, 2007 (a tutorial on arbitration and synchronization from ASYNC/NOCS 2008 can be found: http://async.org.uk/async2008/async-nocs-slides/Tutorial-Monday/Kinniment-ASYNC-2008-Tutorial.pdf)
- **Asynchronous on-chip interconnect:** John Bainbridge, Asynchronous System-on-Chip Interconnect, BCS Distinguished Dissertations, Springer-Verlag, 2002 (electronic version of the PhD thesis can be found on: http://intranet.cs.man.ac.uk/apt/publications/thesis/bainbridge00_phd.php)

DATE 16

DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

Metastability and Clock Domain Crossing

David M. Zar

Blendics, Inc.

DATE 16
DESIGN, AUTOMATION & TEST IN EUROPE
14 – 18 March, 2016 · ICC · Dresden · Germany
The European Event for Electronic
System Design & Test

Modern Clocking Strategies:
Metastability and
Clock Domain Crossing

David M. Zar

Blendics, Inc

BlendICS
BLENDED INTEGRATED CIRCUIT SYSTEMS

Overview

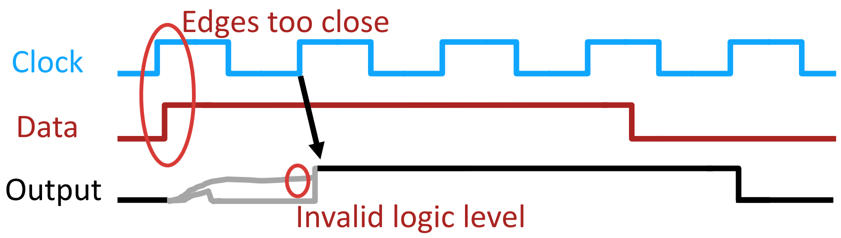
- **Clock Domain Crossing (CDC) Fundamentals**
- **How Synchronizers Can Fail**
- **Common Synchronizer Structures**
- **Analysis of Synchronizer Failure**
- ***Good* Synchronizers**

Clock Domain Crossing (CDC) Fundamentals

- A CDC occurs whenever a signal crosses from one clock domain to another
- Any data input not synchronous to a clock is a CDC, in general (e.g. an asynchronous input)
- In modern SOCs, there could be hundreds, or even thousands, of CDCs ocurring between any number of disparate clock domains
- If any of these CDCs are not dealt with, the circuit may fail due to a synchronization failure:
  - Metastability failure or
  - Under/over sampling failure

Metastability Failure

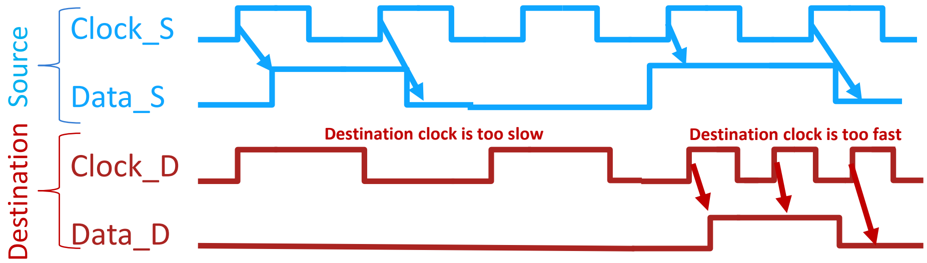
- A violation of input timing at the synchronizer can cause the output of the synchronizer to go metastable:



- During the transition, the output must not be sampled at the destination until it has resolved

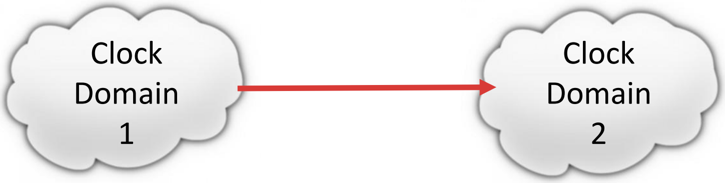
Sampling Failure

- **If the clock frequency of the receiver is too fast/slow, a sampling failure may occur:**

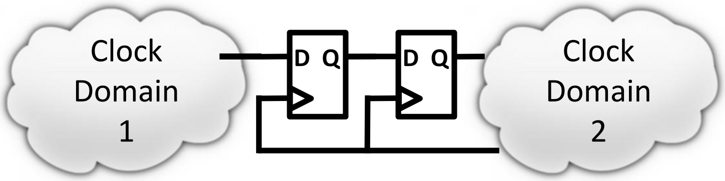


- **Destination clock too slow → data lost**
- **Destination clock too fast → data repeated**

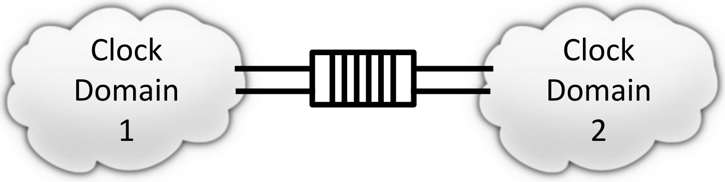
Properly Dealing With CDC



Here we have a CDC with no synchronization. Nobody does this, right?

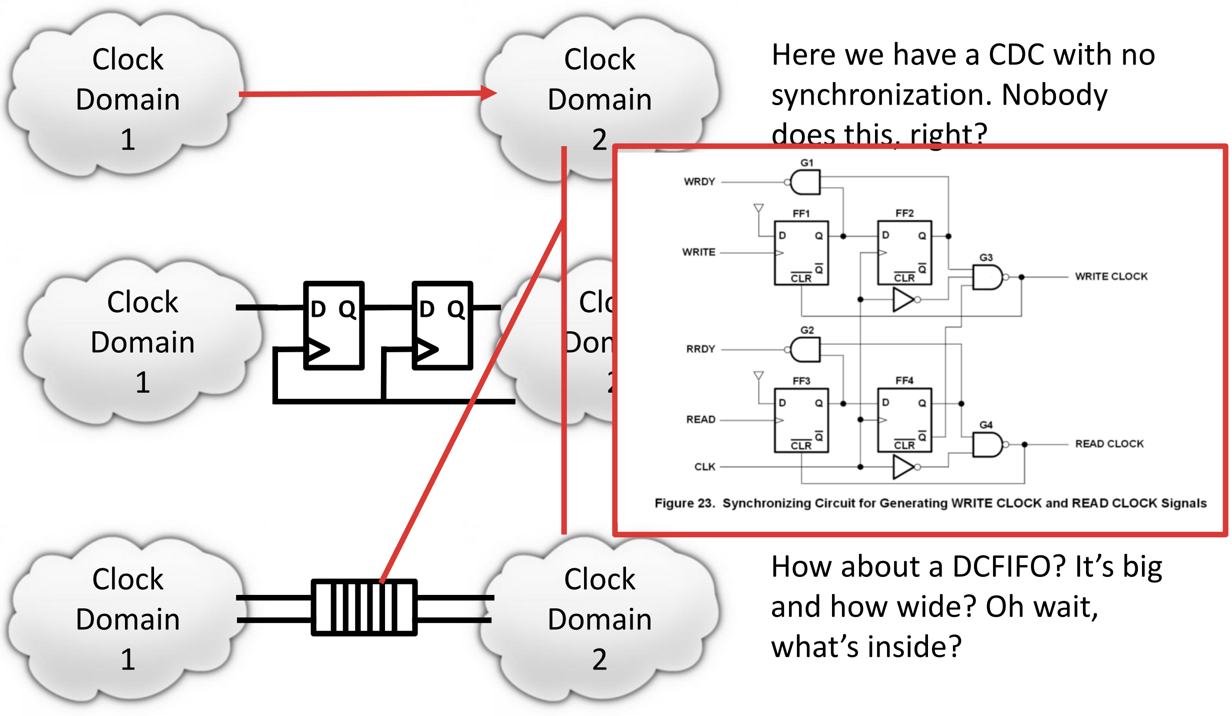


Use the 2-stage synchronizer we all learned about. But what is being synchronized? How long can you wait? Are two stages enough?



How about a DCFIFO? It's big and how wide? Oh wait, what's inside?

Properly Dealing With CDC



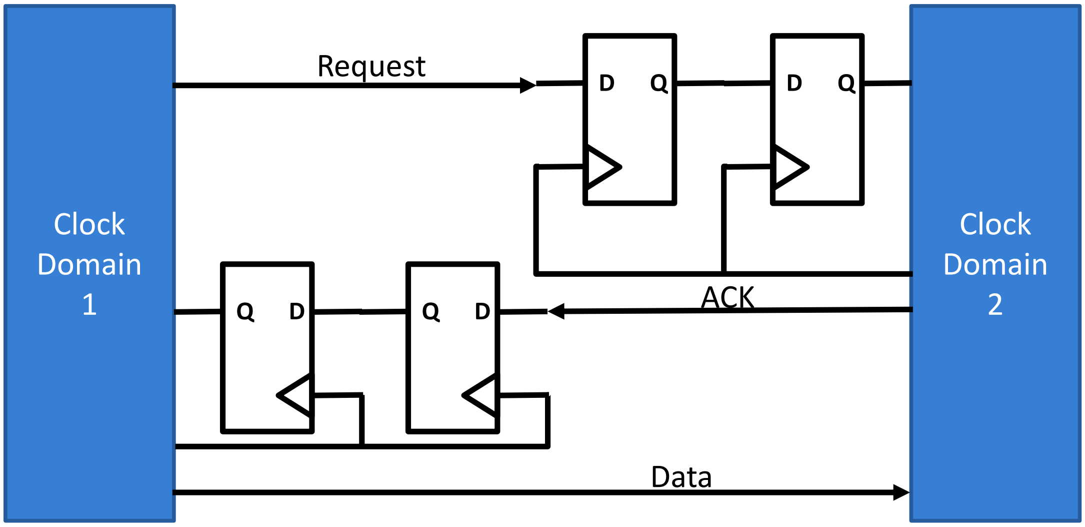
Here we have a CDC with no synchronization. Nobody does this, right?

Figure 23. Synchronizing Circuit for Generating WRITE CLOCK and READ CLOCK Signals

How about a DCFIFO? It's big and how wide? Oh wait, what's inside?

Handshaking Solves the CDC Problem! [?]

- So it seems until you look under the covers:



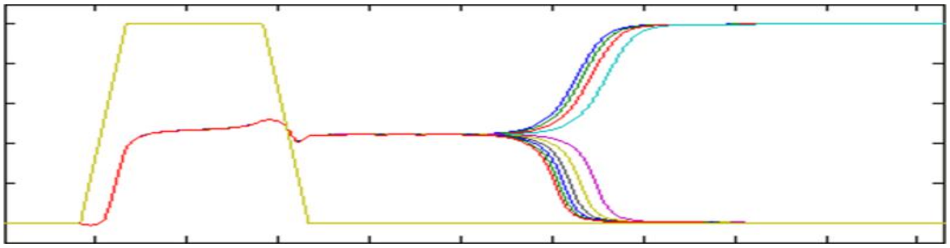
- This still can have a synchronizer failure!

CDC Identification

- The good news is that identifying CDCs is *easy*:
  - Designer should know, right:
    - Static analysis – did you find them all?
    - Power islands – dynamic powering up/down?
    - Reset – where does it go and when?
  - Helpful tools:
    - Blue Pearl ACE
    - Mentor Graphics Questa CDC
    - Real Intent Meridian
    - Synopsys SpyGlass CDC (Formerly Atrenta)
    - vSync Circuits vChecker
    - Others…

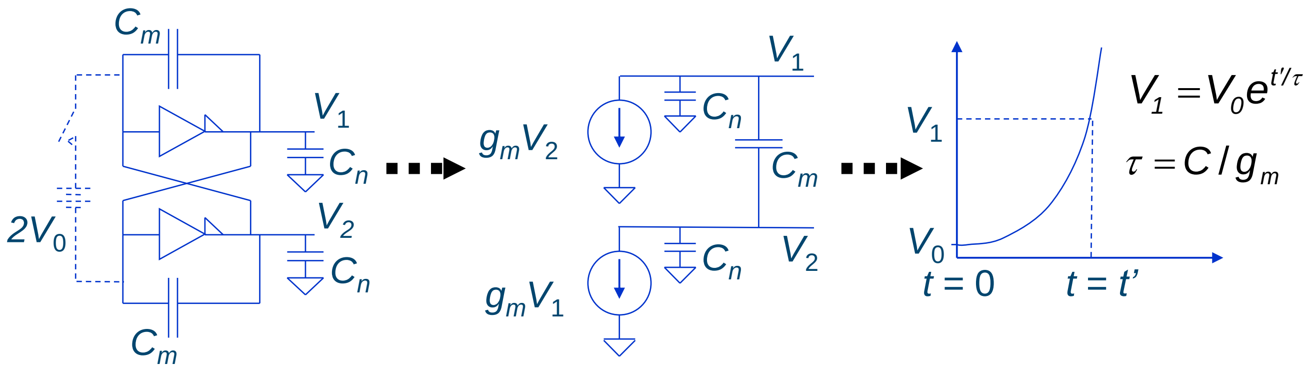
Analysis of Synchronizer Failure

- If you have removed all unnecessary CDCs and then *properly* added synchronization at the remaining CDCs, your synchronizer may still fail
- What *is* a synchronizer failure?
  - A synchronizer failure is when the output fails to resolve to a valid logic level by a certain point in time (typically clock period minus setup time)
  - Is caused by the inputs changing at the *wrong time* (like violating setup or hold time at a flip-flop)



How Does a Latch Resolve?

- Flip-flop is too complex, but is made up of latches; use a latch!
- Now small signal analysis finds the relationship between output voltage (V_1) and initial output voltage (V_0)

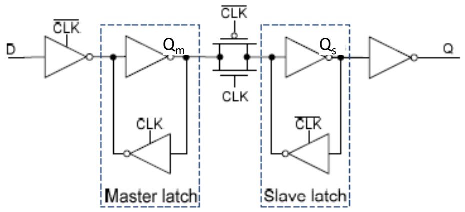


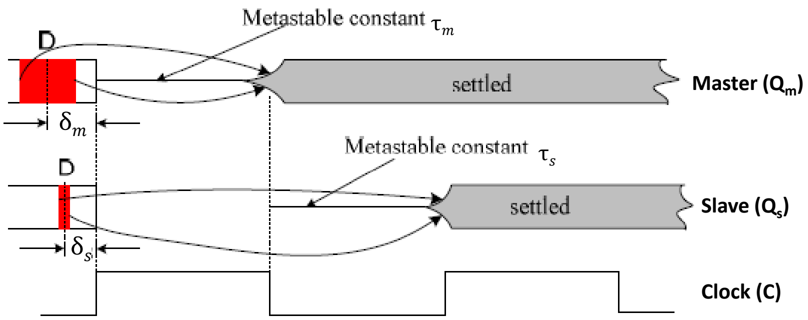
For V_0 small

- τ is a time constant related to the time it takes to get from the metastable voltage to a valid logic voltage level

Failing Master-Slave Flip-Flop

- Given a data-clock offset (δ) in the red window for D, output Q_m resolves near or past the falling edge of C
- A narrower red window causes output Q_s to resolve near or past next rising edge of C. (We define T_w as the largest such window that causes the output to "push out" past the nominal clock-output delay)
- When C is high, the settling behavior at Q_m is a function of τ_m
- When C is low, Q_s is a function of τ_s



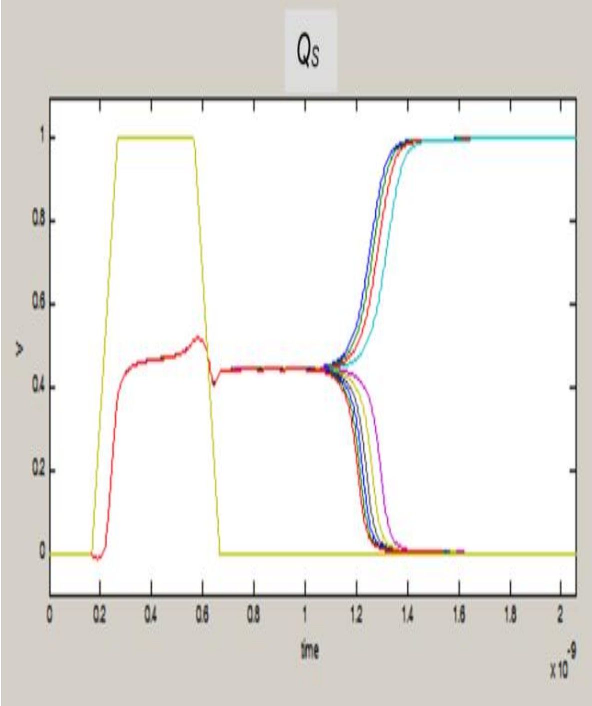


CDC Failure Analysis

- So given that you can identify CDCs in circuits, and
- Given that you cannot guarantee they will always work,
- What are you to do?
  - Give up?
  - Pretend the problem won't happen because you used a two flip-flop synchronizer?
  - Design *good* synchronizers, use them, and analyze their failure rates to be sure you are at acceptable levels?

Failure Analysis

- For a given synchronizer
  - We can determine some intrinsic parameters that govern how fast it resolves (τ and T_w)
  - We can determine some extrinsic parameters that describe the environment (clock frequency, data arrival rates, etc.)
  - We can then determine the failure rate for any particular synchronizer



Failure Rate/MTBF/Pr(safe)

- We commonly use Mean Time Between Failures (MTBF) when speaking of synchronizer failures:

$$MTBF = \frac{e^{S/\tau}}{T_w f_d f_c}$$

***S* is the settling time by which the signal must be resolved,**
***f*$_d$ is the data rate and *f*$_c$ is the sampling clock rate**

- Failure Rate = 1/MTBF (also known as Failures In Time, or FIT)
- For N such synchronizers in C chips shipped, the total Failure Rate is NC/MTBF (or many orders of magnitude higher than for a single synchronizer!)
- Pr(safe) – Probability of Being Safe – measures the probability that all units in the field perform safely through the average lifetime of said units:

$$Pr(safe) = \exp\left[\frac{-NCL}{MTBF}\right]$$

Where L is the average lifetime of a unit

MTBF/Failure Rate Fallacies

- **MTBF is not how long it will take, on average, for your circuit to fail.**
  - *MTBF ~~of 100 years is fine for one failure in 100~~ years and is accep~~table for my calculator.~~*
- **MTBF is not the expected lifetime of your device.**
  - *MTBF ~~of 5 years for a cell phone is fine if nobody~~ uses it for that lo~~ng.~~*
- **MTBF is not the service time of your device.**
  - *MTBF ~~of 2 years for the IV pump is acceptable~~ if we will service it ever~~y year, test it, etc.~~*
- **For example, an MTBF of 10 years implies that:**
  - **In the first year, we expect ~10% of our devices to fail,**
  - **After 10 years, we expect ~63% of our devices to fail!**

Know Your MTBF

- It is imperative to know the MTBF for your synchronizer:
  - Turns out, as feature size goes below about 65 nm, τ starts to increase (lowering MTBF) relative to FO4 delay

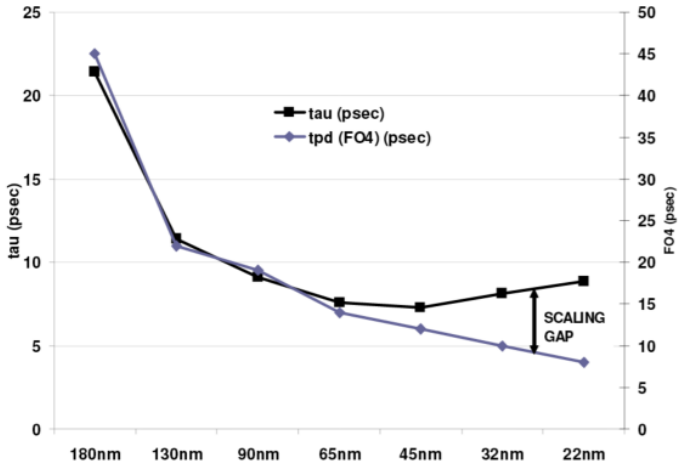
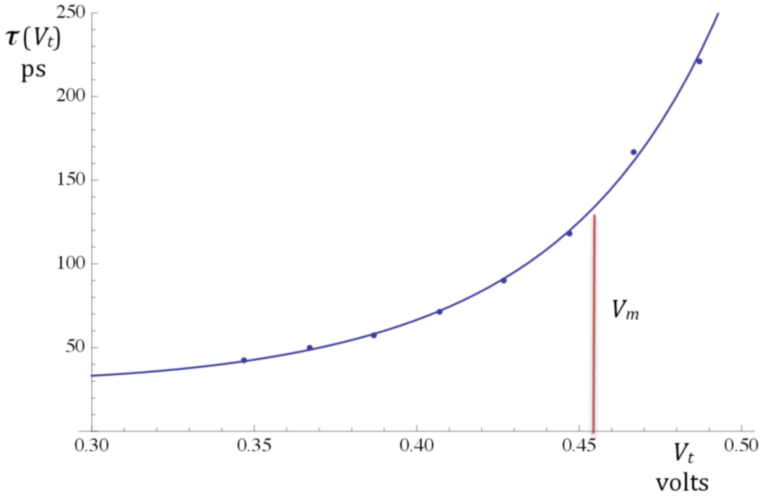


Figure 20: The τ degradation effect

From Beer, et al. *Devolution of Synchronizers*

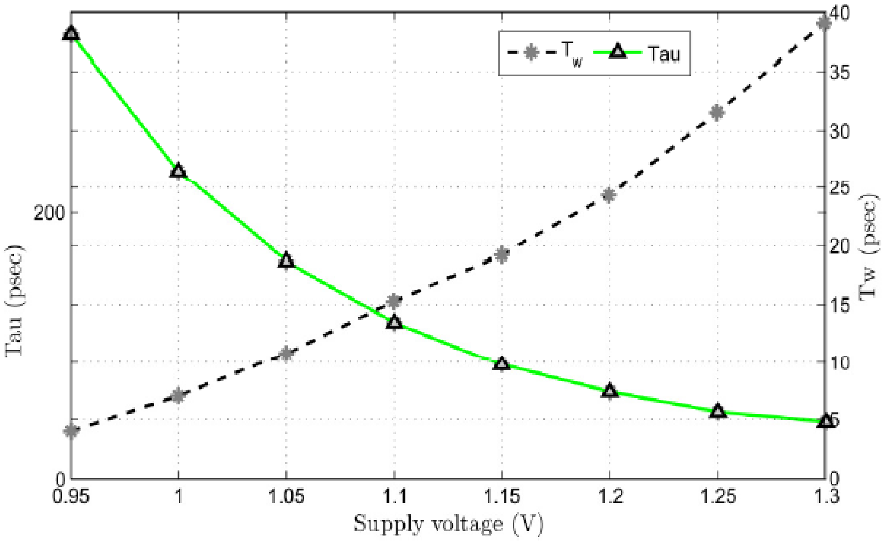
Know Your MTBF (2)

- Process variability affects MTBF
  - For example, τ is sensitive to V_t



Know Your MTBF (3)

- V_{dd} matters, too



Know Your MTBF (4)

- The duty cycle of the clock can affect MTBF in master-slave type flip-flops/synchronizers
  - Typically, τ is assumed to be only a function of the master (or slave) of a master-slave FF; This is not the case.

$$\tau_{eff} = \left(\frac{\alpha}{\tau_m} + \frac{(1-\alpha)}{\tau_s}\right)^{-1}$$

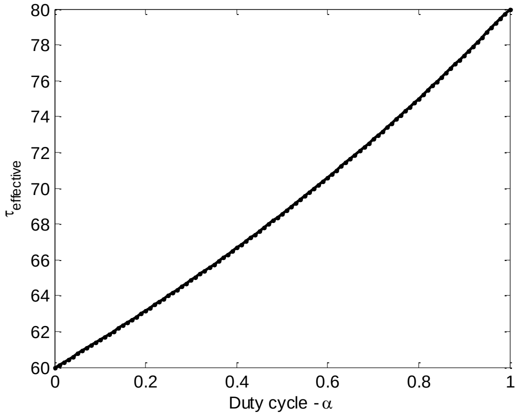
- Where τ_{eff} is the *effective* τ of the device.
- α is the duty cycle (e.g. 0.4 means the clock is high for 40% of the clock period)
- τ_m is the τ for the master latch; τ_s is the τ for the slave latch

Effective τ Example

- A τ_{eff} will vary between the τ values of the master and the slave, being their harmonic mean at a 50% duty cycle when the two τ values are close to each other.
- Things are more interesting when the τ values are very different

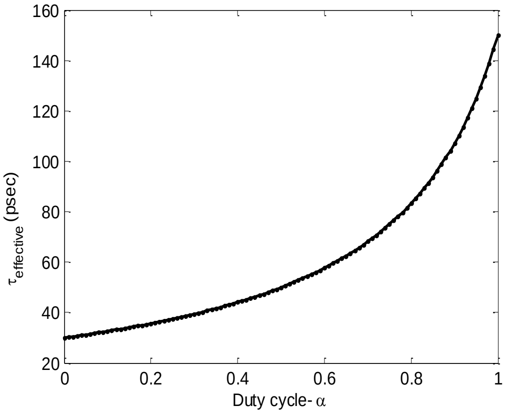
Nearly the Same

$\tau_m = 80$ ps, $\tau_s = 60$ ps



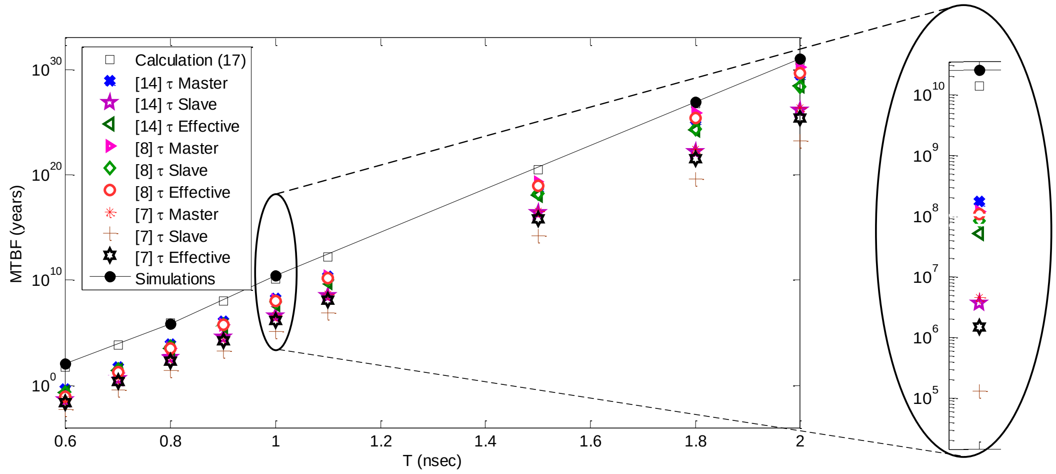
Very Different

$\tau_m = 150$ ps, $\tau_s = 30$ ps



Know Your MTBF (5)

- **Finally, most published MTBF models are not correct for multi-stage FF-based synchronizers (see Beer, *MTBF Bounds for Multistage Synchronizers*)**

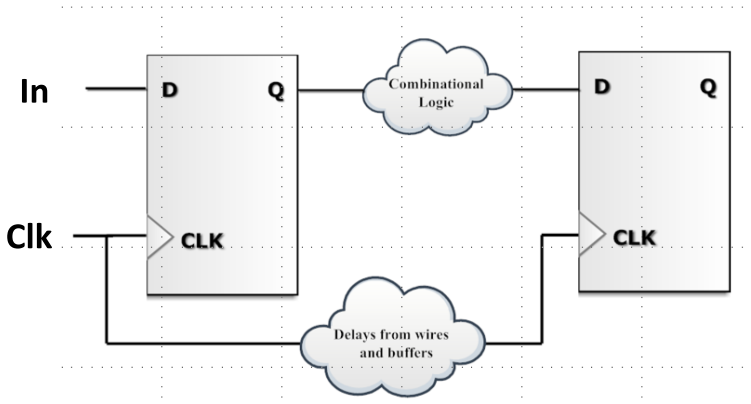


Good Synchronizers

(Or… Why You Should Never Use a Data Flip-Flop as a Synchronizer)

- Data Flip-Flop
  - Used for temporary storage of data
    - Prevent data values from corruption during a clock cycle
    - Hold data values for multiple clock cycles
  - Designed for deterministic cycle-to-cycle operation
    - Implies large setup/hold times
- Synchronizer Flip-Flop
  - Used to minimize *Pr(failure)*
    - Data/clock may arrive at any time which may cause a setup/hold violation at a following data flip-flop
  - Needs to preserve data transition sequence
    - No guarantee of deterministic cycle-to-cycle timing

Data Flip-Flops Vs. Synchronizer Flip-Flops

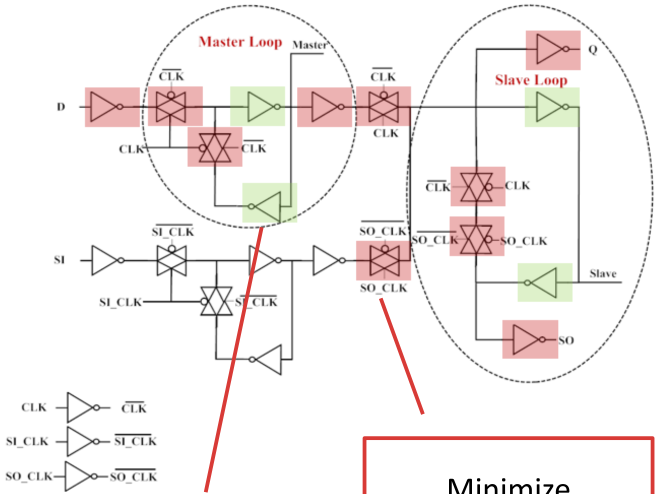


Different performance characteristics to optimize based on FF use:

	t_{pd}	t_{su}	t_h	τ	T_w
Data FF	minimize	minimize	0	-	-
Synchronizer FF	-	-	0	minimize	minimize

Good Synchronizing Scan D Flip-Flop (SDFF)

- **From Cox, *Synchronization and Data Flop-Flops are Different***
- ***Green devices are what you want to optimize***
- ***Red devices are what should be minimized***

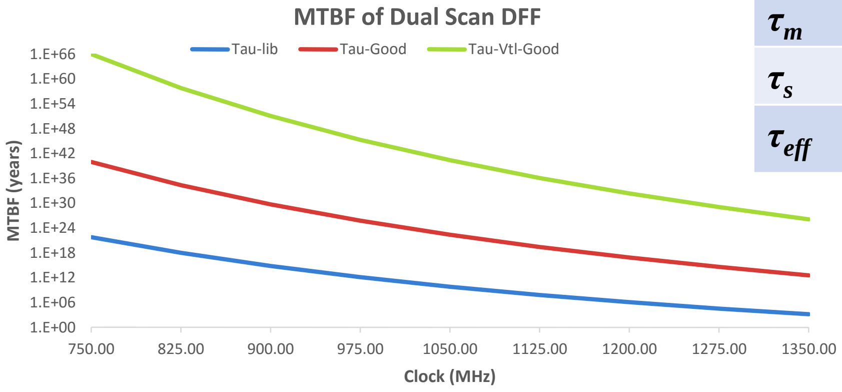


Maximize gain-bandwidth product at green devices

Minimize capacitance at red devices/nodes

How Good is a *Good* Synchronizer?

- **A SDFF cell from a library was compared to this *good* cell (at 45 nm) (VTG uses standard V_t devices whereas VTL uses low-V_t devices)**



	Library VTG (ps)	Good VTG (ps)	Good VTL (ps)
τ_m	19	14	10
τ_s	55	31	19
τ_{eff}	28	19	13

Design for Good Synchronization

- You want a large MTBF
- So you need small τ
  - Use low-V_t transistors if you can
  - Use largest V_{dd} you can
  - Reduce capacitance on nodes in regenerative loops - including output node(s)
  - Design a custom synchronization flip-flop (as in reference: *Synchronization and Data Flop-Flops are Different*)
- BEWARE: Many *clever* synchronizing flop-flop designs have new issues. Why be complicated when you can analyze?!

So…

- To properly handle the CDC in your circuit, you need to identify all such occurrences
- Then you need to ensure a proper synchronization mechanism is used
- Then you need to determine the MTBF/FIT for each CDC
- Finally, you can calculate the MTBF/FIT/Pr(safe) for the entire circuit based on the individual MTBF/FIT, the number of each, the length of service and the number of copies you expect to be in use

Questions

???

References

- Beer, S. et al., *MTBF Bounds for Multistage Synchronizers,* ASYNC2013, http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6546190&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6546190
- Beer, S et al., *The Devolution of Synchronizers,* *http://kolodny.eew.technion.ac.il/files/2013/08/The-devolution-of-synchronizers-ASYNC-2010.pdf*
- *Beer, S et al., Supply Voltage and Temperature Influence in Circuit Synchronization,* DATE2013, http://webee.technion.ac.il/~ran/papers/BeerVoltageTempVariations.pdf
- Cox, J, *Beware of Parameter Variability in Clock Domain Crossings,* SemiWiki.com, 2015 https://www.semiwiki.com/forum/content/4625-beware-parameter-variability-clock-domain-crossings.html?s=30b01668a0a490331fd3f496f7a4f86a
- Cox, J, et al., *Synchronization and Data Flop-Flops are Different,* ASYNC 2015, http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7152686&url=httpA%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D7152686
- Cox, J, *Synchronizer Reliability Metrics, 2014,* SemiWiki, https://www.semiwiki.com/forum/content/3294-synchronizer-reliability-metrics.html
- Golson, S, *Synchronization and Metastability,* 2014, http://blendics.com/wp-content/uploads/2015/06/golson_snug14.pdf

DATE 16

DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

Adaptive Clocking

Jordi Cortadella

Universitat Politècnica de Catalunya

DATE 16

DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

Modern Clocking Strategies: Adaptive Clocking

Jordi Cortadella
Universitat Politècnica de Catalunya

Outline

- Sources of variability
  - Static/dynamic, local/global
- Reducing Margins for Static and Slow Variability
  - Binning
  - Voltage-Frequency Scaling
- Reducing Margins for Fast Variability
  - Resilient circuits
  - Adaptive Clocks
- Reactive Clocks and GALS

The cost of variability

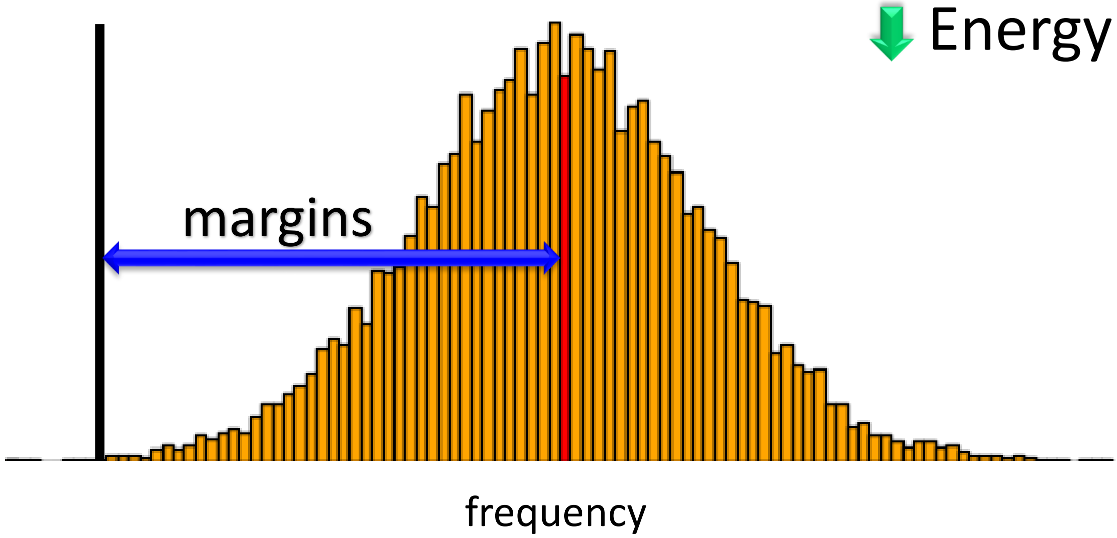
Goal:

Reduce margins

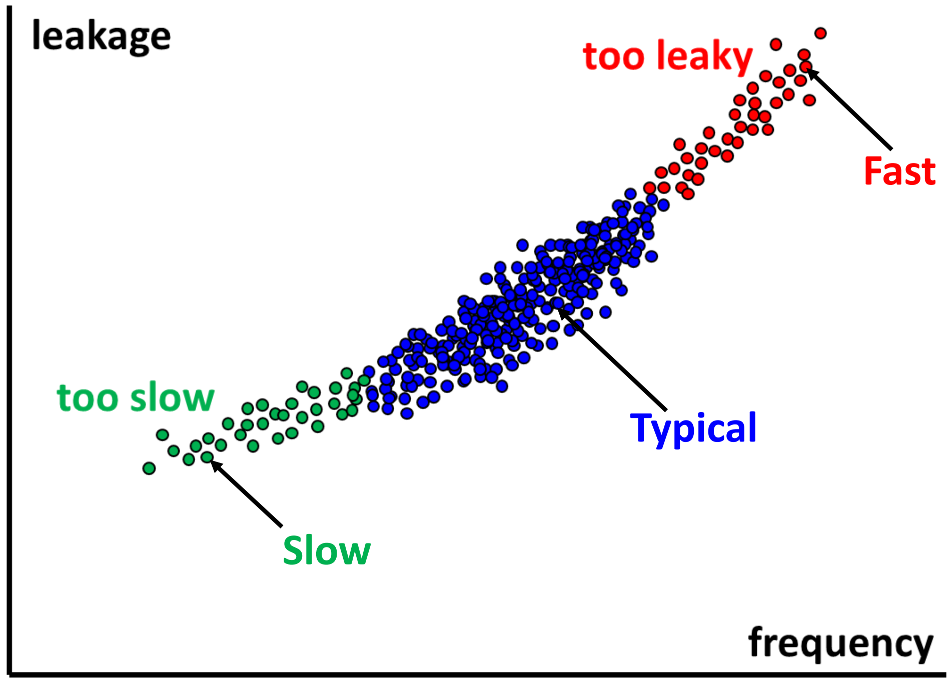
Impact:

↑ Speed

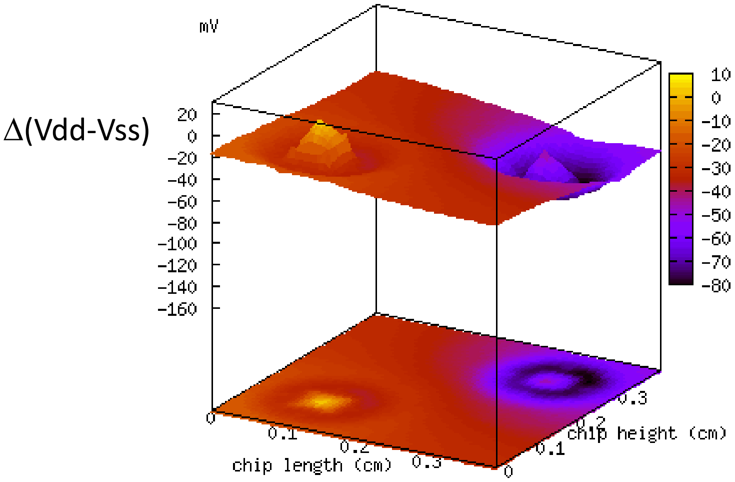
↓ Energy



Process variability



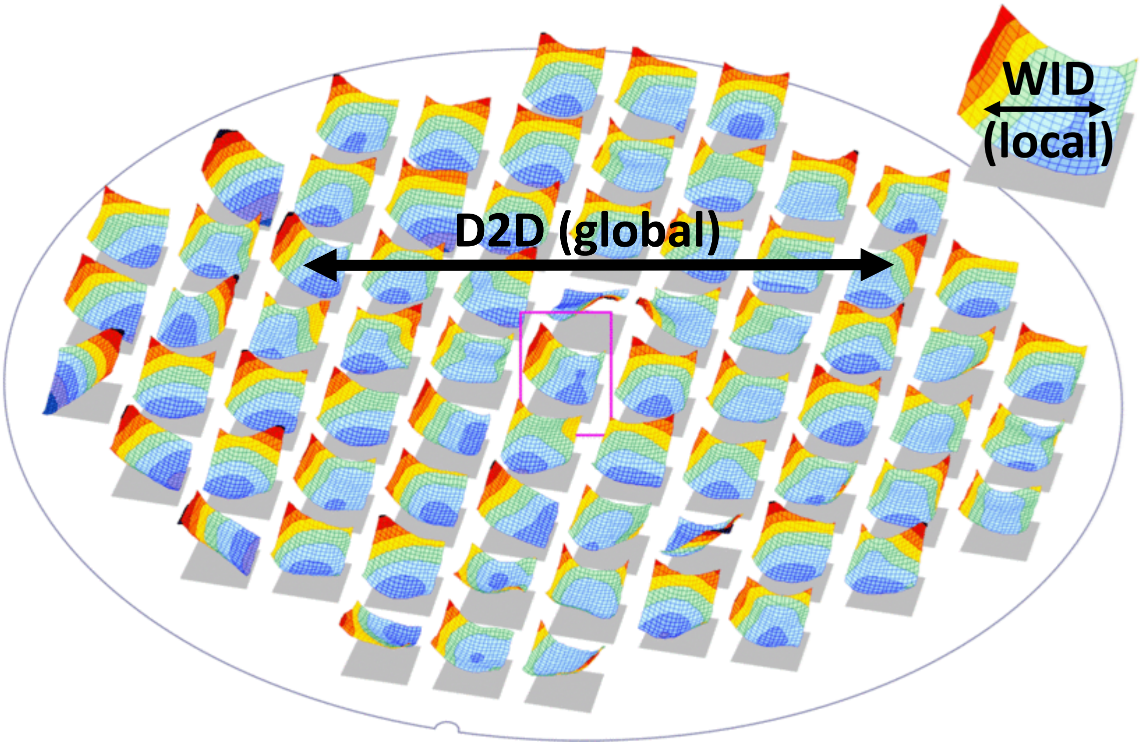
Dynamic variability



Raj Nair (Anasim Corp.),
IC floorplanning and Power Integrity, SOCcentral, Aug 2, 2010.
http://www.soccentral.com/results.asp?entryID=31901.
From SOCcentral.com, Copyright 2002 - 2014 Tech Pro Communications

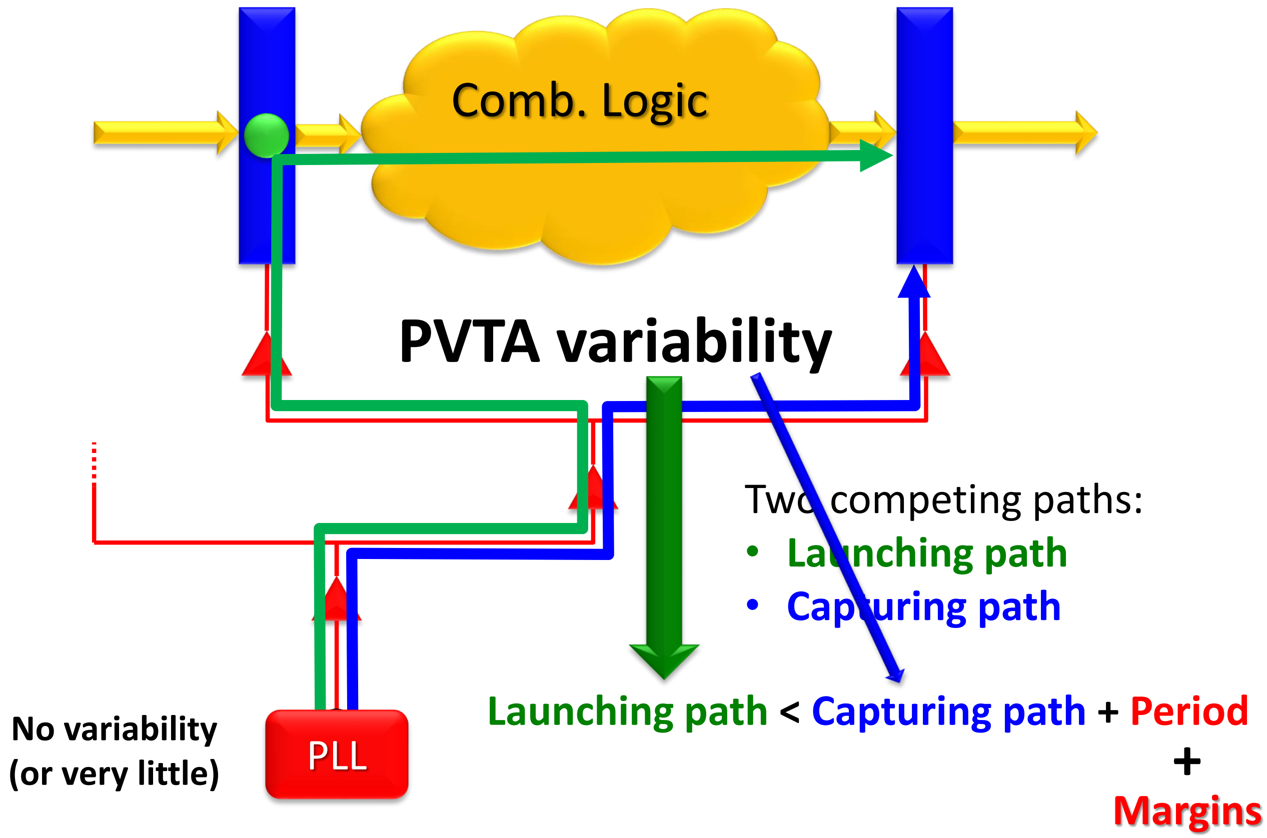
- Dynamic variability (V,T) depends on the actual activity of the circuit
- Assumptions required on the min/max values for voltage and temperature
- Ranges may depend on the application domain (automotive, HPC, handheld devices, …)

Global and local variability



Source: H. Masuda et al. (ICICC 2005)
Challenge: Variability Characterization and Modeling for 65- to 90-nm Processes

Timing: setup constraint



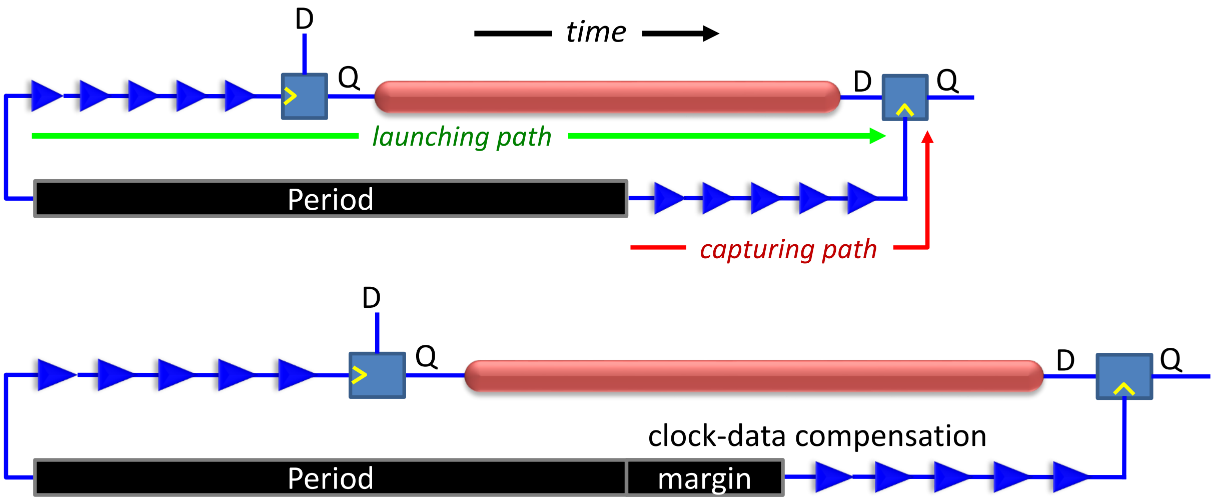
No variability
(or very little)

Two competing paths:

- **Launching path**
- **Capturing path**

Launching path < Capturing path + Period
+
Margins

Margins for Variability



Margins are required to account for the
worst-case variability (static and dynamic)

Is there room to reduce margins?

Variability and margins

	Static	Dynamic	
		Slow (ms)	Fast (ns)
Global (corners)	PV	VTA	V
Local (OCV)	PV	VTA	V

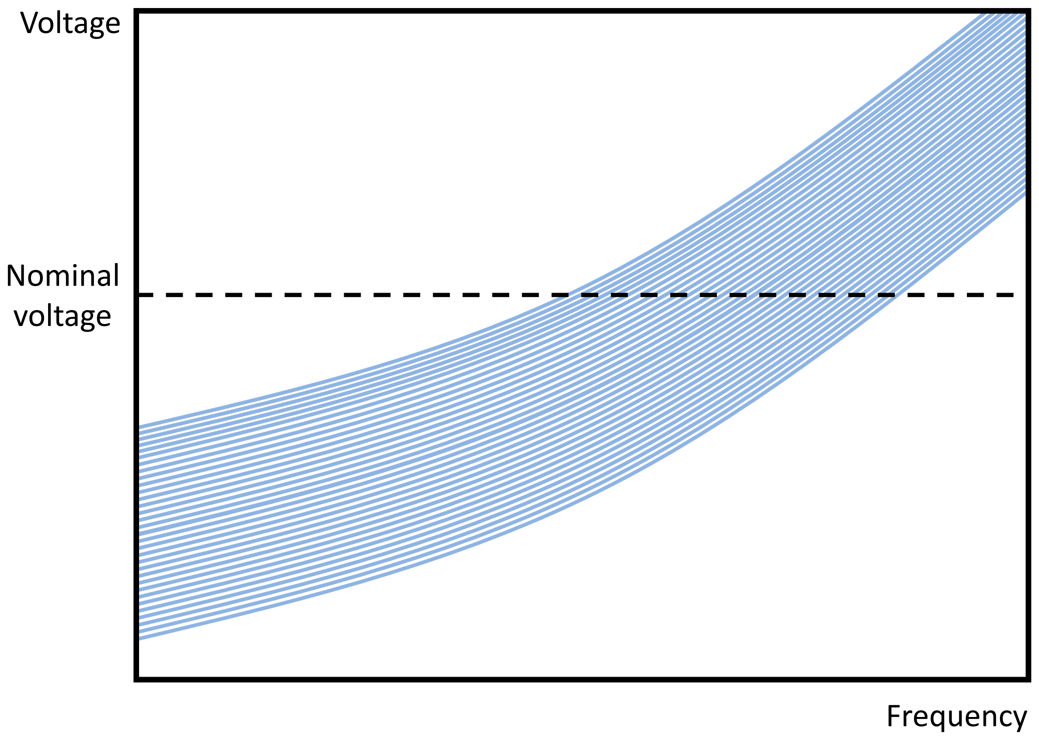
(P:Process; V: Voltage; T: Temperature; A: Aging)

Voltage variability:

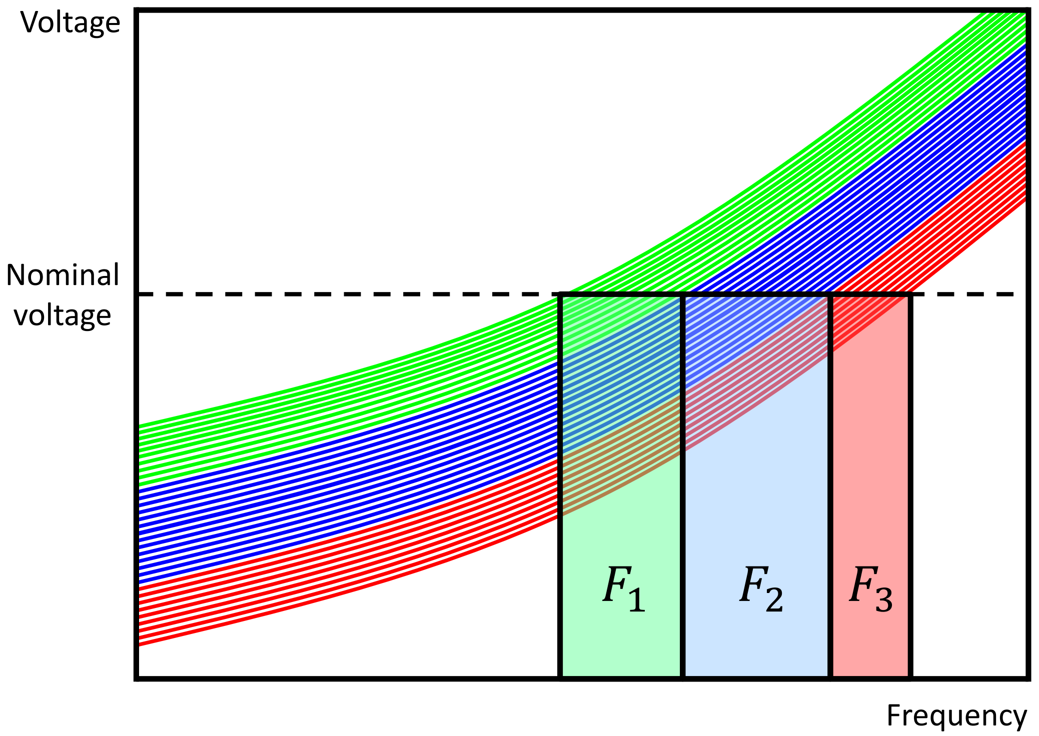
- Static IR drop
- Dynamic IR drop
- Inductive noise (L·*di/dt*)

Reducing Margins for Static and Slow Variability

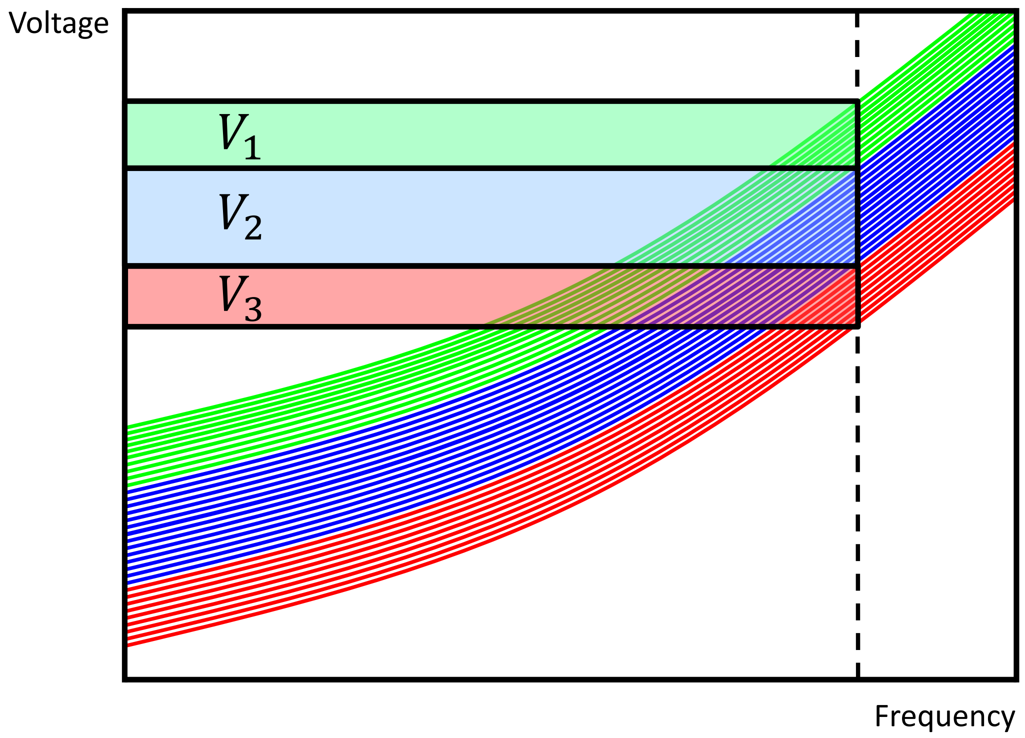
V-F operation points



Speed binning

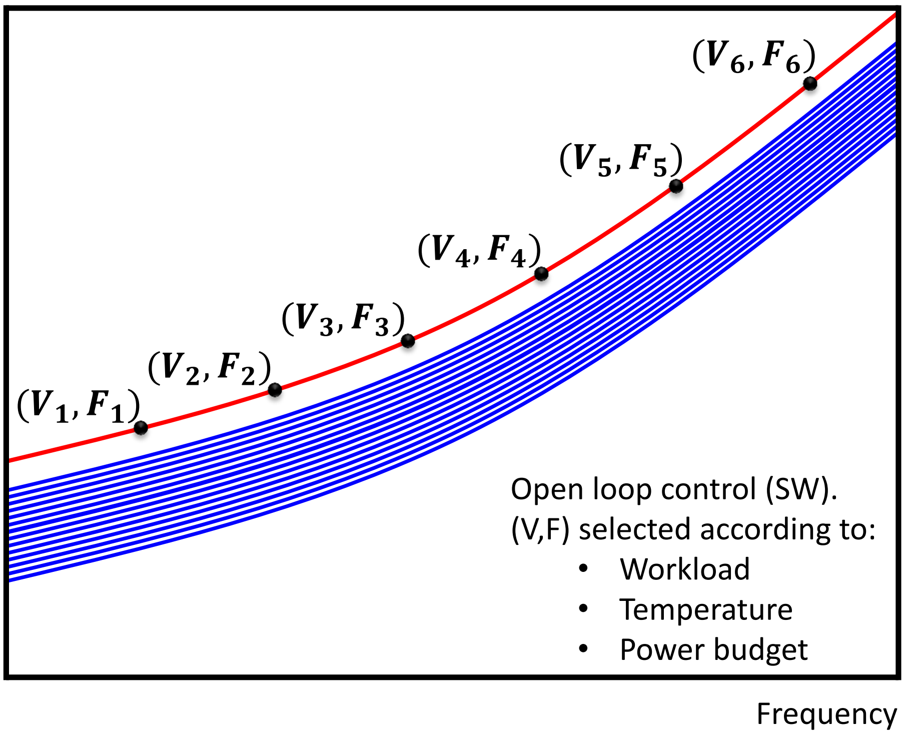


Voltage binning



Dynamic Voltage Frequency Scaling (DVFS)

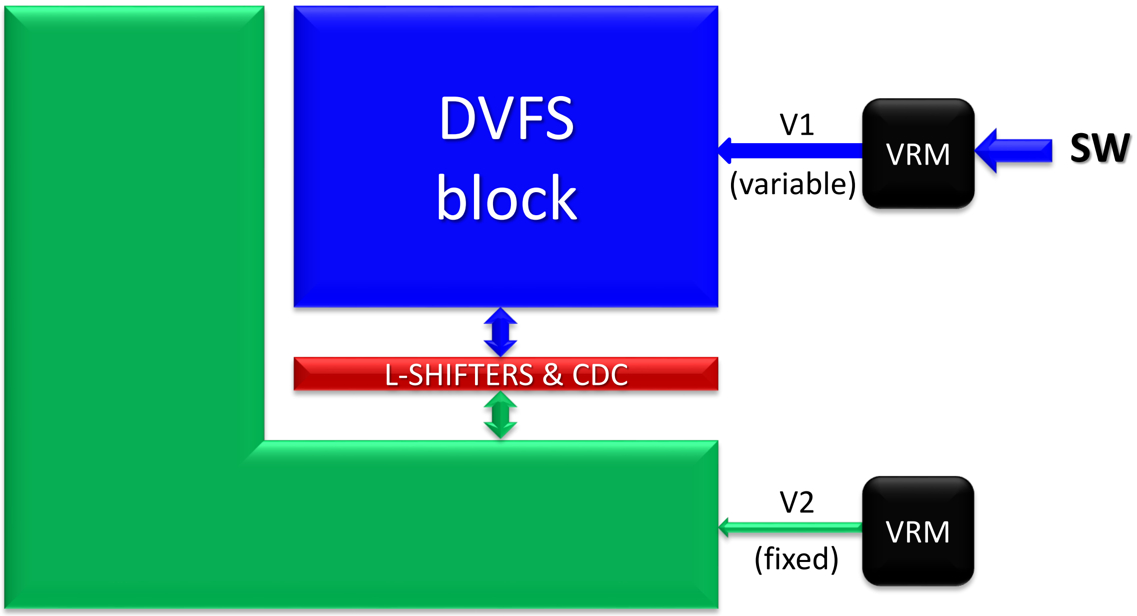
DVFS Table	
V_1	F_1
V_2	F_2
V_3	F_3
V_4	F_4
V_5	F_5
V_6	F_5



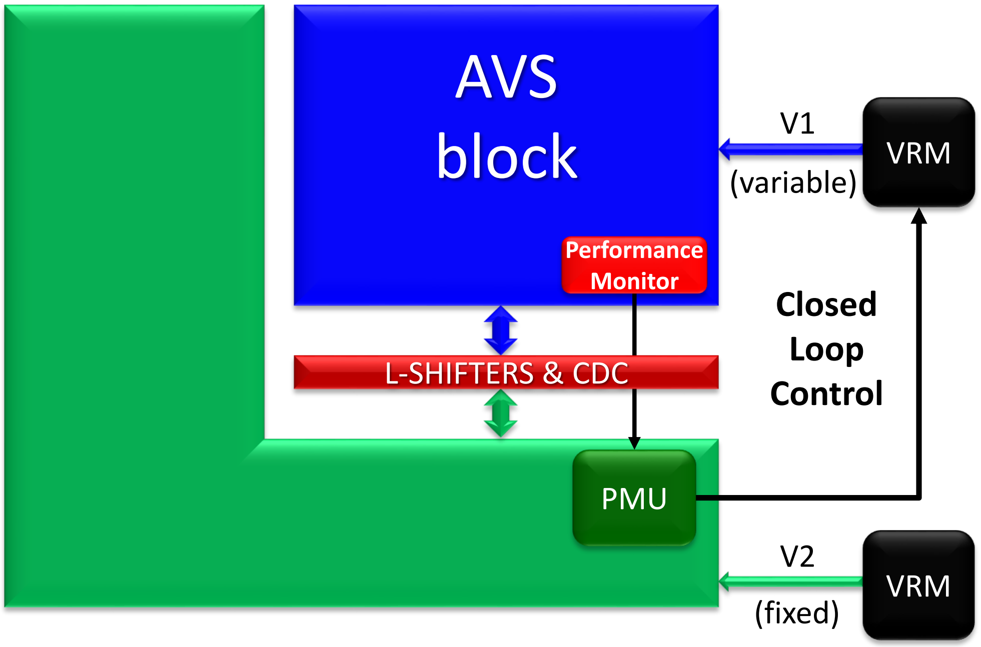
Open loop control (SW).
(V,F) selected according to:

- Workload
- Temperature
- Power budget

DVFS scheme



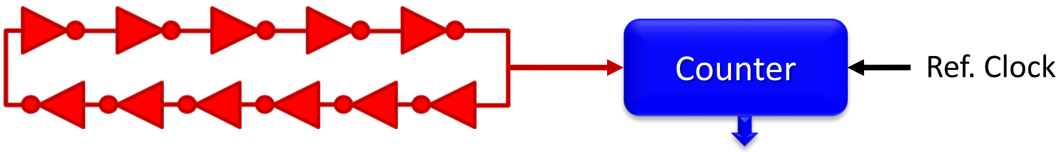
AVS scheme



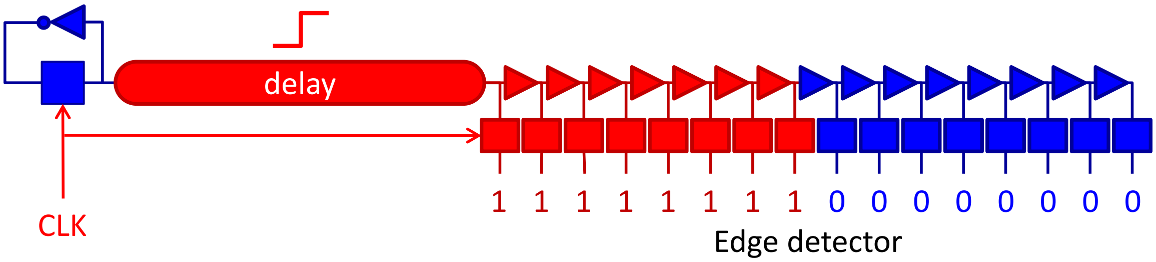
- Every die works at its best (V,F) point (plus some guardband margins)
- AVS can compensate slow variability (temperature, aging)

Performance monitors

- Estimate the voltage/frequency relationship of the actual silicon
- Examples:



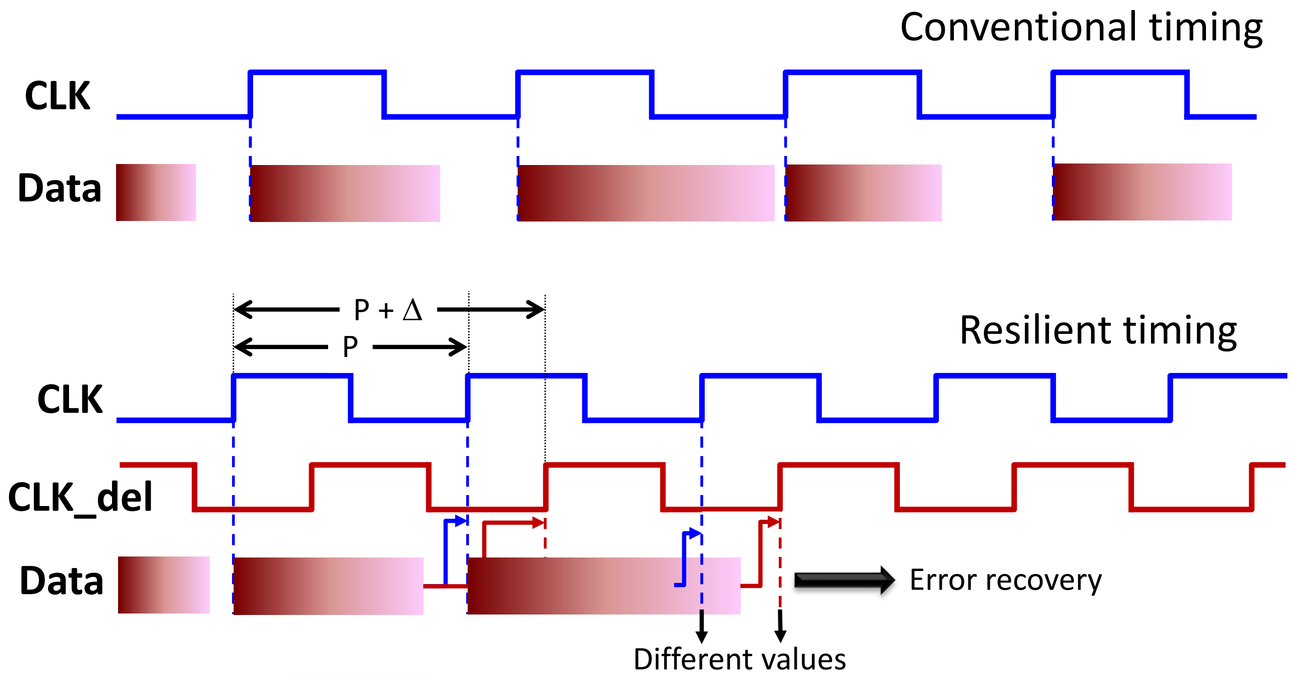
Ring oscillator

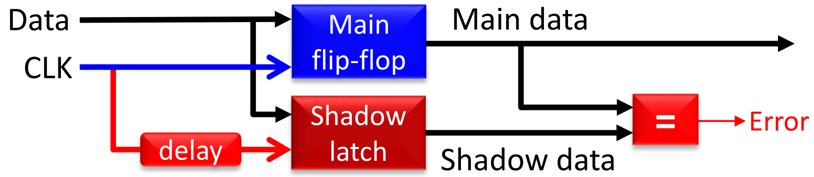


Time-to-digital converter

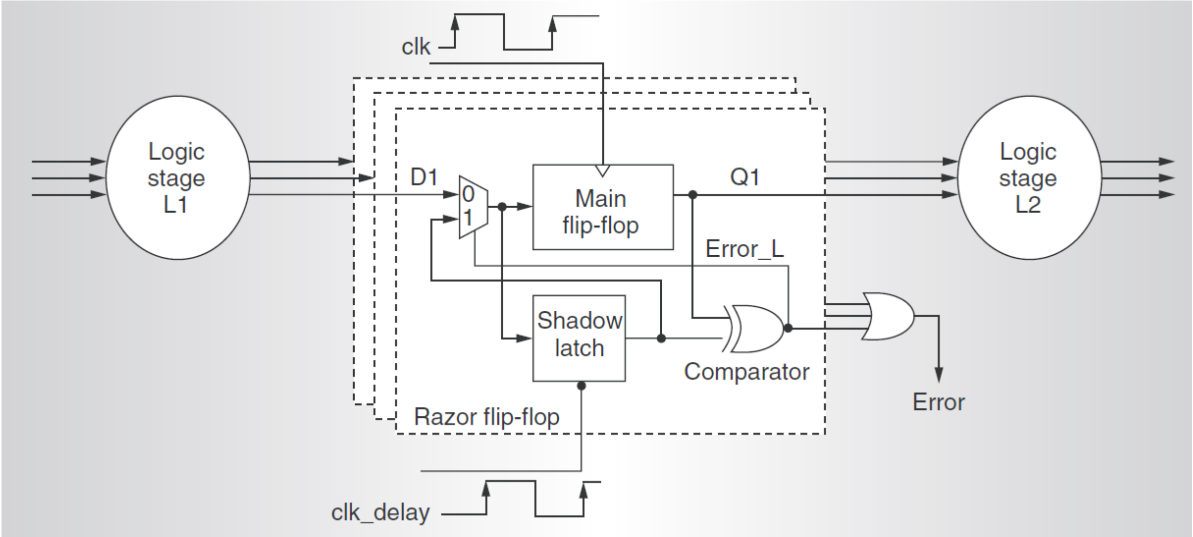
Reducing Margins for Fast Variability

Resilient circuits



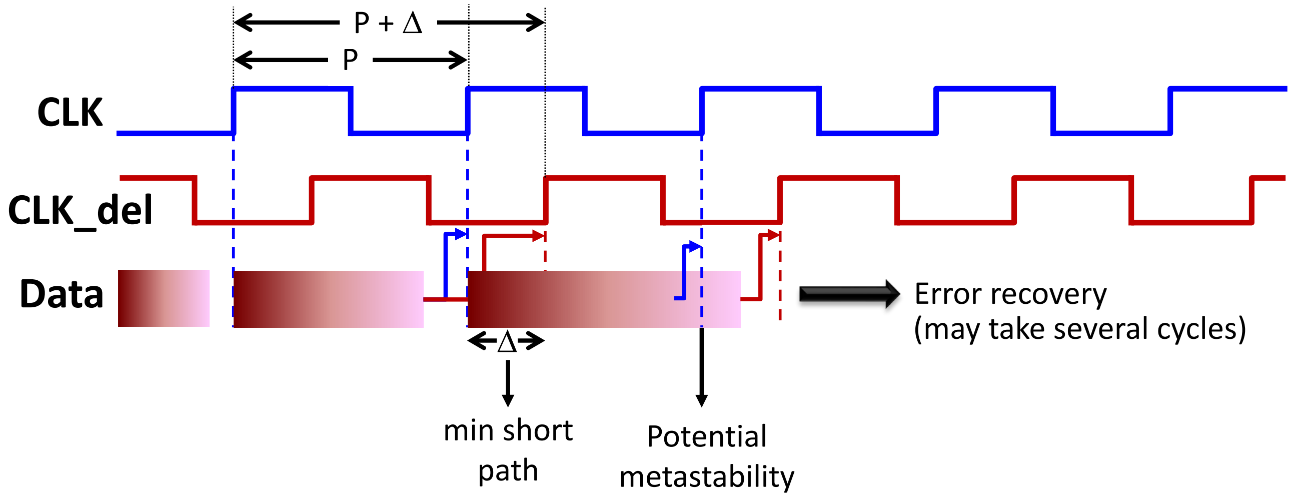


Resilient circuits

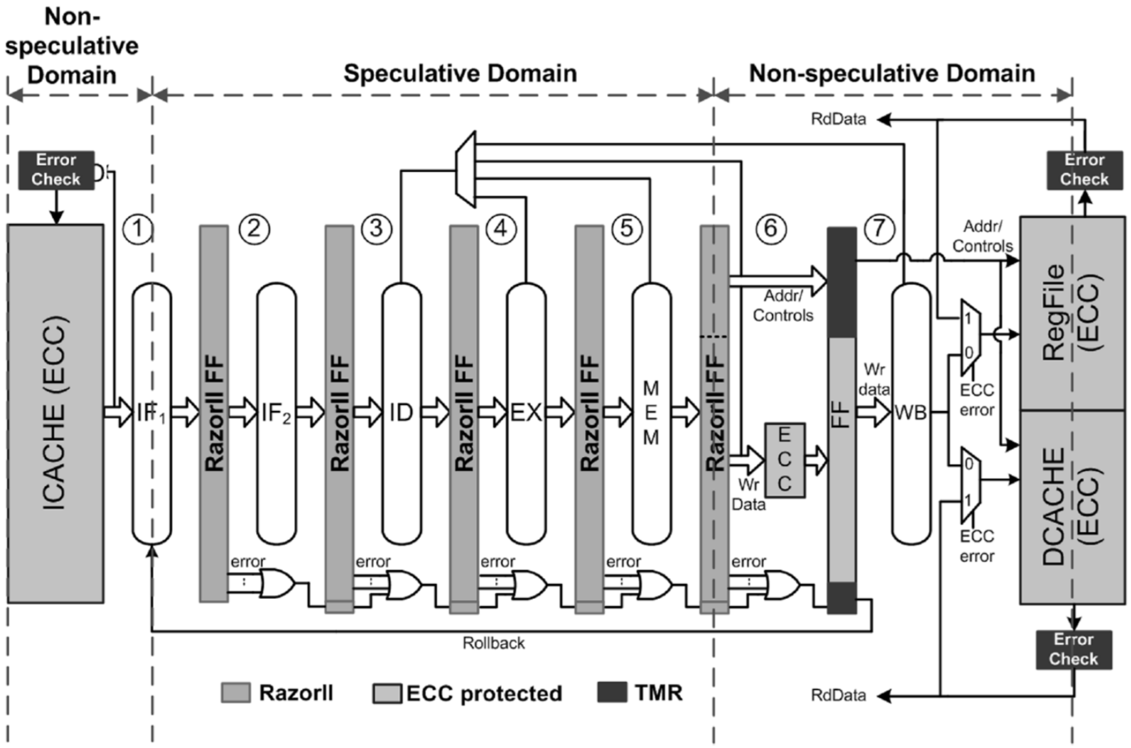


Source: D. Ernst et al.,
Razor: circuit-level corrrection of timing errors for low-power operation,
IEEE Micro, Nov-Dec 2004.

Resilient circuits: technical issues



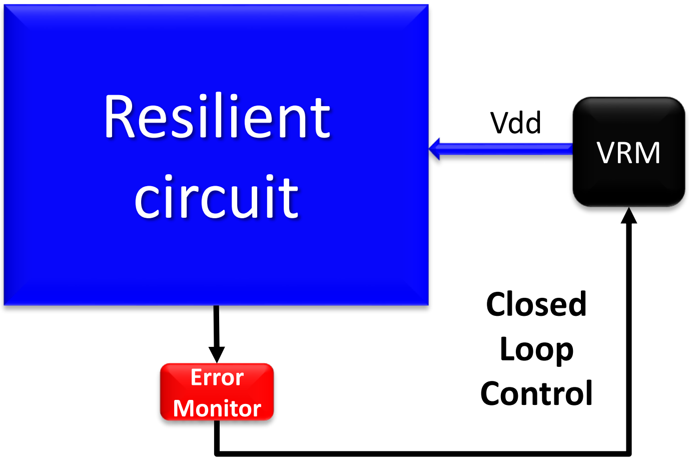
- Only critical flip-flops require shadow latches
- Short paths must have a min delay
- Metastability may take several cycles to resolve
- Sophisticated error-recovery procedures are required



Source: S. Das et al.,
RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance.
IEEE JSSC 44(1), Jan 2009.

Resilient circuits

$$P_{eff} = \hat{P}(1 + \underbrace{n}_{\text{num cycles for error recovery}}\underbrace{\mathcal{E}}_{\text{probability of error}})$$



Closed Loop Control:

- Raise voltage if too many errors
- Lower voltage if too few errors

Error recovery mechanism:

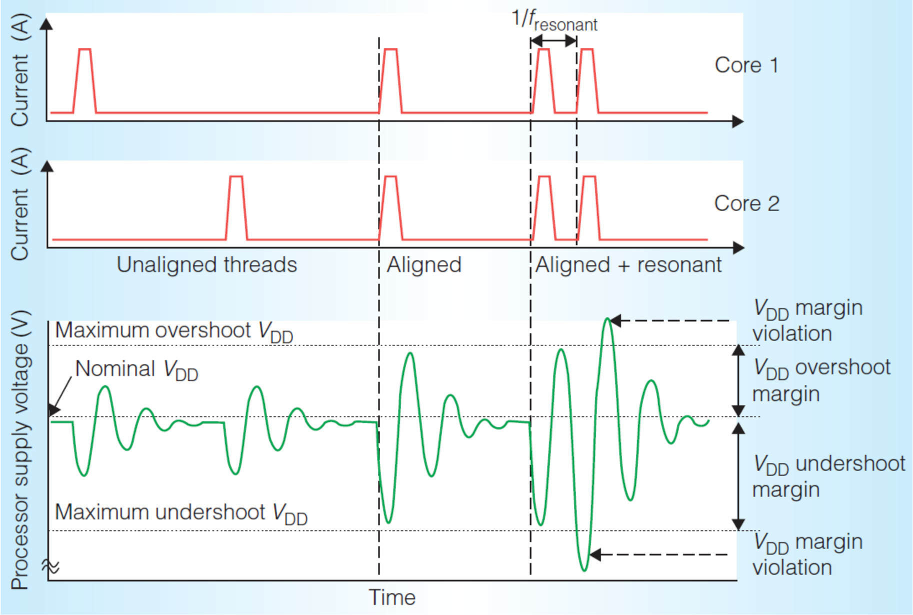
- Reuse existing checkpoint schemes in advanced processors

Schemes with resilient circuits

- ARM + University of Michigan
  - Different variations: Razor I, Razor II, Bubble Razor
- Intel:
  - Transition Detector with Time Borrowing (TDTB)
  - Double Sampling with time Borrowing (DSTB)
- Using pausable clocks and metastability detectors
  - SafeRazor (Torino, Technion and UPC)
  - Blade (USC)

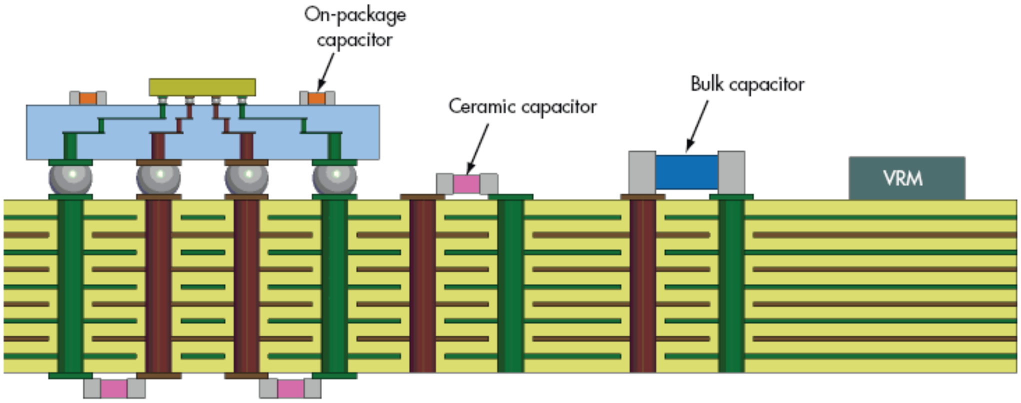
Voltage droops

Voltage droops are the main source of fast variability



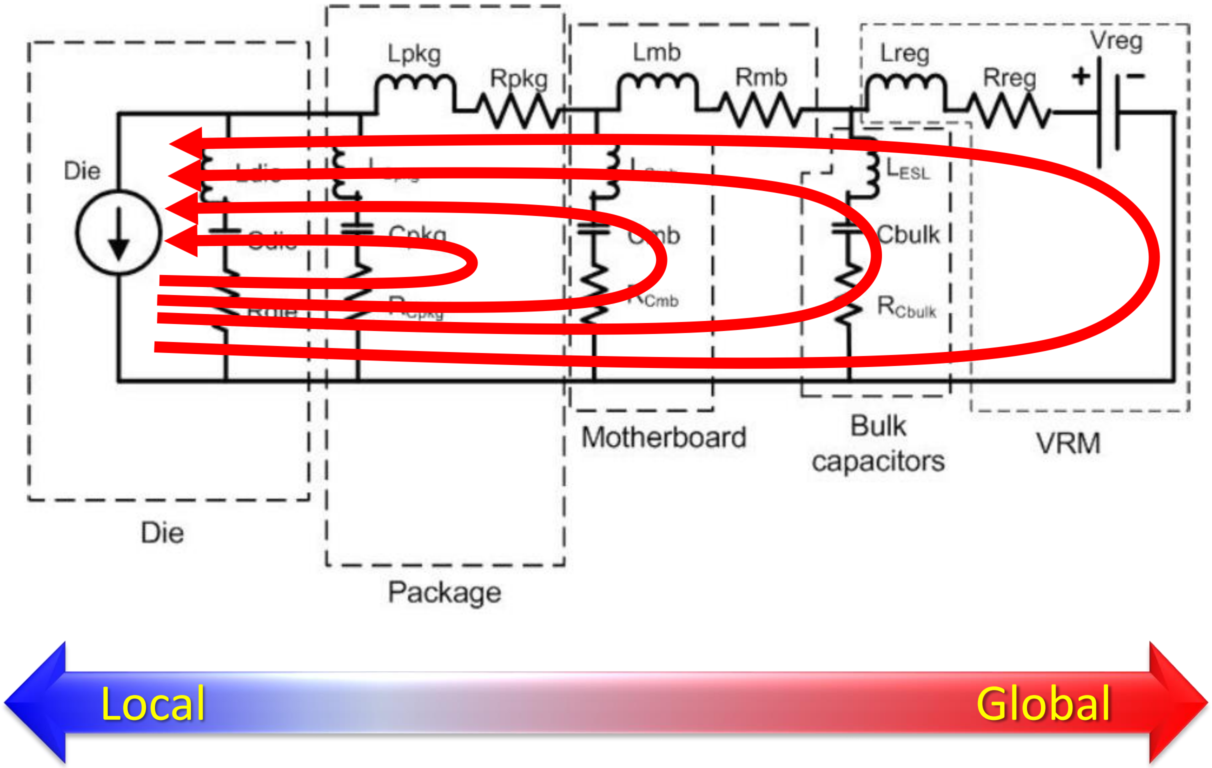
Source: Y. Kim et al.,
Automating Stressmark Generation for Testing Processor Voltage Fluctuations,
IEEE Micro, July-August 2013.

Power Delivery Network

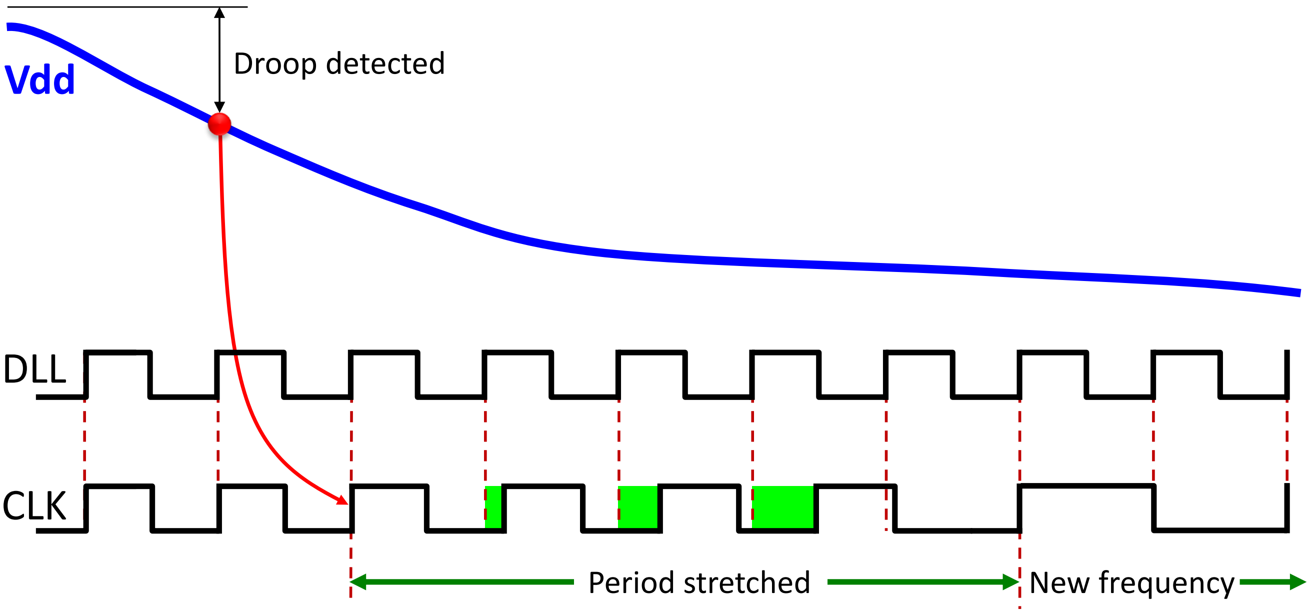


Source: Jitesh Shah
Floorplanning A Power Delivery Network With Spice Electronic Design
July 24, 2008
http://electronicdesign.com/energy/floorplanning-power-delivery-network-spice

PDN model



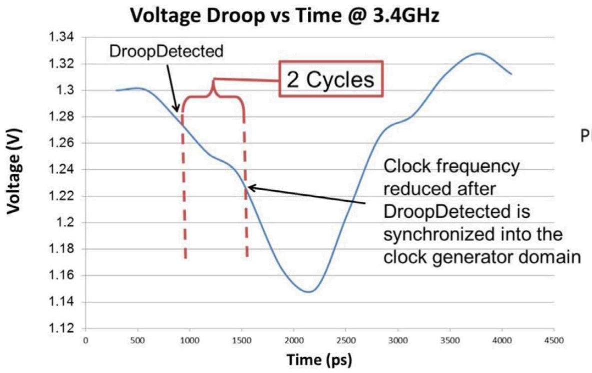
Adaptive Clock

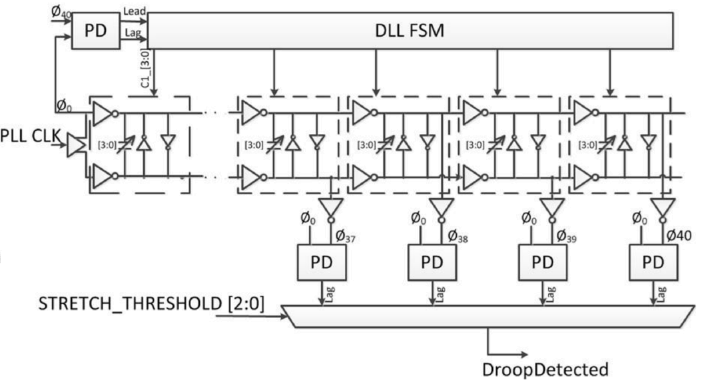


Mechanisms are needed for:

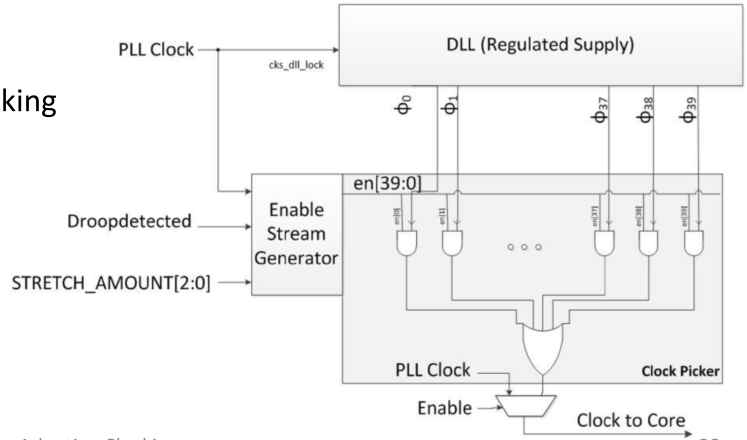
- Droop detection
- Clock period stretching
- Frequency reduction

AMD Steamroller

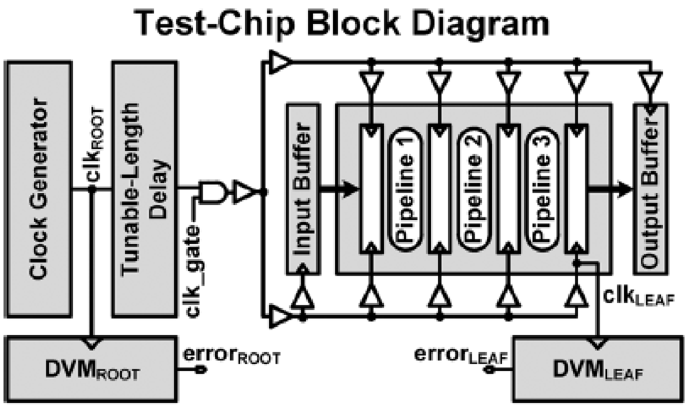


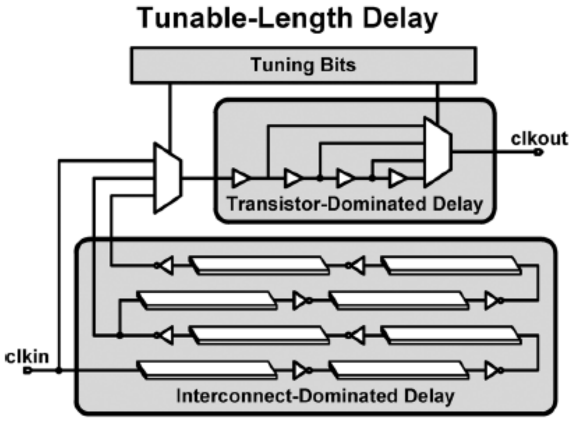


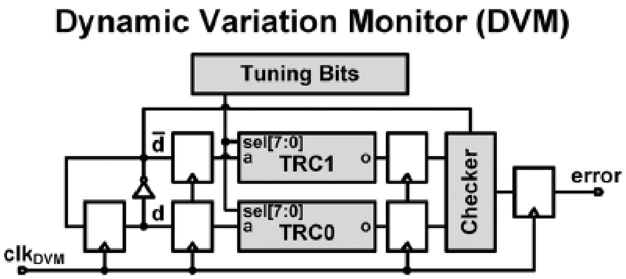
Source: K. Wilcox et al.
Steamroller Module and Adaptive Clocking System in 28nm CMOS
IEEE JSSC 50(1), Jan 2015.



Intel Adaptive Clock

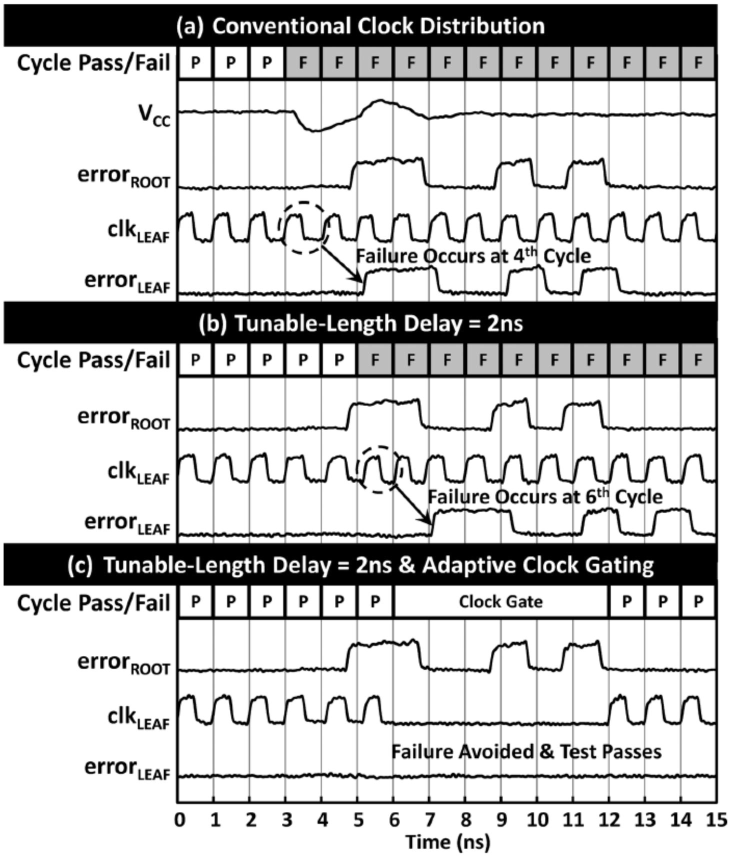




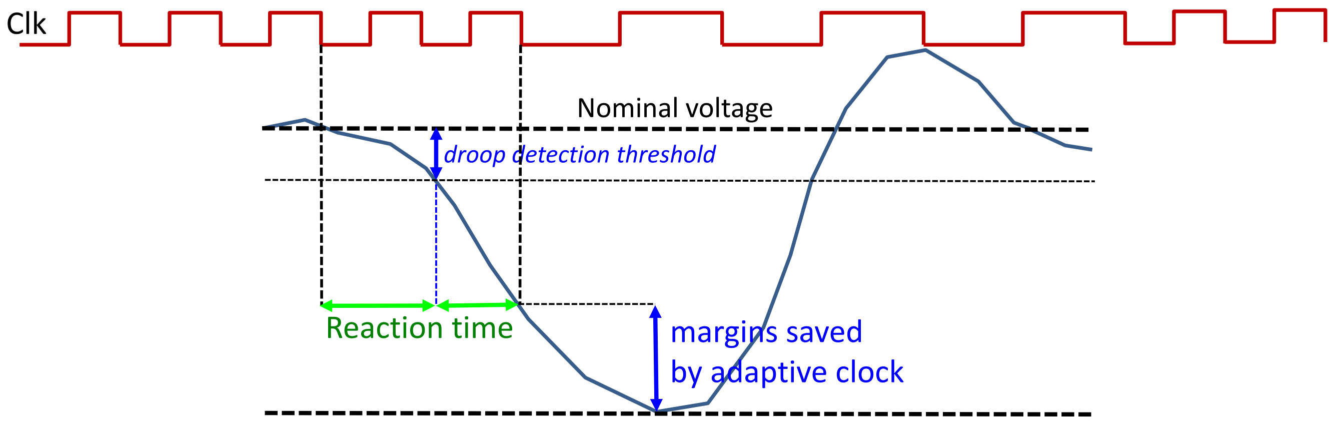


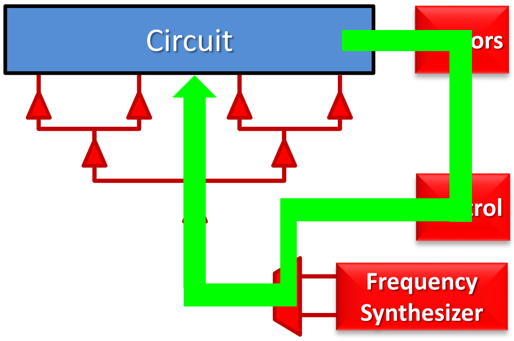
Source: K.A. Bowman et al.
A 22 nm All-Digital Dynamically Adaptive Clock Distribution for Supply Voltage Droop Tolerance,
IEEE JSSC 48(4), April 2013.

Intel Adaptive Clock



Margins for Adaptive Clocks





Other approaches:

- N. Kurd, P. Mosalikanti, M. Neidengard, J. Douglas, and R. Kumar, Next generation Intel core micro-architecture (Nehalem) clocking, IEEE JSSC 44(4), 2009.
- K. Chae and S. Mukhopadhyay, All-digital adaptive clocking to tolerate transient supply noise in a low-voltage operation, IEEE TCAS II 59(12), 2012.
- C. Lefurgy, A. Drake, M. Floyd, M. Allen-Ware, B. Brock, J. Tierno, J. Carter, and R. Berry, Active guardband management in Power7+ to save energy and maintain reliability, IEEE Micro 33(4), 2013.

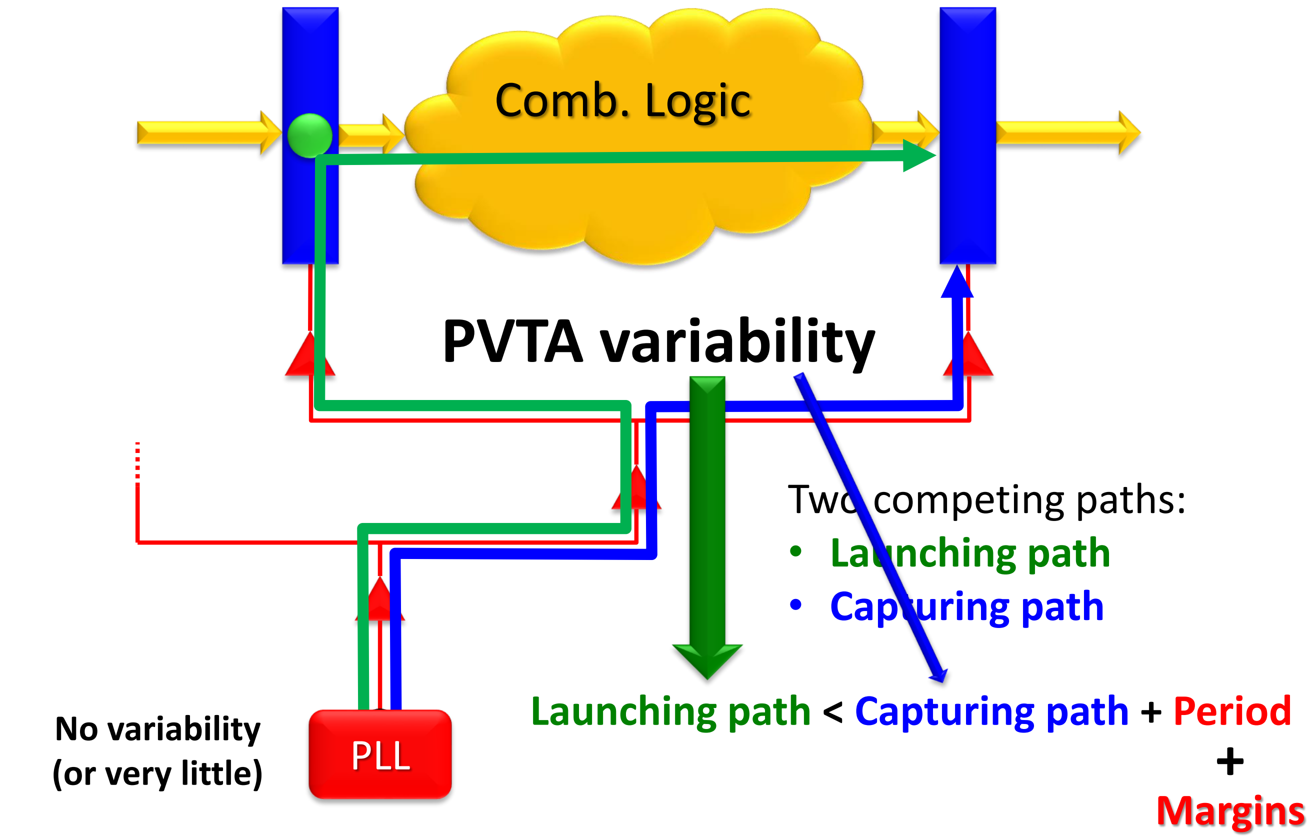
Reactive Clocks

Mechanisms to reduce margins

	Static	Dynamic	
		Slow (ms)	Fast (ns)
Global (corners)	PV	VTA	V
Local (OCV)	PV	VTA	V

Mechanism	Reduces margins for …
Binning, DVFS (open loop)	Static variability
AVS (close loop)	+ Slow global variability
Adaptive Clocks	+ Fast global variability (moderate)
Reactive Clocks	**+ Fast global variability (aggressive)**
Resilient circuits	Any variability (aggressive)

Timing: setup constraint

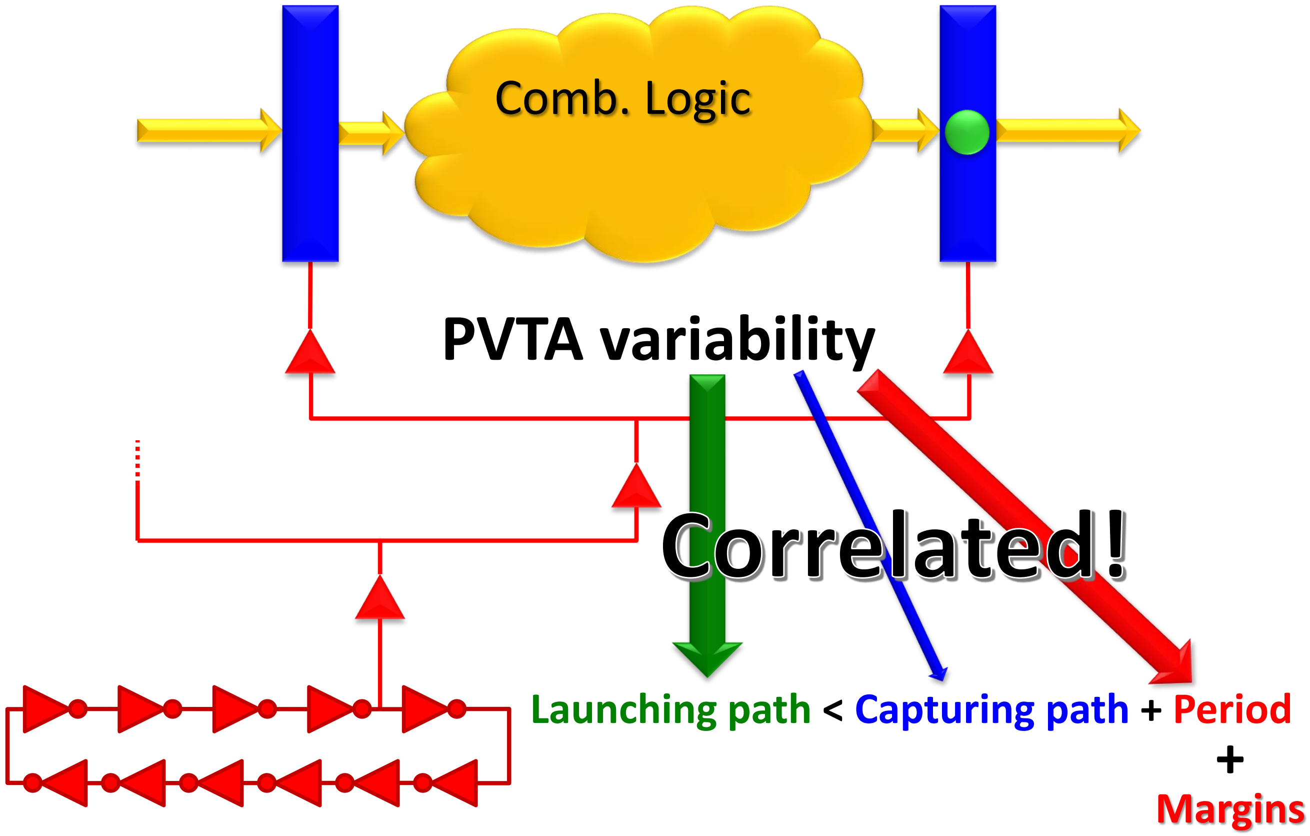


Two competing paths:

- Launching path
- Capturing path

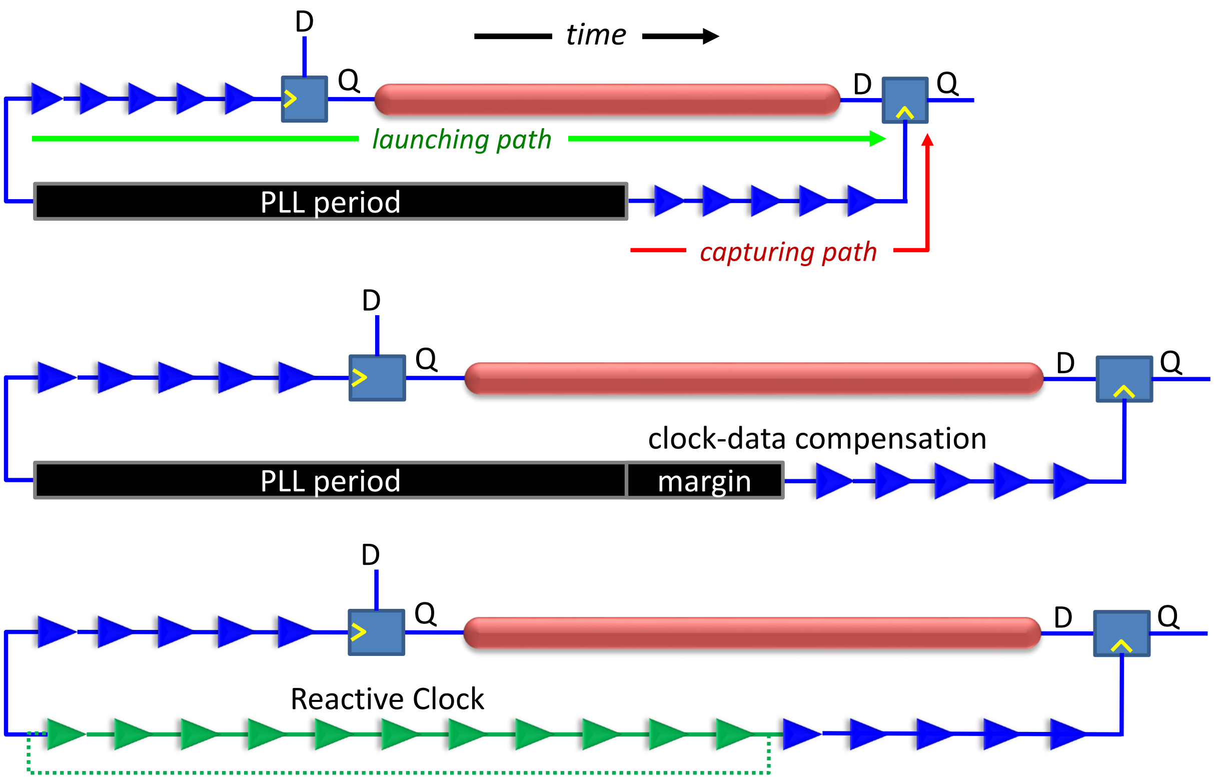
Launching path < Capturing path + Period
+
Margins

Reactive Clock

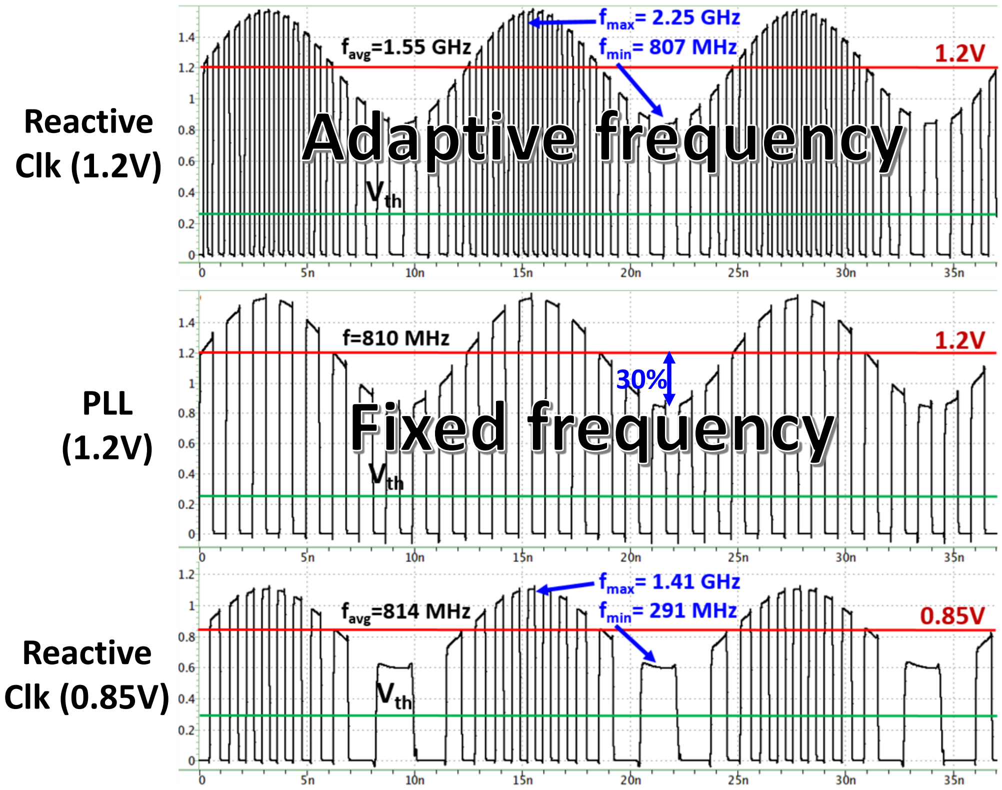


Launching path < Capturing path + Period
+
Margins

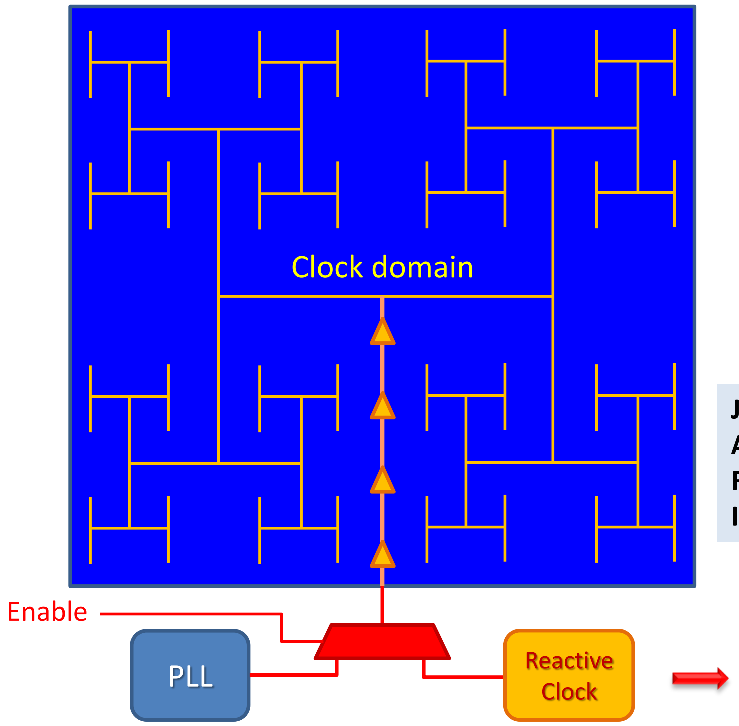
Variability: PLL vs. Reactive Clock



PLL vs. Reactive Clock



Reactive Clock



Gain: 40% less Energy or 1.6x speed-up

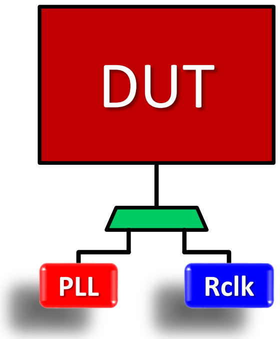
Pain: Negligible

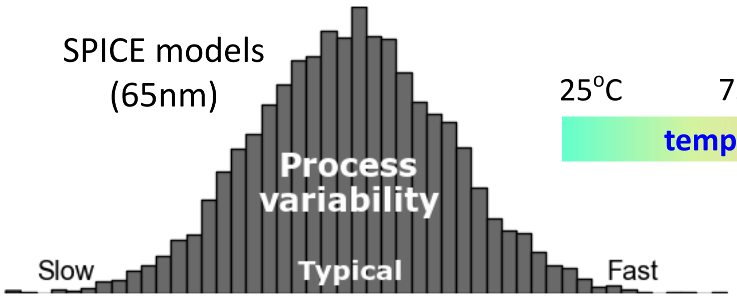
Risk: Zero

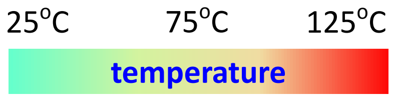
J. Cortadella, L. Lavagno, P. López, M. Lupon, A. Moreno, A. Roca, and S. S. Sapatnekar. Reactive clocks with variability-tracking jitter. ICCD 2015.

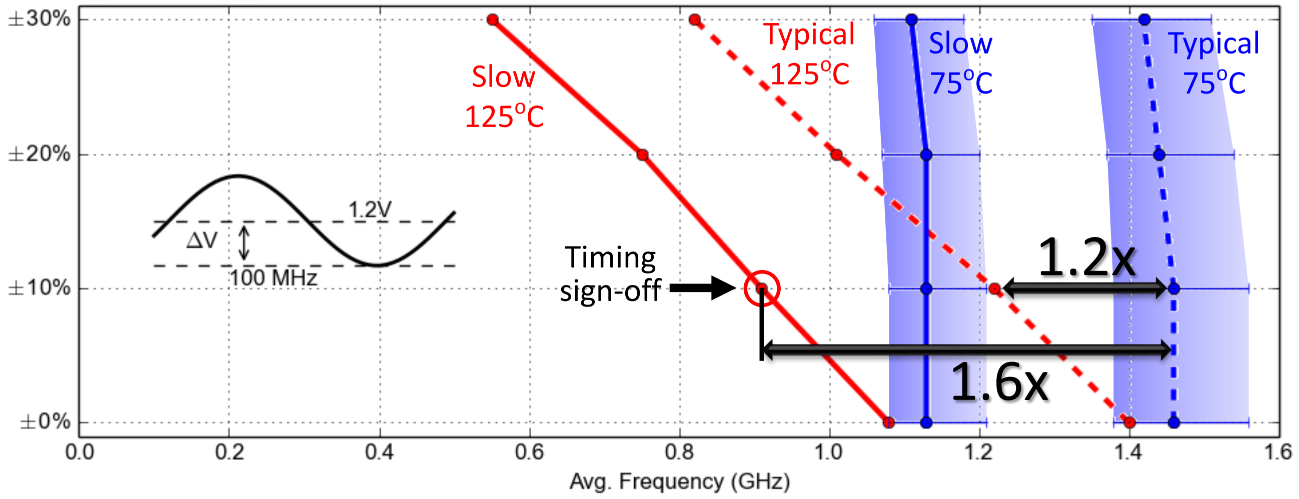
- Original circuit not modified
- Negligible area
- Post-tapeout calibration (by SW)
- Conventional timing (PrimeTime)

PLL vs. Reactive Clocks

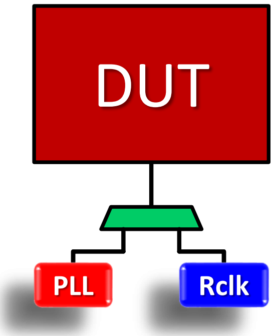


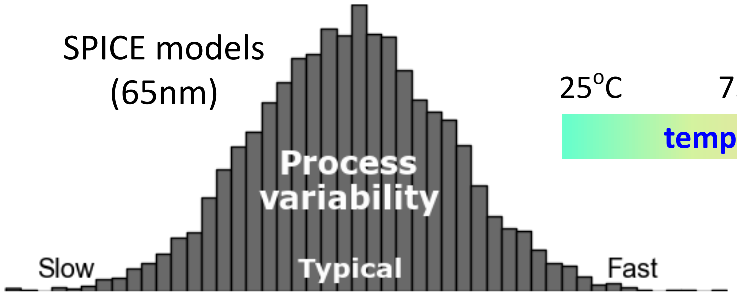


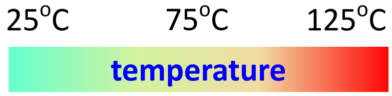


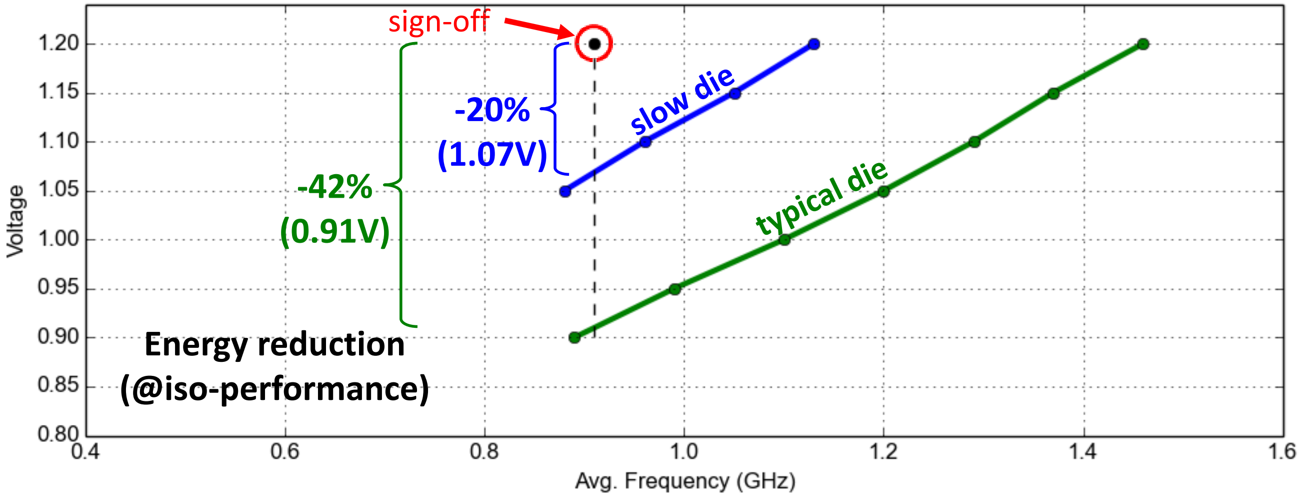


PLL vs. Reactive Clocks

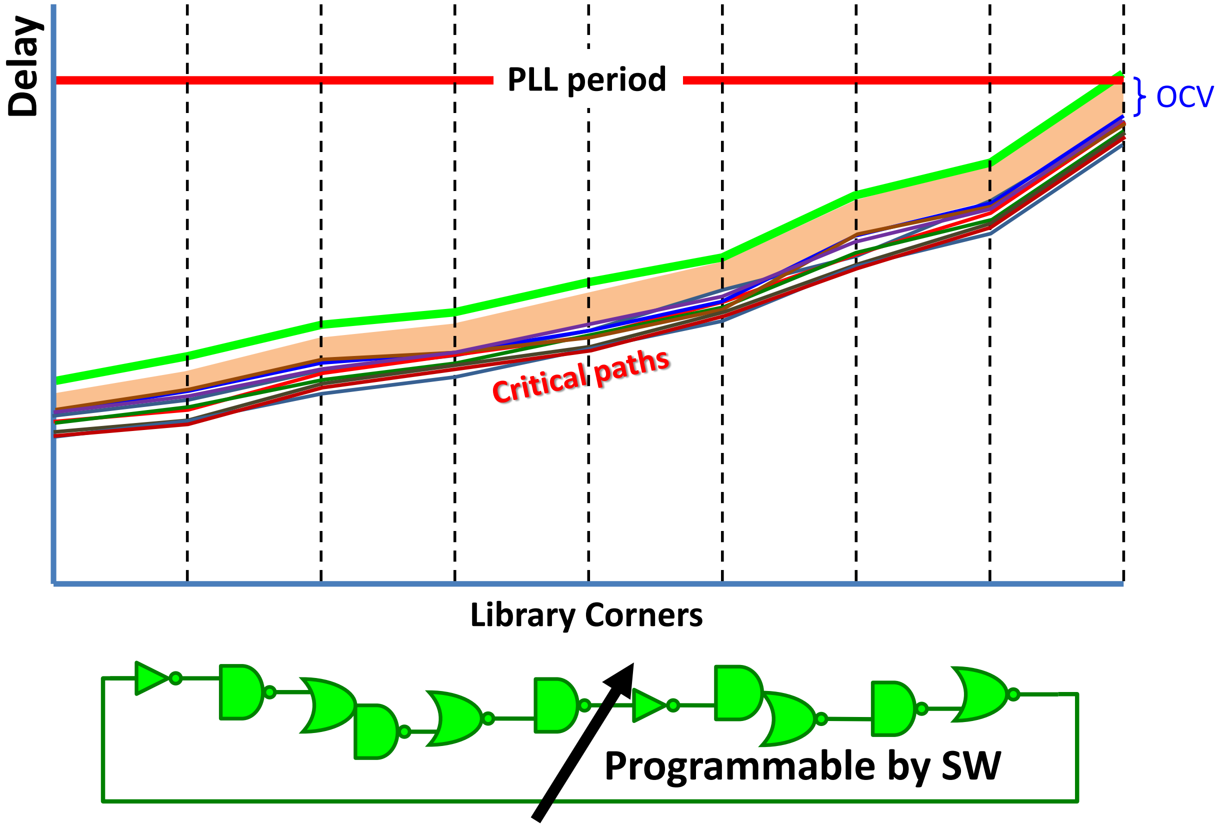




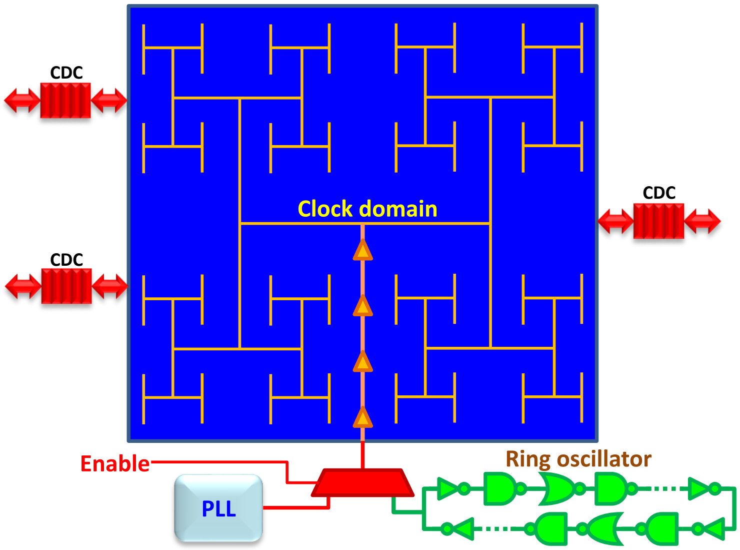




Synthesis of the Ring oscillator

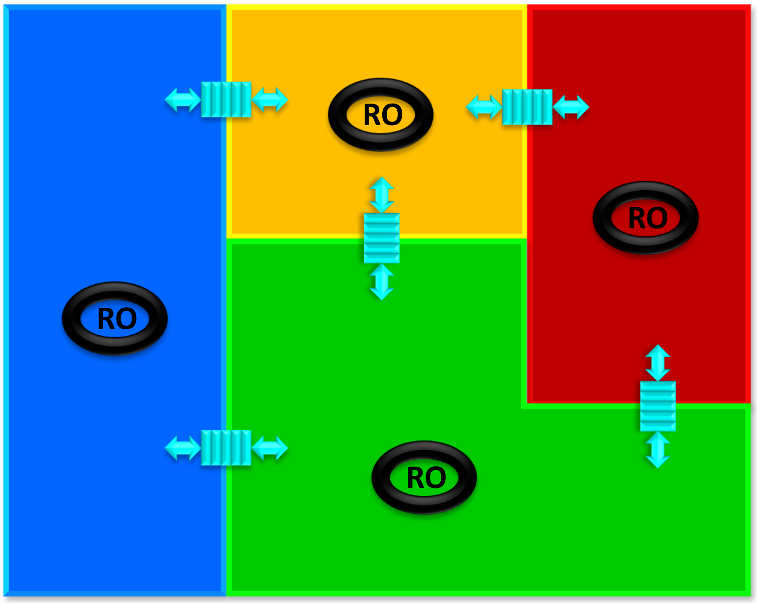


Reactive Clock Domain



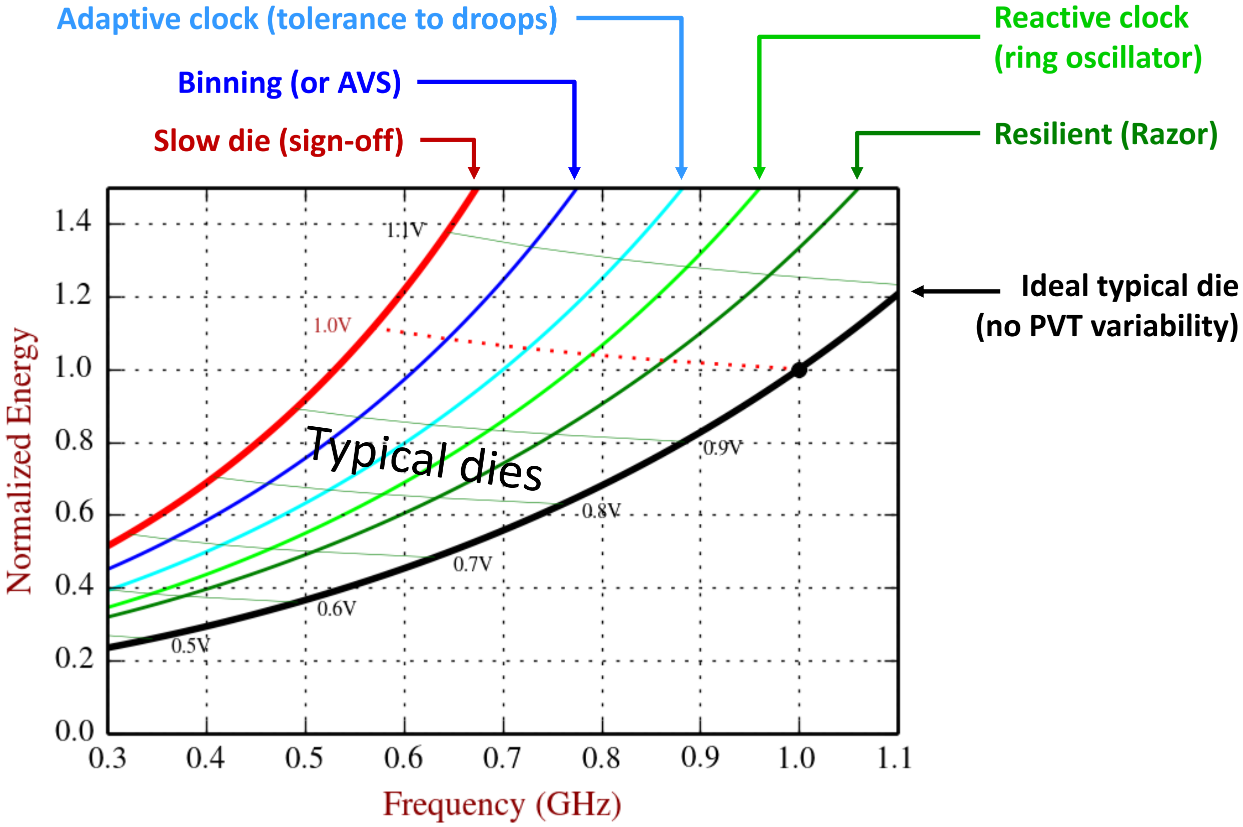
CDC structures required for communication outside the clock domain

GALS with Reactive Clocks



- A different Reactive Clock for each domain
- Clock Domain Crossing structures between domains

Summary



Adaptive clock (tolerance to droops)

Binning (or AVS)

Slow die (sign-off)

Reactive clock (ring oscillator)

Resilient (Razor)

Ideal typical die (no PVT variability)

Typical dies

Conclusions

- Moore’s law is (economically) over. It is time to exploit what is left in established nodes.

- What is left? Margins for variability.

- Dilemma: how to spend your money?
  - – introducing techniques to reduce margins, or
  - – moving to a new (expensive) technology node