Multi-Synchronous Relative Timing Providing Order-of-Magnitude Energy Reduction

Kenneth S. Stevens

University of Utah

Granite Mountain Technologies

Multi-Synchronous Wins Big

1. **Efficiency** in **power** and **performance** is new game in town
2. We need to think about problems differently
3. New timing model is one excellent path to progress
4. Multi-synchronous design gives ave. 10$\times$ $e\tau^2$ improvement
   - Pentium: $e\tau^2 = 17.5\times$
   - FFT: $e\tau^2 = 16.9\times$

Design	Energy	Area	Freq.	Latency	Aggregate
Pentium F.E.	2.05	*0.85*	2.92	2.38	12.11$\times$
64-pt FFT	3.95	2.83	2.07	3.37	77.98$\times$

Outline

1. New Generation of commercial multi-synchronous architectures
   - Best *multi*-synchronous is *a*-synchronous
   - Mixed clocked and asynchronous circuits
   - Enabler: Relative Timing
   - Foundation: Formal Methods
2. Relative Timed Design Flow Overview
3. Comparison to Clocked Technology
   - 10$\times$ improvements in $e\tau^2$
   - Demonstrated across broad set of design classes

Commercializing Multi-Synchronous ICs

1. **Timing is a key method of gaining $e\tau^2$ improvement**
   - Multi-synchronous allows best optimization of design
   - Exploit affect of time in our circuits and architectures
   - *Relative Timing* supports all time methods & models
2. **Utilize best capabilities in industry**
   - No change plus minimal enhancements to EDA / CAD / flows
   - Cell libraries unmodified
3. **Leverage designer's creativity**
   - Provide familiar design environment
   - Enhance modularity and design visibility
   - Do not restrict circuits, architectures, flows

Disruptive Technology

"In some sense, Apple's most fundamental problem, perhaps, is that a superior technology is still an inferior solution if it lacks synergy with the mainstream."

Michael Slater, MPR Vol 11 No. 17, Dec. 29 1997, p27

None of the technical issues matter if the disruptive technology doesn't integrate with the mainstream.

New Direction for Async Design:

- Utilize clocked CAD to synthesize, place, and route unclocked designs.

. . . but how?

Disruptive Technology

"In some sense, Apple's most fundamental problem, perhaps, is that a superior technology is still an inferior solution if it lacks synergy with the mainstream."

Michael Slater, MPR Vol 11 No. 17, Dec. 29 1997, p27

None of the technical issues matter if the disruptive technology doesn't integrate with the mainstream.

New Direction for Async Design:

- Utilize clocked CAD to synthesize, place, and route unclocked designs.

. . . but how? **relative timing!**

Timing and Sequencing

Traditional representation of timing:

- Metric values
  - On an IC we measure it to picoseconds
  - In track and ski racing, we measure it to milliseconds

But what do we really care about?

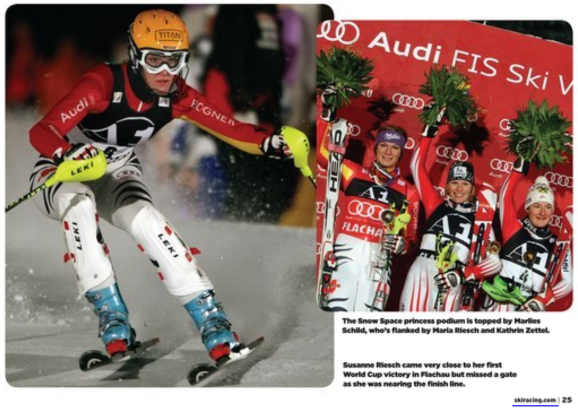
- *it isn't the number on the stop watch. . .*

Timing and Sequencing

Traditional representation of timing:

- Metric values
  - On an IC we measure it to picoseconds
  - In track and ski racing, we measure it to milliseconds

But what do we really care about?



The Snow Space princess podium is topped by Marlies Schild, who's flanked by Maria Riesch and Kathrin Zettel.

Susanne Riesch came very close to her first World Cup victory in Flachau but missed a gate as she was nearing the finish line.

skiracing.com | 25

- *it isn't the number on the stop watch. . .*

We care about who wins!!

The key: **Timing results in sequencing**

Relative Timing formally represents the signal sequencing *produced by circuit timing*

New *Formal* Abstract Model: Relative Timing

- **Timing** is both the technology differentiator and barrier
- **Relative Timing** is the generalized solution
- The **key property** of time is the **sequencing** it imposes

Sequence gives winner, performance, etc.

- true in semiconductors as well as sports
- absolute stopwatch value is auxiliary

Novel relativistic formal logic

representation of time (relative timing):

$$\text{pod} \mapsto \text{poc}_1 \prec \text{poc}_2$$

Sequencing relative to common reference

- can now *evaluate* sequencing
- can now *control* sequencing

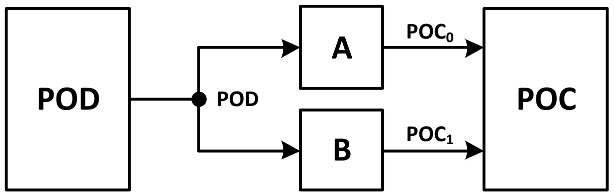
The Snow Space princess podium is topped by Marlies Schild, who's flanked by Maria Riesch and Kathrin Zettel.

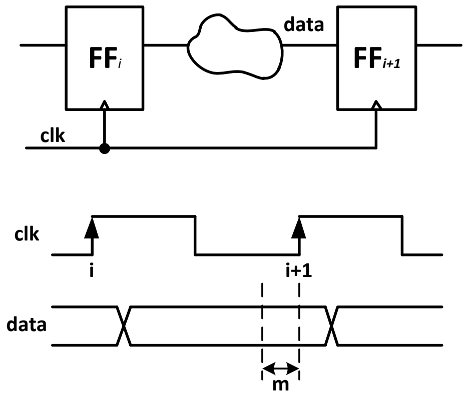
Susanne Riesch came very close to her first World Cup victory in Flachau but missed a gate as she was nearing the finish line.

skiracing.com | 25

Relative Timing

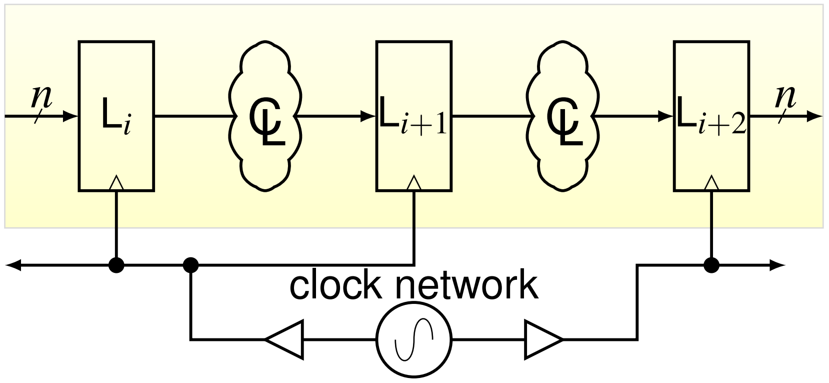
1. Relative Timing
   - Sequences signals at poc (*point of convergence*)
   - Requires a common timing reference: pod (*point of divergence*)
2. Formal representation: pod $\mapsto$ poc$_1 \prec$ poc$_2 +$ margin
3. RT models timing in ALL systems
   - Clocked: pod $=$ clock poc $=$ flops
   - Async: pod $=$ request poc $=$ latches
4. RT enables direct commercial CAD support of general timing requirements
   - formal RT constraints mapped to sdc constraints

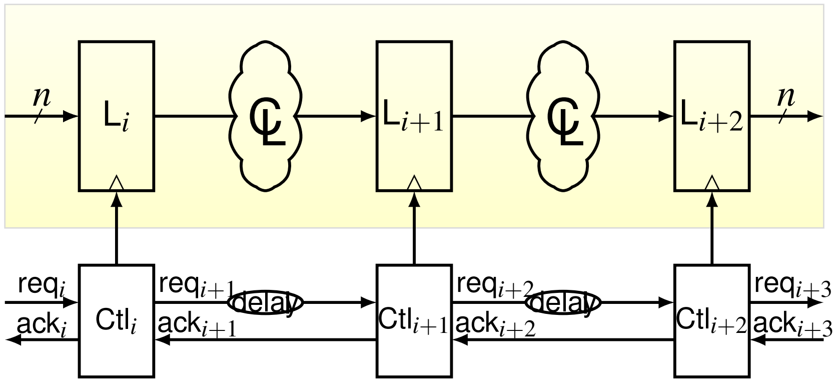




Relative Timed Design: Bundled Data

Bundled data design is much like clocked.





Frequency based (clocked) design. Clock frequency and datapath delay of first pipeline stage is constrained by
$L_i/clk{\uparrow}_i \mapsto L_{i+1}/d{+}s \prec L_{i+1}/clk{\uparrow}_{i+1}$

Timed (bundled data) handshake design. Delay element sized by RT constraint:
$req_i{\uparrow} \mapsto L_{i+1}/d{+}s \prec L_{i+1}/clk{\uparrow}$

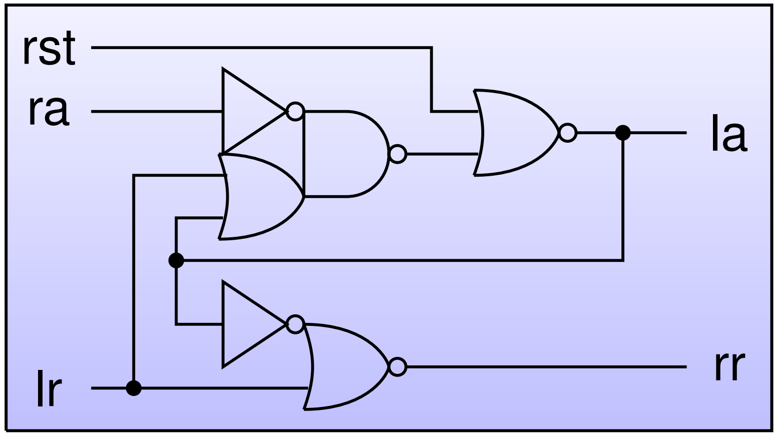
Relative Timing technology supports DI and bundled data styles. However, productivity, area, power, synthesis and library issues all favor the bundled data style.

Relative Timing Technology

Only method that supports integrated asynchronous and clocked methodologies within traditional ASIC CAD flows

Circuit Level Tools/Flows

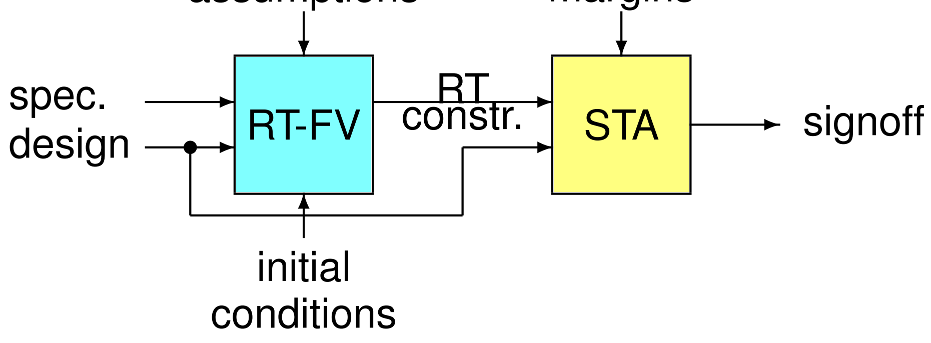
- optimized circuits & protocols
- relative timing characterization
    - formal proofs of correctness
- timing mapped to traditional CAD



System Level Tools/Flows

- design multi-frequency systems
- formal verification of system protocols
- mixed clock & async design flows

$$\underbrace{\text{clk}_i \mapsto \text{data} \prec \text{clk}_{i+1}}_{\text{timing assumptions}} \underbrace{+m}_{\text{absolute margins}}$$



Multi-Synchronous is Fundamental

Timing is where the rubber hits the road

- Best *multi*-synchronous is *a*-synchronous
  - Asynchronous design is continuous in time
  - No penalty for moving between frequency domains
  - Enables energy efficient, small solutions
- RT produces multi-synchronous compatibility with clocked EDA
  - Same CAD, map timing constraints as sdc
  - Timing driven flow using clocked tools
  - Compatible with any cell library
  - New circuit templates and support CAD

Relative Timed Productivity vs. Creativity

Hide the Complexity

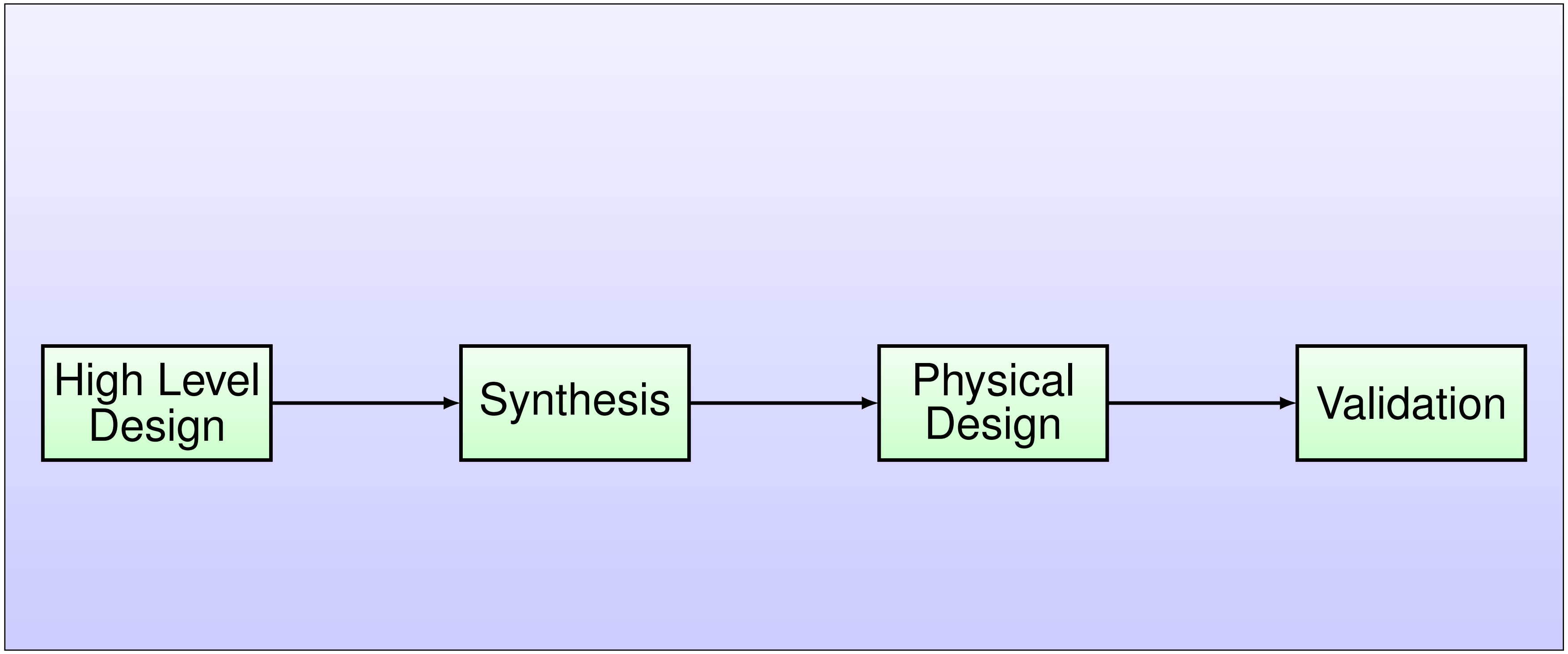
```
set d0_fdel 0.600
set d0_fdel_margin [expr $d0_fdel + 0.050]
set d0_bdel 0.060

set_size_only -all_instances [find -hier cell lc1]
set_size_only -all_instances [find -hier cell lc3]
set_size_only -all_instances [find -hier cell lc4]

set_disable_timing -from A2 -to Y [find -hier cell lc1]
set_disable_timing -from B1 -to Y [find -hier cell lc1]
set_disable_timing -from A2 -to Y [find -hier cell lc3]
set_disable_timing -from B1 -to Y [find -hier cell lc3]

set_max_delay $d0_fdel -from a -to l0/d
set_max_delay $d0_fdel -from b -to l0/d
set_min_delay $d0_fdel_margin -from lr -to l0/clk
set_max_delay $d0_bdel -from lr -to la
#margin 0.050 -from a -to l0/d -from lr -to l0/clk
#margin 0.050 -from b -to l0/d -from lr -to l0/clk
```

Retain the modularity

Nathan Sawaya, Lego Artist

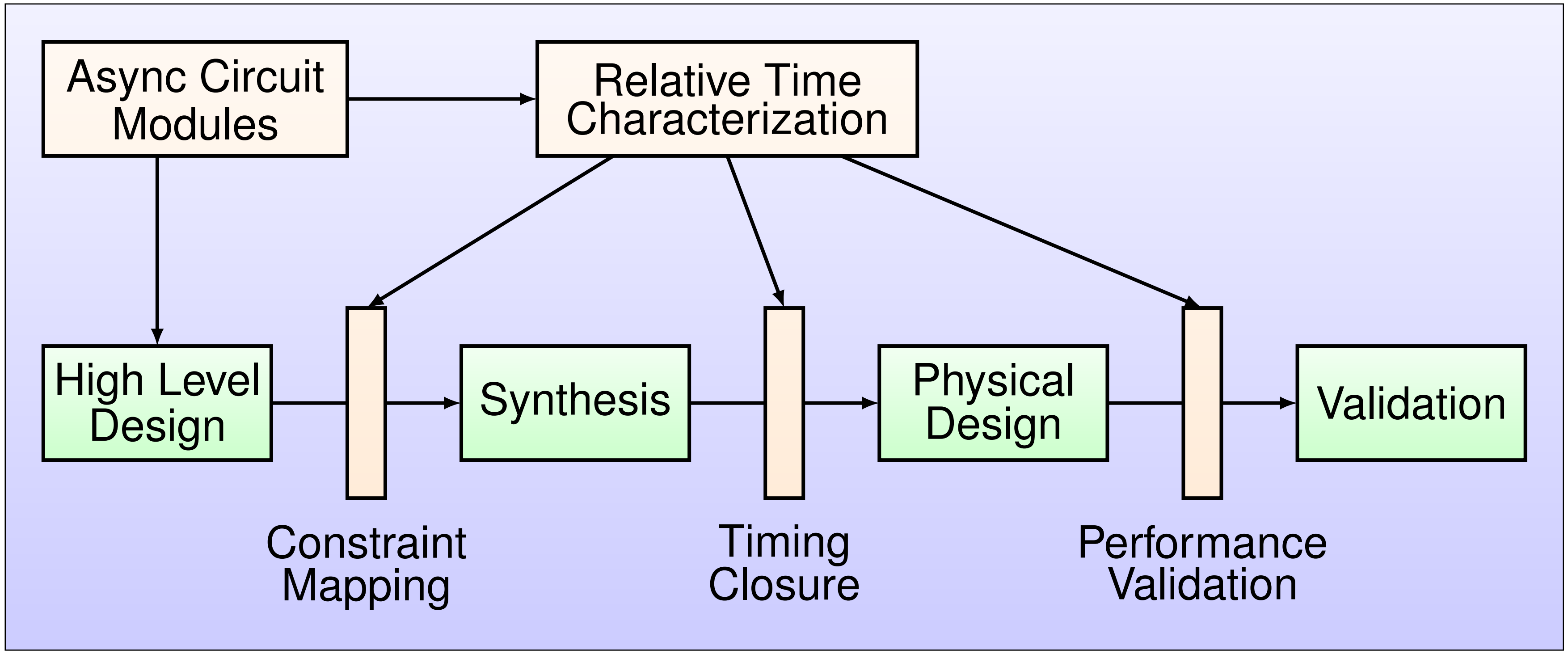
Simplified RT Design Flow

ASIC Flow used as an example: Clocked Design



Simplified RT Design Flow

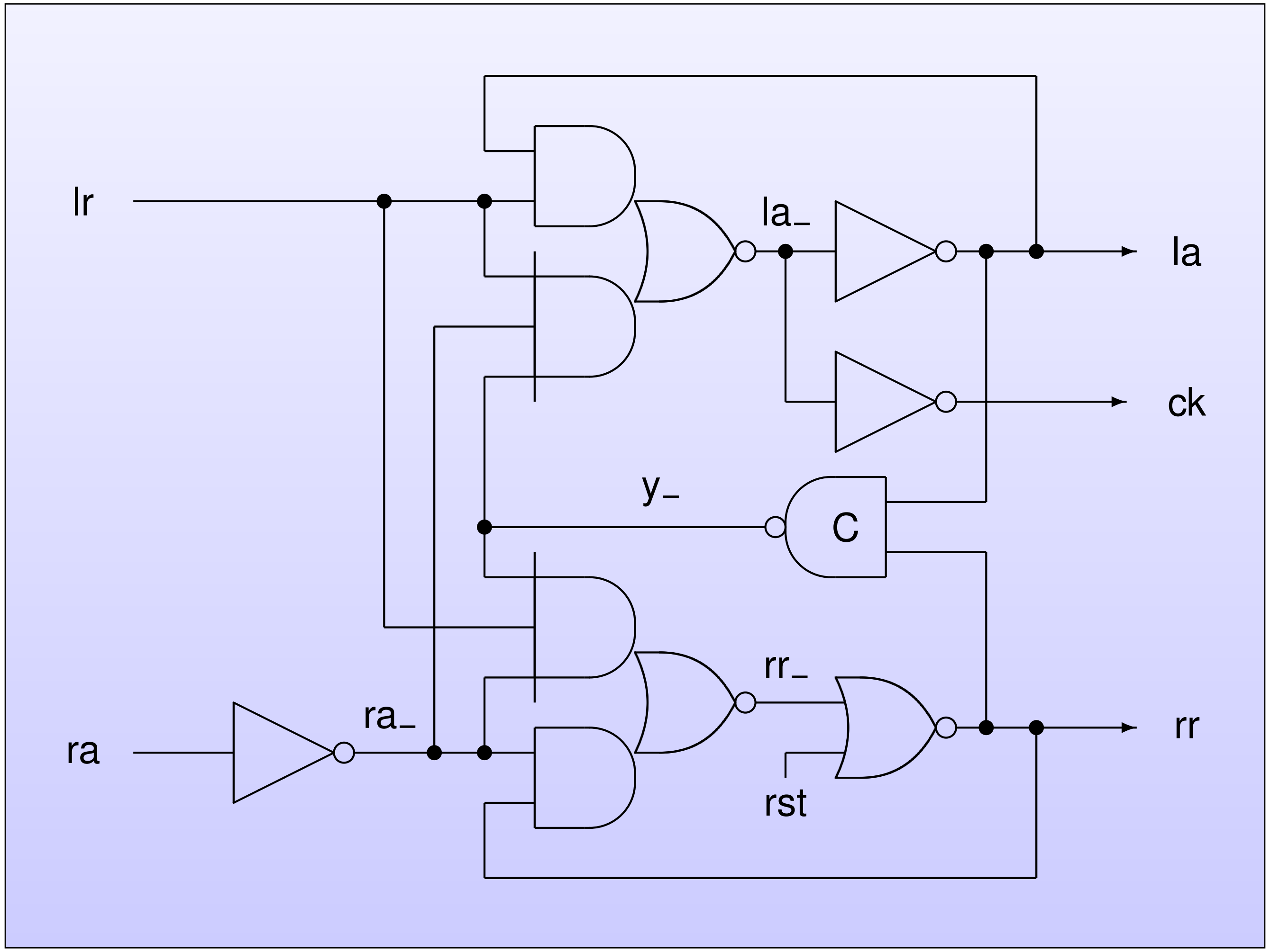
ASIC Flow used as an example: Async / Clocked Design



Works for custom flows as well.

RT Characterization: Pipeline Control Example

Example pipeline controller:



Structural Design Modules

Requires **structural** asynchronous design modules

```
module pipe_ctl (lr, la, rr, ra, ck, rst);
    input          lr, ra, rst;
    output         la, rr, ck;
    INVX1A12TH     lc0   (.A(ra), .Y(ra_));
    AOI32X1A12TH   lc1   (.A0(lr), .A1(ra_), .A2(y_), .B0(lr), .B1(la), .Y(la_));
    INVX1A12TH     lc2   (.A(la_), .Y(la));
    AOI32X1A12TH   lc3   (.A0(ra_), .A1(lr), .A2(y_), .B0(ra_), .B1(rr), .Y(rr_));
    NOR2X1A12TH    lc4   (.A(rr_), .B(rst), .Y(rr));
    c_element_     lc5   (.A(la), .B(rr), .Y(y_));
    INVX1A12TH     lc6   (.A(la_), .Y(ck));
endmodule // pipe_ctl
```

These are now inserted into a “standard” Verilog design.

Characterization and Constraint Mapping

```
set d0_fdel 0.600
set d0_fdel_margin [expr $d0_fdel + 0.050]
set d0_bdel 0.060

set_size_only -all_instances [find -hier cell lc1]
set_size_only -all_instances [find -hier cell lc3]
set_size_only -all_instances [find -hier cell lc4]

set_disable_timing -from A2 -to Y [find -hier cell lc1]
set_disable_timing -from B1 -to Y [find -hier cell lc1]
set_disable_timing -from A2 -to Y [find -hier cell lc3]
set_disable_timing -from B1 -to Y [find -hier cell lc3]

set_max_delay $d0_fdel -from a -to l0/d
set_max_delay $d0_fdel -from b -to l0/d
set_min_delay $d0_fdel_margin -from lr -to l0/clk
set_max_delay $d0_bdel -from lr -to la
#margin 0.050 -from a -to l0/d -from lr -to l0/clk
#margin 0.050 -from b -to l0/d -from lr -to l0/clk
```

“Lego” Design Flow

Simple *conceptual* asynchronous design almost same as clocked:

- Replace clocked pipeline `always @ (posedge clock)` with
  1. asynchronous pipeline controller template
  2. memory array (usually latch bank) template
  3. handshake steering logic templates

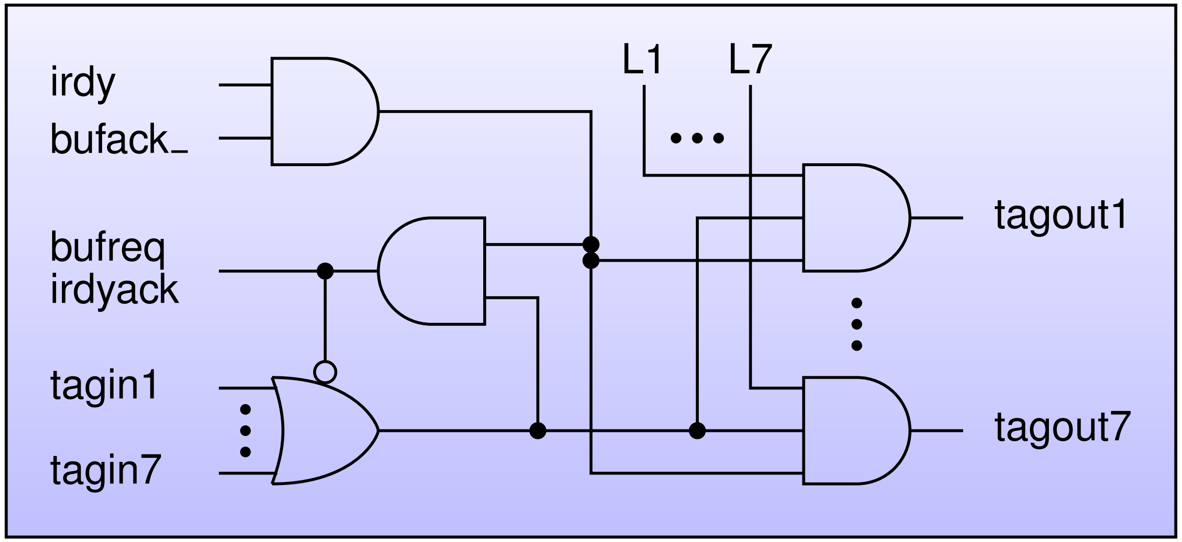
  Becomes a “schematic” design style
- Add behavioral datapath
- Create async architecture
  - optimize design frequencies, . . .

Then we can create architectures using traditional CAD flows.

Concurrency and Time

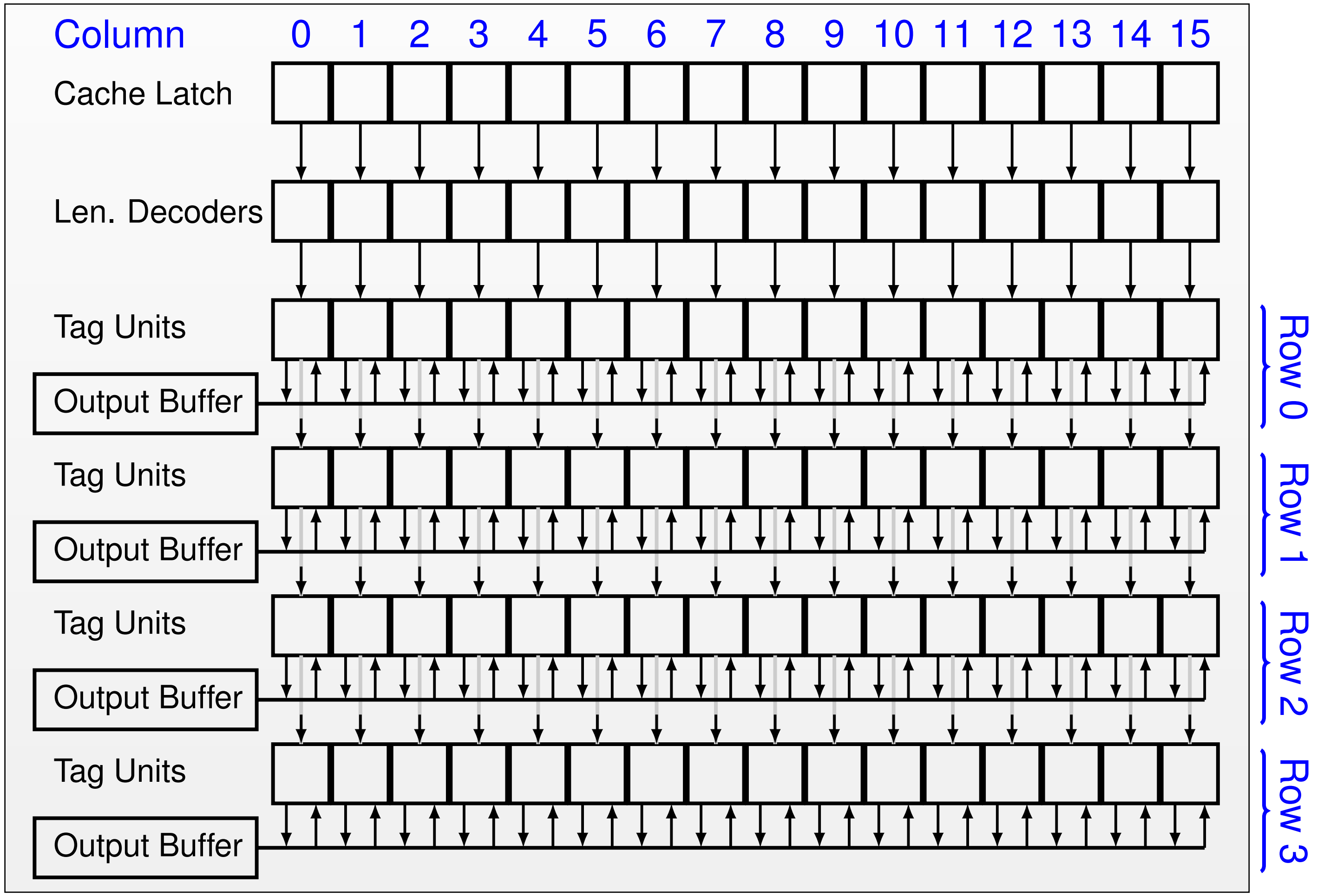
1. ***What is the relationship between time and concurrency in an integrated circuit system?***
2. ***Can time be exploited to improve a design or protocol?***

Observation: System faster if assume logic faster than cycle time: note 7-input domino OR gate, cell operates at 3.6GHz in 250nm



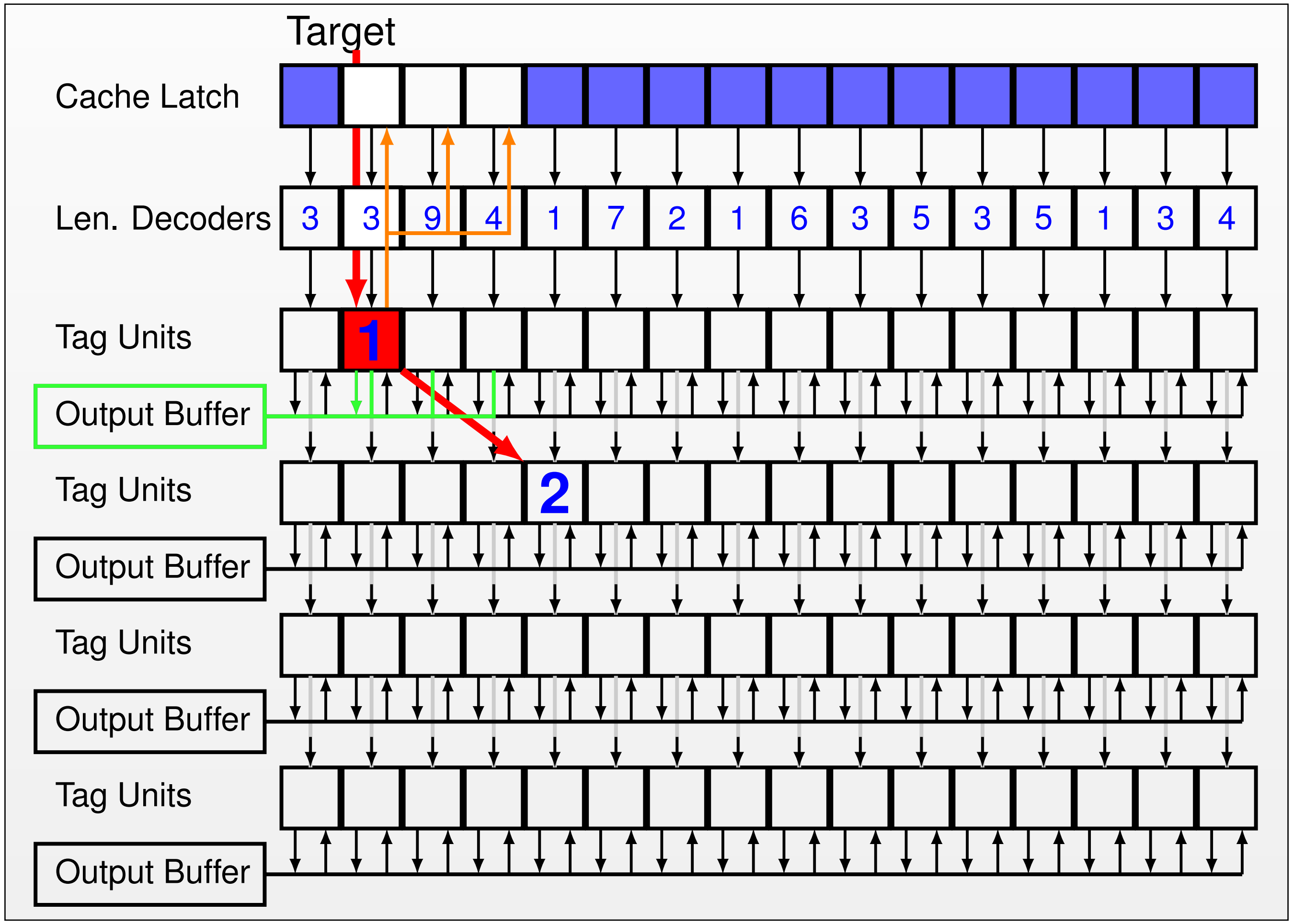
Concurrency and Time

Architectural level timing experiment: Pentium front end



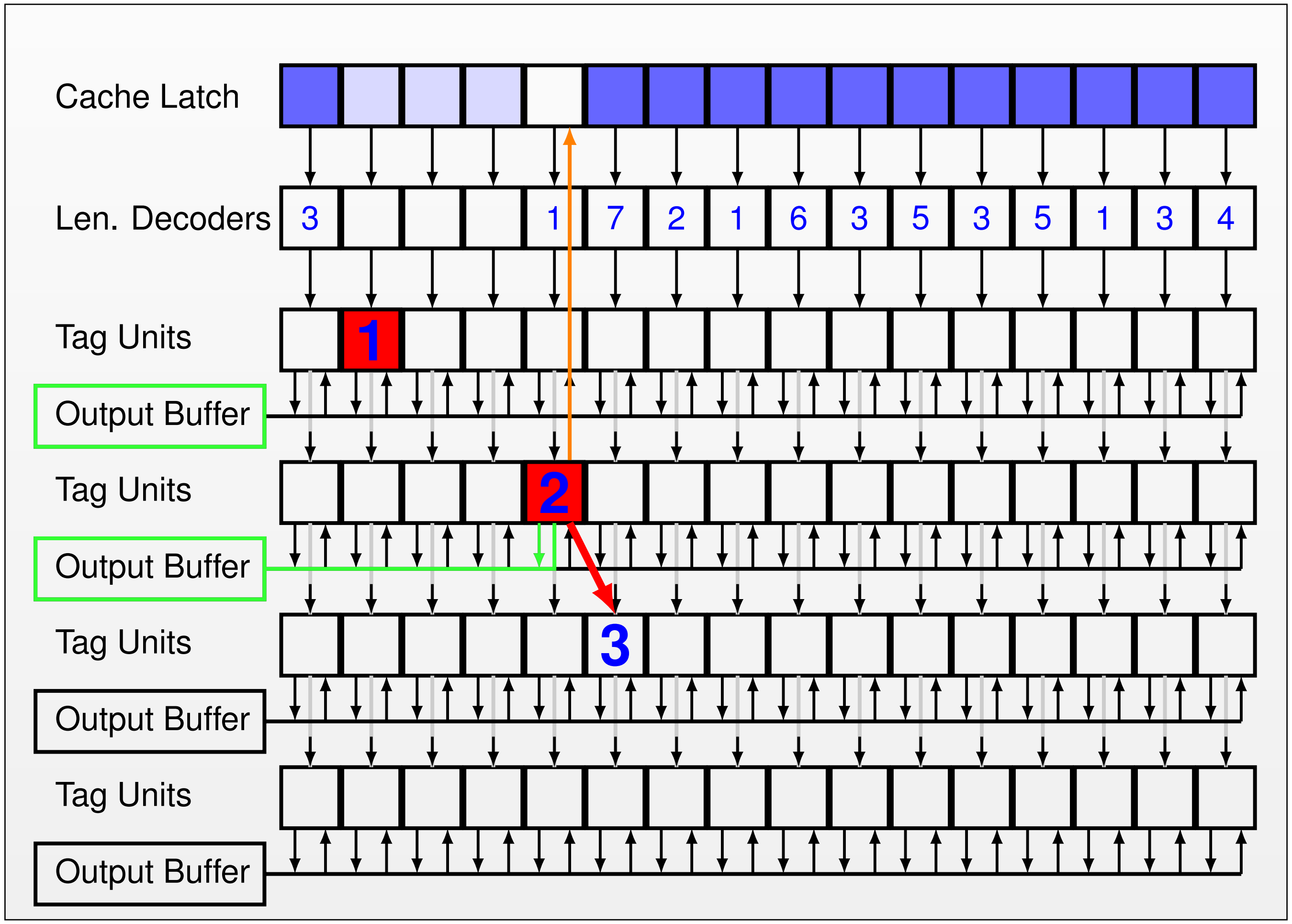
Concurrency and Time

Architectural level timing experiment: Pentium front end



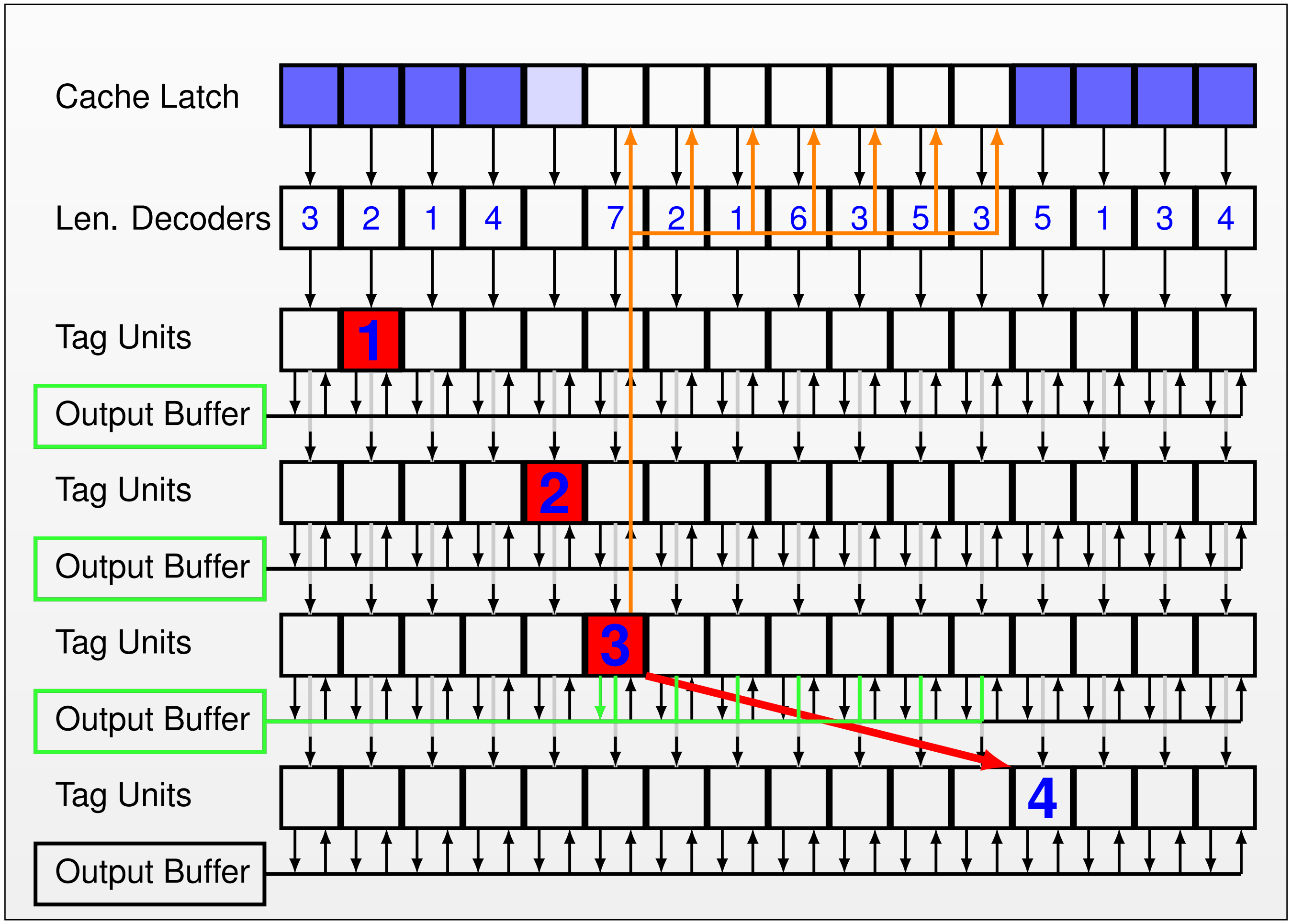
Concurrency and Time

Architectural level timing experiment: Pentium front end



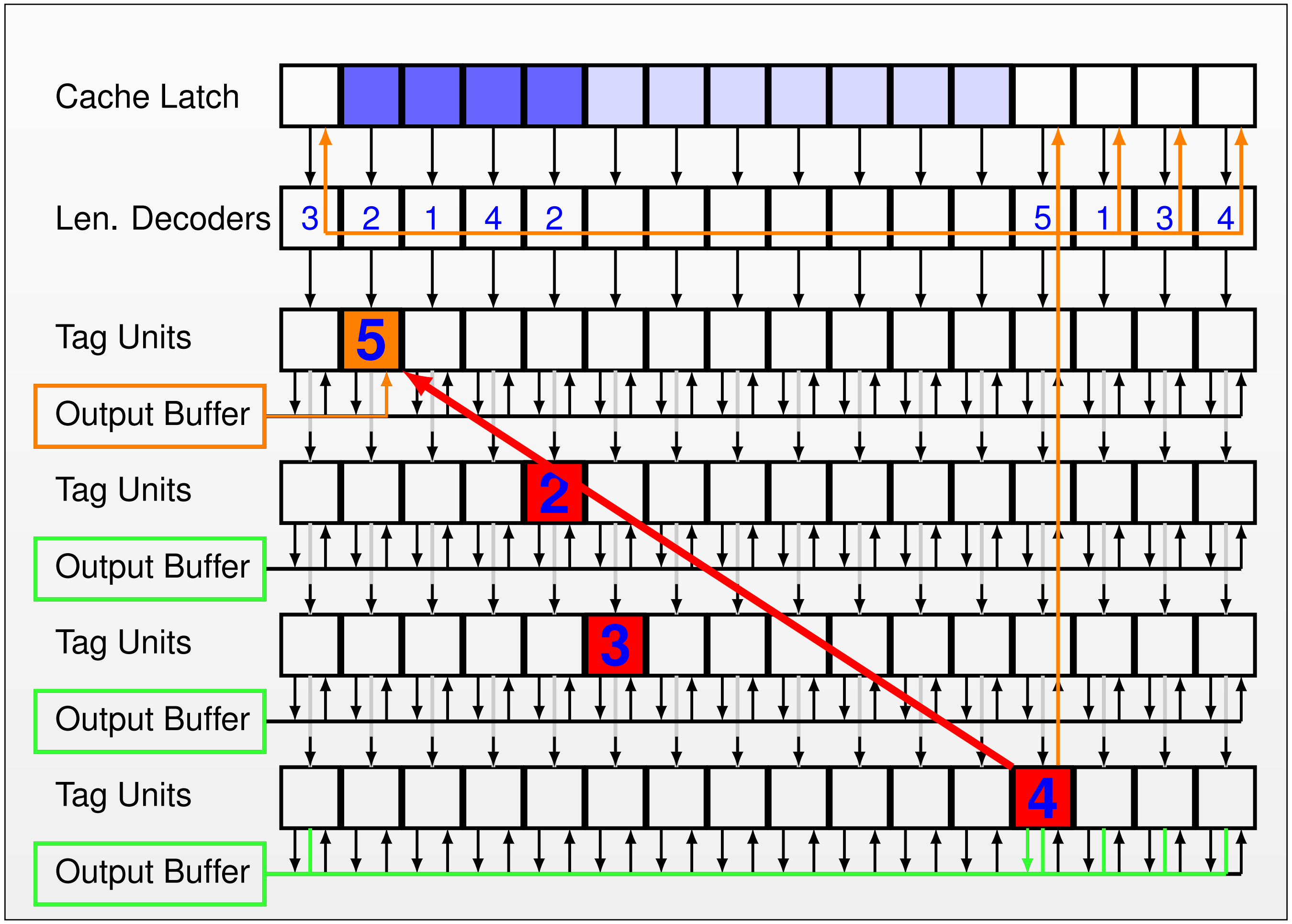
Concurrency and Time

Architectural level timing experiment: Pentium front end



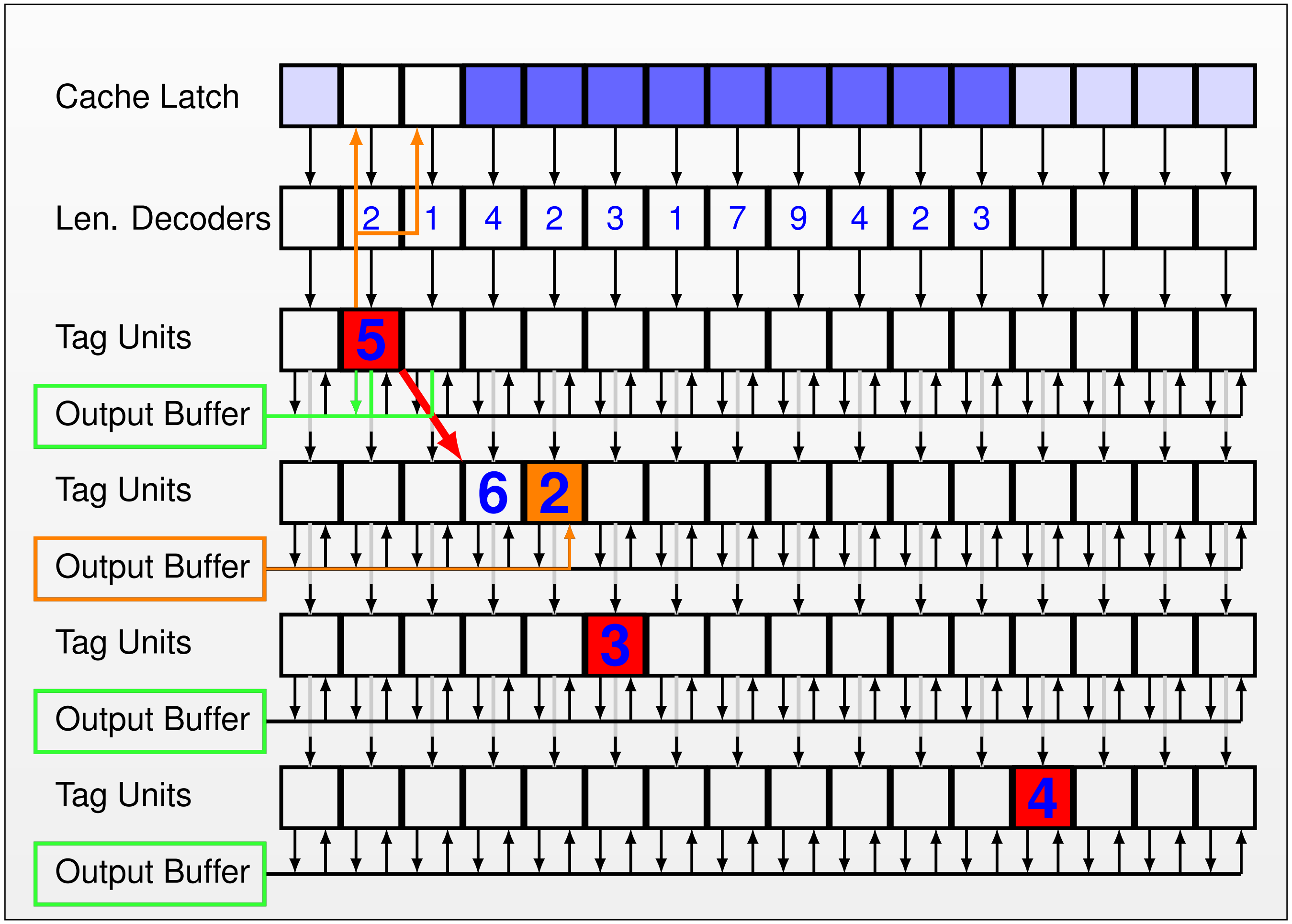
Concurrency and Time

Architectural level timing experiment: Pentium front end



Concurrency and Time

Architectural level timing experiment: Pentium front end



Timed Asynchronous Designs

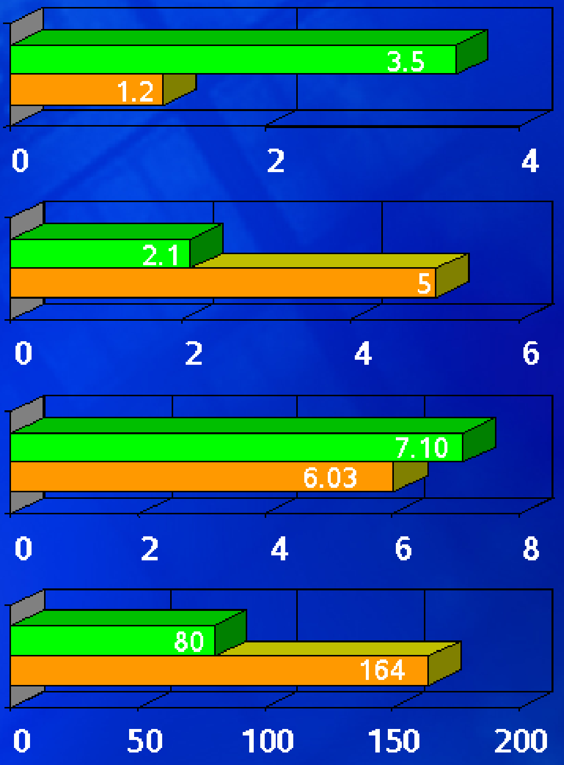
- 1997 RAPPID Si: 0.25µ, 1.8V, 35°C, for common instr
- Comparing to 400MHz Deschutes Processor

Throughput [Inst./nS]

Latency [nS]

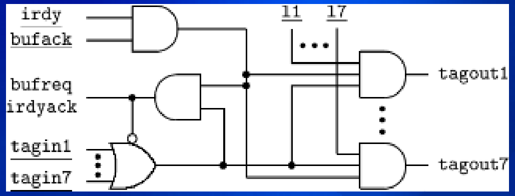
Area [mm²]

Power [nJ]



🟩	RAPPID
🟧	Clocked

Testability: 95.9% (BIST stuck-at)



Key pipeline circuit

intel

Multi-rate 64-Point FFT Architecture

Initial design target: high performance military applications

- Mathematically based on $W_N = e^{-j\frac{2\pi}{N}}$ notation
- Hierarchical multi-rate design: $N = N_1 N_2$
- Decimate frequency ($\downarrow$) by N_2
  - operate on N_2 low frequency streams
- Transmute data & frequency to N_1 low frequency streams
- Expand ($\uparrow$) by N_1 to reconstruct original frequency stream

Design Models

Hierarchical derivation of multi-frequency design:

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \left[W_N^{m_1 n_2} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) W_{N_1}^{m_1 n_1} \right] W_{N_2}^{m_2 n_2}$$

- N_2 FFTs using N_1 values as the inner summation
- Scaled and used to produce N_1 FFTs of N_2 values

Hierarchically scale design

- Base case when $N = 4$, $X(m) = W^4 x(n)$
- 4-point FFT performed without multiplication
  - Multiplication constants W^4 become ± 1

FFT-64

Implemented on IBM's 65nm 10sf process, Artisan academic library

Three design blocks:

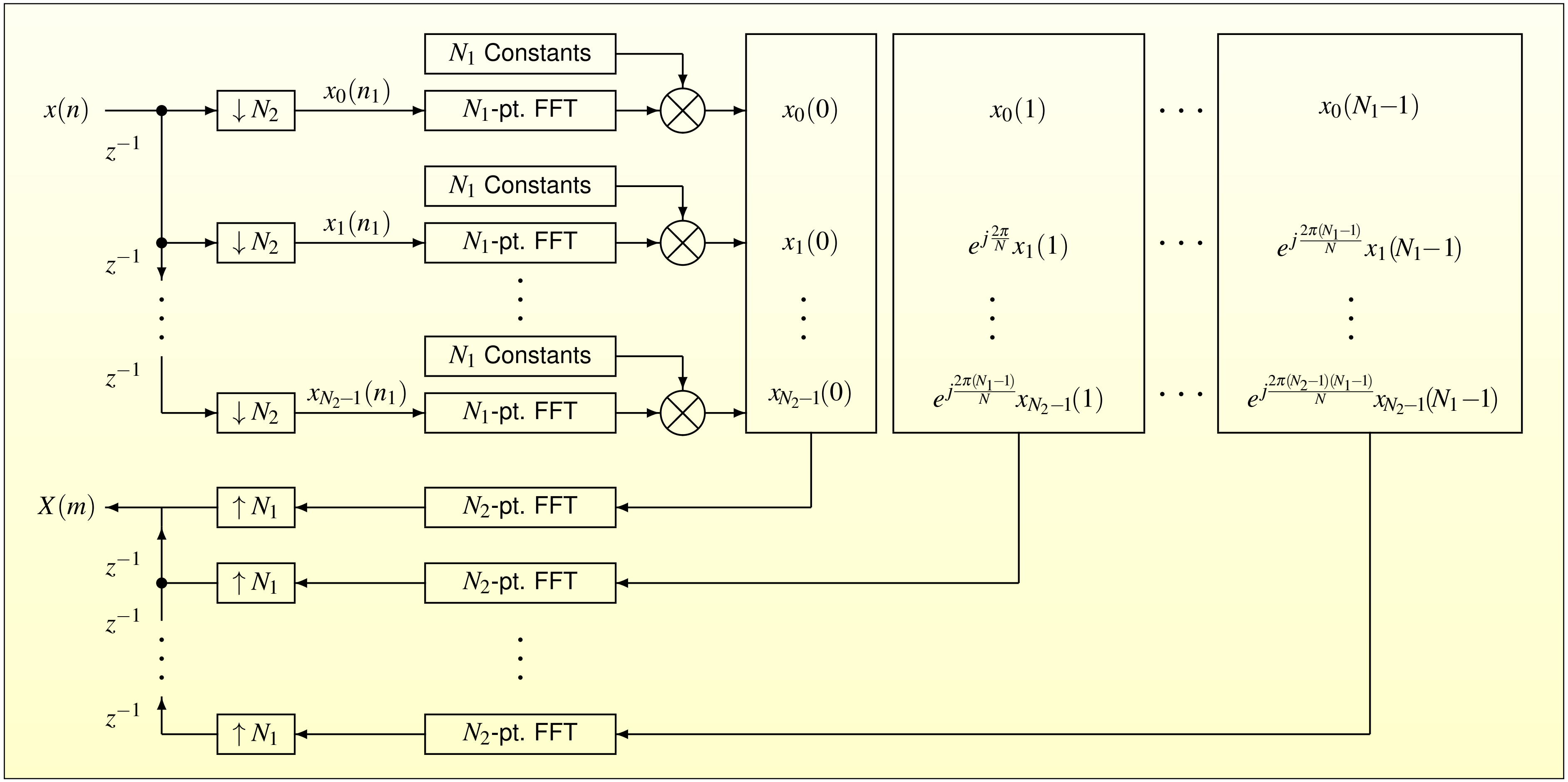
- FFT-4
- FFT-16 $N_1, N_2 = 4$
- FFT-64 $N_1 = 16, \quad N_2 = 4$

Two designs:

- Clocked Multi-Synchronous
- Relative Timed Multi-Synchronous
    - near identical architectures
    - additional RT area / pipeline optimized version for FFT-64

General Multi-rate FFT Architecture

1.25GHz　　313MHz　　313MHz to 78MHz

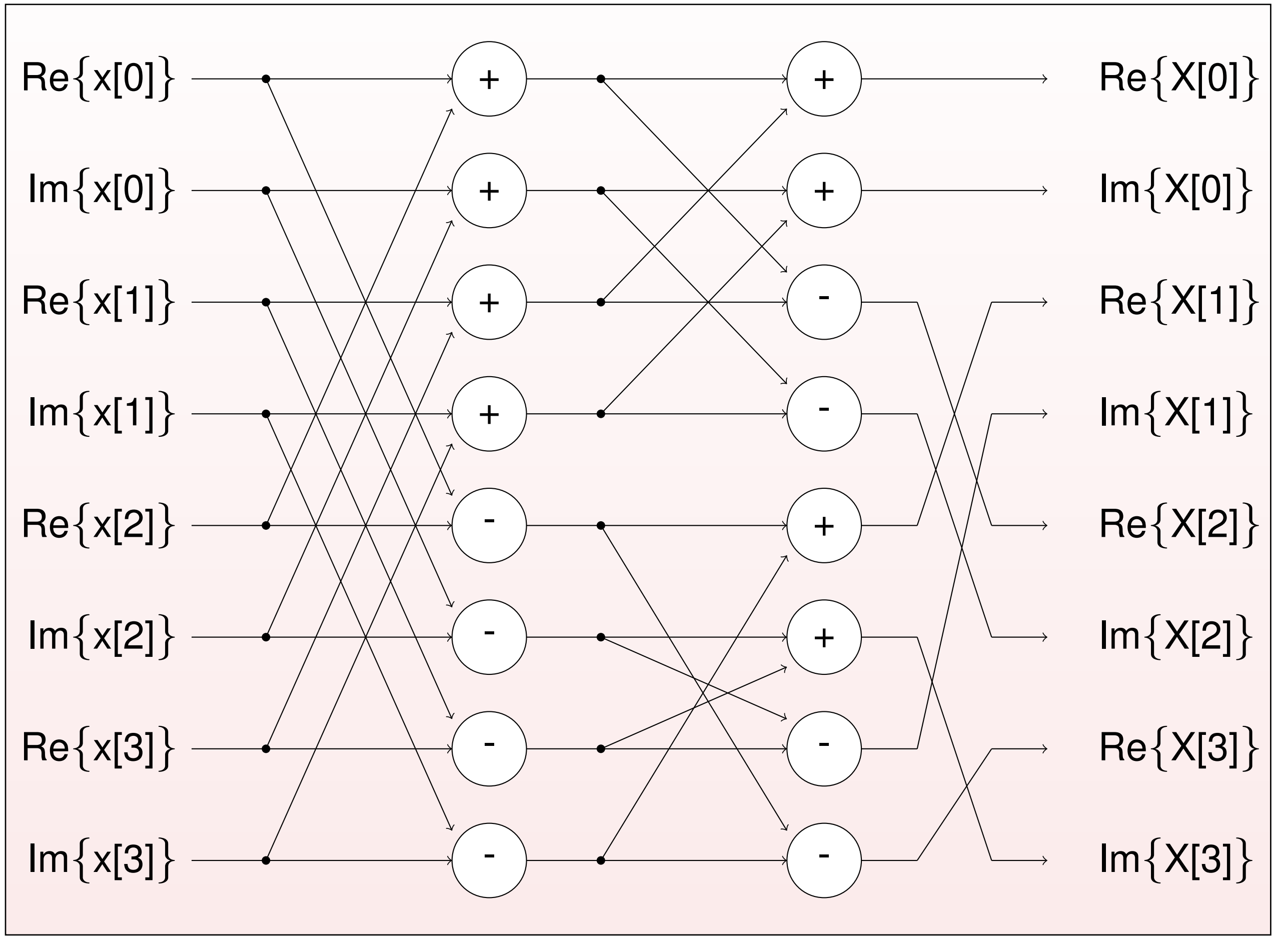


1.25GHz　　78MHz

ASIC tool flow, 65nm technology

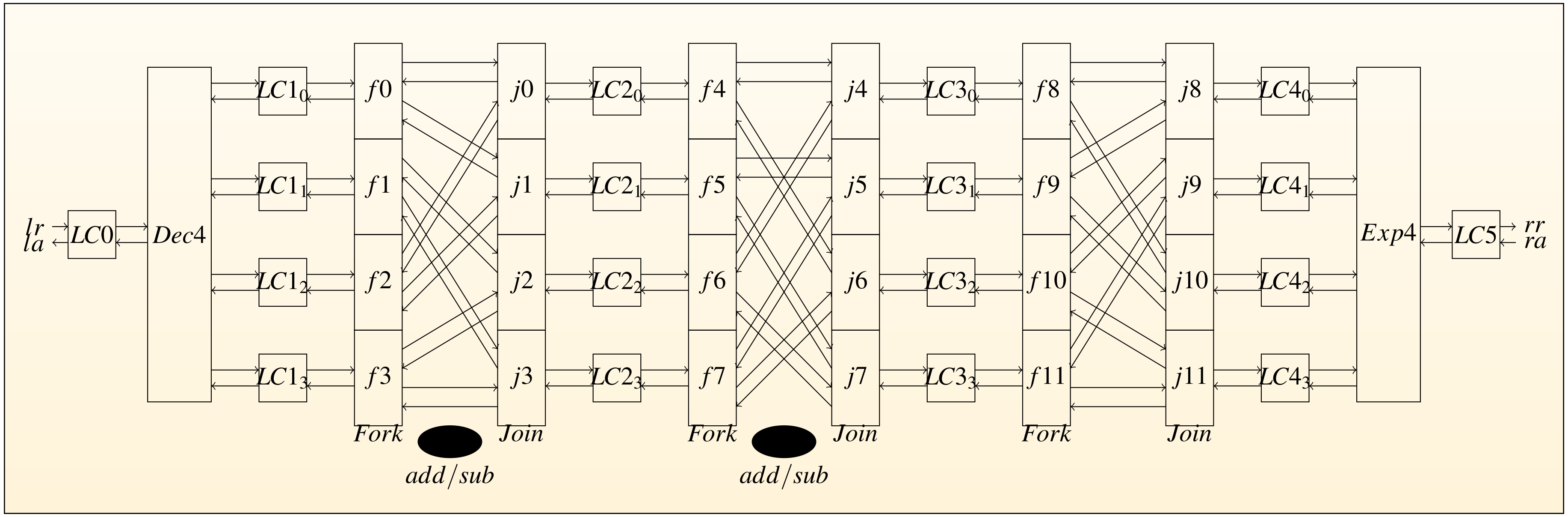
FFT-4 Building Block

Data flow graph of pipelined 4-Point FFT design:



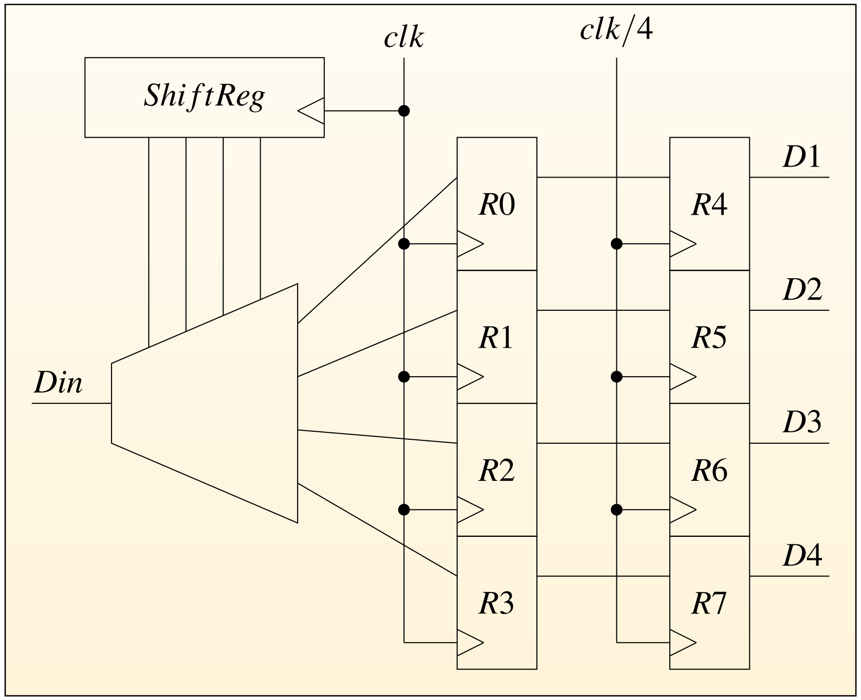
Pipelined Asynchronous 4-Point Architecture

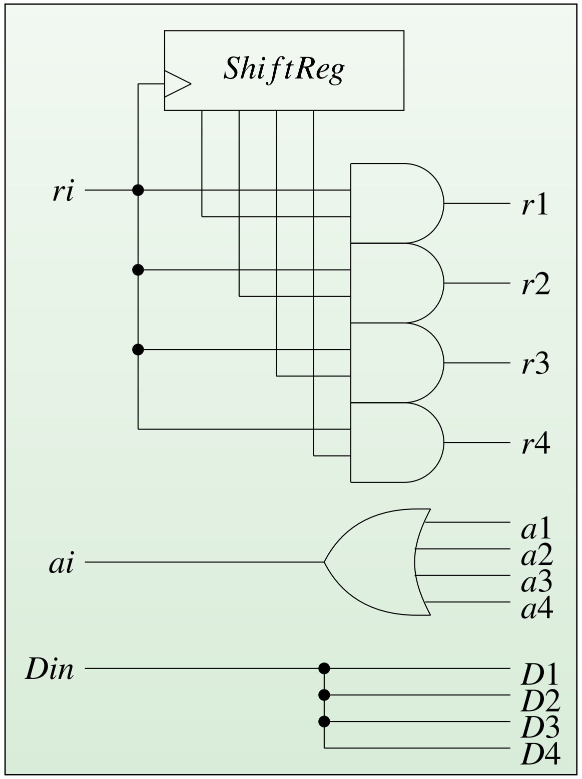
- Operates at 1/4 the input frequency
- Synchronization occurs between decimated rows
    - Fast internal pipeline stages essential



Decimator-4 Design Comparison

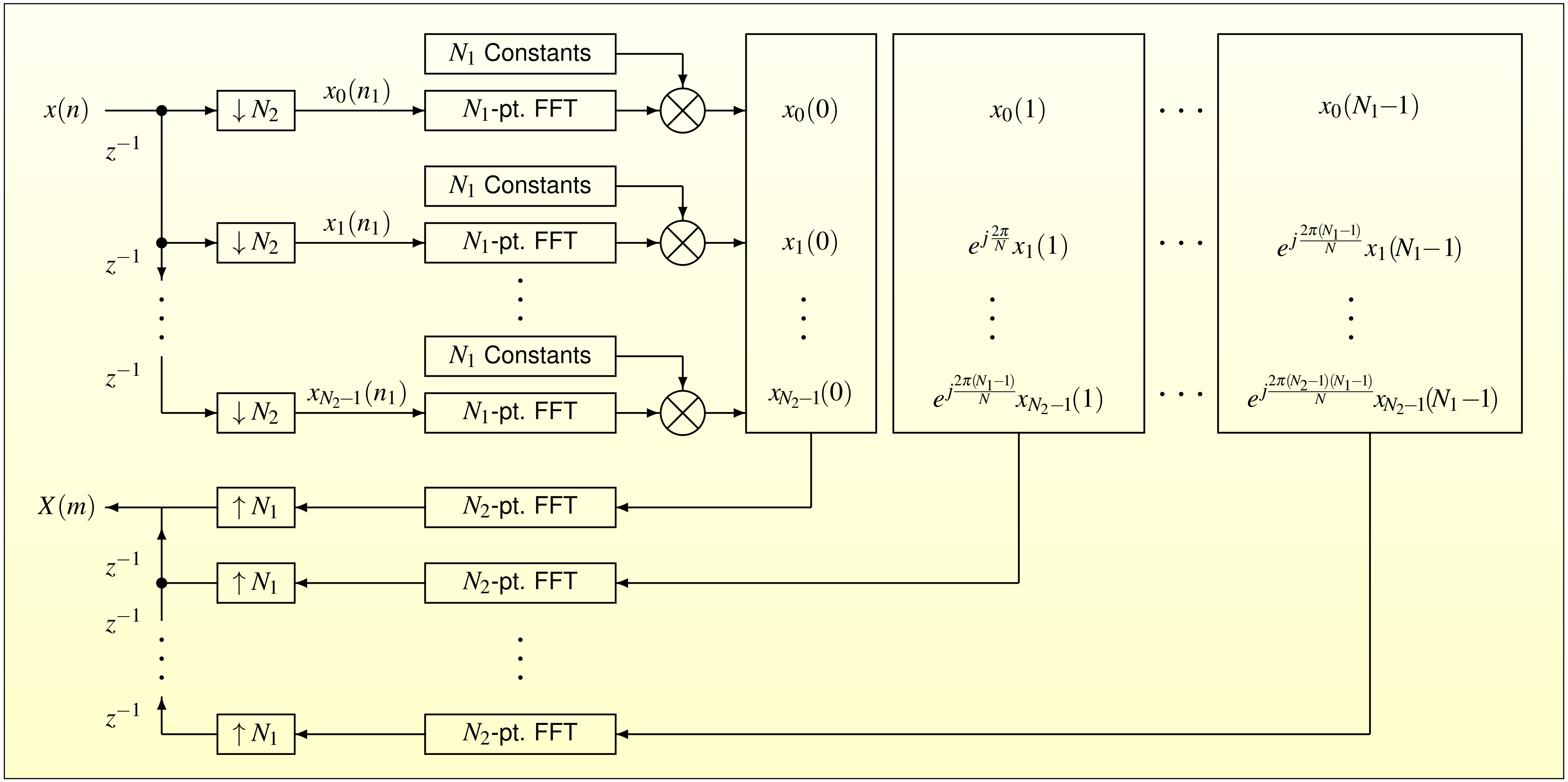
- Clocked block requires pipeline to change frequency
- Async block latency combinational and concurrent





General Multi-rate FFT Architecture

1.25GHz 313MHz 313MHz to 78MHz



1.25GHz 78MHz

ASIC tool flow, 65nm technology

Coding Like Schematic Capture

```
module FFT_64 (ri, ai, DI, ro, ao, DO, reset);
   linear_control tk0 (.lr_(ri), .la_(ai), .rr_(p0r), .ra_(p0a), .ck(ck0), .rst(reset));
   latch P0 (.d(DI), .clk(ck0), .q(P0D0));
   decimator_2 D2_00 (.dI(P0D0), .d1(P00D1), .d2(P00D2), .ri(p0r), .ai(p0a), .r1(p00r1), .r2(p00r2),
                      a1(p00a1), .a2(p00a2), .reset(reset));
   linear_control tk00 (.lr_(p00r1), .la_(p00a1), .rr_(p01r), .ra_(p01a), .ck(ck00_1), .rst(reset));
   latch P00 (.d(P00D1), .clk(ck00_1), .q(P01D1));
   decimator_2 D2_01 (.dI(P01D1), .d1(P0DT1), .d2(P0DT3), .ri(p01r), .ai(p01a), .r1(p0rt1), .r2(p0rt3),
                      a1(p0at1), .a2(p0at3), .reset(reset));
   linear_control tk01 (.lr_(p00r2), .la_(p00a2), .rr_(p02r), .ra_(p02a), .ck(ck00_2), .rst(reset));
   latch P01 (.d(P00D2), .clk(ck00_2), .q(P01D2));
   decimator_2 D2_02 (.dI(P01D2), .d1(P0DT2), .d2(P0DT4), .ri(p02r), .ai(p02a), .r1(p0rt2), .r2(p0rt4),
                      a1(p0at2), .a2(p0at4), .reset(reset));

   FFT_16 F16_0 (.ri(p0rt1), .ai(p0at1), .dI(P0DT1), .ro(p1rt1), .ao(p1at1), .dO(P1DT1), .reset(reset));
   FFT_16 F16_1 (.ri(p0rt2), .ai(p0at2), .dI(P0DT2), .ro(p1rt2), .ao(p1at2), .dO(P1DT2), .reset(reset));
   FFT_16 F16_2 (.ri(p0rt3), .ai(p0at3), .dI(P0DT3), .ro(p1rt3), .ao(p1at3), .dO(P1DT3), .reset(reset));
   FFT_16 F16_3 (.ri(p0rt4), .ai(p0at4), .dI(P0DT4), .ro(p1rt4), .ao(p1at4), .dO(P1DT4), .reset(reset));

   linear_control tk2_0 (.lr_(p1rt1), .la_(p1at1), .rr_(p2rt1), .ra_(p2at1), .ck(ck1_0), .rst(reset));
   latch P2_0 (.d(P1DT1), .clk(ck1_0), .q(P2DT1));

   CB64_1 CB_1 (.update(p1at2), .dO(CDT2), .en(endt2), .reset(reset));
   comp_mult CM_1 (.A(P1DT2), .B(CDT2), .P(CP2), .en(endt2));
   linear_control tk2_1 (.lr_(p1rt2), .la_(p1at2), .rr_(p2rt2), .ra_(p2at2), .ck(ck1_1), .rst(reset));
   latch P2_1 (.d(CP2), .clk(ck1_1), .q(P2DT2));

   CB64_2 CB_2 (.update(p1at3), .dO(CDT3), .en(endt3), .reset(reset));
   comp_mult CM_2 (.A(P1DT3), .B(CDT3), .P(CP3), .en(endt3));
```

Results

The 16-point FFT Comparison Result (* values are scaled ideally to 65 nm technology)

	Points	Word	Time for 1K-point	Clock	Tech.	Energy/point	Area	Power	Energy	Area	Throughput
		bits	μs	MHz	nm	$pJ/data-point$		mW	Benefit	Benefit	Benefit
Our Design(Async)	16-1024	32	0.83	1274	65	25.05	54 Kgates	30.9	8.01	2.77	8.32
Our Design(clock)	16-1024	32	1.73	588	65	41.83	71 Kgates	24.7	4.8	2.07	3.98
Guan [1]	16-1024	16	6.91*	653*	130	200.68	147 Kgates	29.7*	1	1	1

The 64-point FFT Comparison Result (* values are scaled ideally to 65 nm technology)

	Points	Word	Time for 1K-point	Clock	Tech.	Energy/point	Area	Power	Energy	Area	Throughput
		bits	μs	MHz	nm	$pJ/data-point$		mW	Benefit	Benefit	Benefit
Our Design(Async-opt)	64-1024	32	0.93	1284	65	62.41	0.41 mm^2	68.5	6.1	0.46	30.16
Our Design(Async)	64-1024	32	0.84	1366	65	59.94	0.50 mm^2	72.9	6.35	0.38	33.42
Our Design(clock)	64-1024	32	3.13	588	65	246.75	1.16 mm^2	80.7	1.54	0.16	8.99
Baireddy [2]	64-4096	-	28.14*	514*	90	380.88	0.19 mm^{2*}	13.86*	1	1	1

The 64-point async-opt design contains 229k gates, our clocked 454k.
* For comparison, these designs were scaled to a 65nm process by doubling the frequency and halving the power in the 130nm technology, and multiplying frequency, power and area in the 90nm design by 1.43, 0.7, and 0.49 respectively.

[1] X. Guan, Y. Fei, and H. Lin, “Hierarchical Design of an Application-Specific Instruction Set Processor for High-Throughput and Scalable FFT Processing” in IEEE *Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 20, No. 3, pp. 551–563, march 2012.

[2] V. Baireddy, H. Khasnis, and R. Mundhada, “A 64-4096 point FFT/IFFT/Windowing Processor for Multi Standard ADSL/VDSL Applications”, in IEEE *International symposium on Signals, Systems and Electronics (ISSSE’07)*, pp. 403–405, 2007.

Commercializing Multi-Synchronous ICs

1. **Timing is a key method of gaining $e\tau^2$ improvement**
   - Multi-synchronous allows best optimization of design
   - Exploit affect of time in our circuits and architectures
   - *Relative Timing* supports all time methods & models
2. **Utilize best capabilities in industry**
   - No change plus minimal enhancements to EDA / CAD / flows
   - Cell libraries unmodified
3. **Leverage designer's creativity**
   - Provide familiar design environment
   - Enhance modularity and design visibility
   - Do not restrict circuits, architectures, flows

Multi-Synchronous Wins Big

1. **Efficiency** in **power** and **performance** is new game in town
2. We need to think about problems differently
3. New timing model is one excellent path to progress
4. Multi-synchronous design gives ave. 10$\times$ $e\tau^2$ improvement
   - Pentium: $e\tau^2 = 17.5\times$
   - FFT: $e\tau^2 = 16.9\times$

Design	Energy	Area	Freq.	Latency	Aggregate
Pentium F.E.	2.05	*0.85*	2.92	2.38	12.11$\times$
64-pt FFT	3.95	2.83	2.07	3.37	77.98$\times$