Relative Timing Driven Multi-Synchronous Design: Enabling Order-of-Magnitude Energy Reduction

Kenneth S. Stevens
University of Utah
Granite Mountain Technologies
Learn from Prof. Kajitana

- Think differently and deeply
- Apply thought to current challenges

Then collaborate

Goals of Presentation:

1. Define and propose “rule breaker” idea
2. Request support from physical design community
Multi-Synchronous Advantage

1. **Efficiency** in *power* and *performance* is new game in town

2. Multi-synchronous design provides optimization opportunity

3. New (asynchronous) timing model is one excellent path

4. Produces average $10 \times e\tau^2$ improvement
   - Pentium: $e\tau^2 = 17.5 \times$
   - FFT: $e\tau^2 = 16.9 \times$

5. But ... *need improved physical design support*

<table>
<thead>
<tr>
<th>Design</th>
<th>Energy</th>
<th>Area</th>
<th>Freq.</th>
<th>Latency</th>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium F.E.</td>
<td>2.05</td>
<td>0.85</td>
<td>2.92</td>
<td>2.38</td>
<td>12.11×</td>
</tr>
<tr>
<td>64-pt FFT</td>
<td>3.95</td>
<td>2.83</td>
<td>2.07</td>
<td>3.37</td>
<td>77.98×</td>
</tr>
</tbody>
</table>
Timing is a Key Issue

Multi-synchronous design produces best results

Single frequency, low skew (small blocks, standard CAD)
1. global block frequencies
2. higher clock power
3. clock design, distribution

Multiple frequencies (SoC reality – localization)
1. blocks operate at best frequency
2. network not synchronized
3. synchronizing FIFOs
Energy Efficient Design

Wine goblet model:

- Energy efficiency has two primary sources
  - System architecture
  - Physical design
- Methodology and CAD unify sources

Best realization:

- Multi-synchronous
  - Defined by system’s critical path
  - Then optimal local power-delay
  - Asynchronous best methodology:
    - no synchronization cost
Interface Matters!

Clocked design requires synchronizers when crossing all domains.

Major location for buffering in a design.
Interface Matters!

No synchronization required into async domain.

Improves power, performance, and modularity
Timed Asynchronous Designs

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

<table>
<thead>
<tr>
<th></th>
<th>Throughput [Inst./nS]</th>
<th>Latency [nS]</th>
<th>Area [mm²]</th>
<th>Power [nJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAPPID</td>
<td>1.2</td>
<td>2.1</td>
<td>6.03</td>
<td>80</td>
</tr>
<tr>
<td>Clocked</td>
<td>3.5</td>
<td>5.0</td>
<td>7.10</td>
<td>164</td>
</tr>
</tbody>
</table>

Testability: 95.9% (BIST stuck-at)

Key pipeline circuit

27 March 2013
UofU and GMT
Multi-Synchronous Architecture

1. Make architectural bottleneck as fast as possible.
2. Make the rest of the design match bottleneck
   - ...normally as slow as possible
3. Optimize locally for power/performance.

Asynchronous Pentium bottleneck circuit
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

27 March 2013 UofU and GMT
Concurrency and Time

Architectural level timing experiment: Pentium front end

27 March 2013 UofU and GMT
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?

- *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):

\[ \text{pod} \rightarrow \text{poc}_1 \prec \text{poc}_2 \]

Sequencing relative to common reference

- can now **evaluate** sequencing
- can now **control** sequencing
Relative Timing

1. Relative Timing
   - Sequences signals at poc (point of convergence)
   - Requires a common timing reference: pod (point of divergence)

2. Formal representation: $\text{pod} \mapsto \text{poc}_1 + \text{margin} < \text{poc}_2$

3. RT models timing in ALL systems
   - Clocked: $\text{pod} = \text{clock}$, $\text{poc} = \text{flops}$
   - Async: $\text{pod} = \text{request}$, $\text{poc} = \text{latches}$

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 $L_i/clk[i]\uparrow \rightarrow L_{i+1}/d+s \prec L_{i+1}/clk[i+1]$.

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

Clocked physical design directly supports the clocked Relative Timing constraints. The asynchronous circuit constraints must be provided as min and max constraints, and are not well supported.
Relative Timing Driven Flow

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
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: \( N = N_1N_2 \)
- Decimate frequency (↓) by \( N_2 \)
  - operate on \( N_2 \) low frequency streams
- Transmute data & frequency to \( N_1 \) low frequency streams
- Expand (↑) by \( N_1 \) to reconstruct original frequency stream
Design Models

Hierarchical derivation of multi-frequency design:

\[ X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \left[ W_N^{m_1n_2} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) W_{N_1}^{m_1n_1} \right] W_{N_2}^{m_2n_2} \]

- \( N_2 \) 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 \( \pm 1 \)
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

\[
x(n) \rightarrow \downarrow N_2 \rightarrow x_0(n_1) \rightarrow N_1 \text{ Constants} \rightarrow x_0(0) \rightarrow \cdots \rightarrow x_0(N_1-1)
\]

\[
x(n) \rightarrow \downarrow N_2 \rightarrow x_1(n_1) \rightarrow N_1 \text{ Constants} \rightarrow x_1(0) \rightarrow \cdots \rightarrow x_1(N_1-1)
\]

\[
x(n) \rightarrow \downarrow N_2 \rightarrow x_{N_2-1}(n_1) \rightarrow N_1 \text{ Constants} \rightarrow x_{N_2-1}(0) \rightarrow \cdots \rightarrow x_{N_2-1}(N_1-1)
\]

\[
X(m) \leftarrow \uparrow N_1 \rightarrow \downarrow N_2 \rightarrow N_2 \text{ Constants} \rightarrow X_{N_2-1}(m)
\]

1.25GHz  78MHz  ASIC tool flow, 65nm technology

27 March 2013  UofU and GMT
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

Multi-Synchronous asynchronous design smaller, faster, lower power
## Results

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

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

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

<table>
<thead>
<tr>
<th>Points</th>
<th>Word bits</th>
<th>Time for 1K-point</th>
<th>Clock MHz</th>
<th>Tech. nm</th>
<th>Energy/point pJ/data – point</th>
<th>Area</th>
<th>Power mW</th>
<th>Energy Benefit</th>
<th>Area Benefit</th>
<th>Throughput Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Design(Async-opt)</td>
<td>64-1024</td>
<td>32</td>
<td>0.93</td>
<td>1284</td>
<td>65</td>
<td>0.41 mm²</td>
<td>62.41</td>
<td>68.5</td>
<td>6.1</td>
<td>0.46</td>
</tr>
<tr>
<td>Our Design(Async)</td>
<td>64-1024</td>
<td>32</td>
<td>0.84</td>
<td>1366</td>
<td>65</td>
<td>0.50 mm²</td>
<td>59.94</td>
<td>72.9</td>
<td>6.35</td>
<td>0.38</td>
</tr>
<tr>
<td>Our Design(clock)</td>
<td>64-1024</td>
<td>32</td>
<td>3.13</td>
<td>588</td>
<td>65</td>
<td>1.16 mm²</td>
<td>246.75</td>
<td>80.7</td>
<td>1.54</td>
<td>0.16</td>
</tr>
<tr>
<td>Baireddy [2]</td>
<td>64-4096</td>
<td>-</td>
<td>28.14*</td>
<td>514*</td>
<td>90</td>
<td>0.19 mm²</td>
<td>380.88</td>
<td>13.86*</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The 64-point async-opt design contains 229k gates, our clocked 454k.

* For comparison, these designs were scaled to a 65nm process by scaling frequency, power, and area in the 130nm technology by 2.0, 0.5, 0.25 ×, and in the 90nm design by 1.43, 0.7, and 0.49 × respectively.


Multi-Synchronous Advantage

1. Efficiency in power and performance is new game in town

2. Multi-synchronous design provides optimization opportunity

3. New (asynchronous) timing model is one excellent path

4. Produces average $10 \times e\tau^2$ improvement
   - Pentium: $e\tau^2 = 17.5 \times$
   - FFT: $e\tau^2 = 16.9 \times$

5. But ... need improved physical design support

<table>
<thead>
<tr>
<th>Design</th>
<th>Energy</th>
<th>Area</th>
<th>Freq.</th>
<th>Latency</th>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium F.E.</td>
<td>2.05</td>
<td>0.85</td>
<td>2.92</td>
<td>2.38</td>
<td>12.11×</td>
</tr>
<tr>
<td>64-pt FFT</td>
<td>3.95</td>
<td>2.83</td>
<td>2.07</td>
<td>3.37</td>
<td>77.98×</td>
</tr>
</tbody>
</table>
RT Physical Design Optimization

Timing, power, and performance optimizations driven by relative timing constraints.

\[ \text{req}_{i}^\uparrow \rightarrow L_{i+1}/d + m < L_{i+1}/\text{clk}^\uparrow \]

Mapped to set_max_delay and set_min_delay constraints

Clock frequency determines min delay, async adds “hold time”
1. Inconsistency between operation and results
   - supported pins & formats, synthesis vs place and route, etc.
2. Min-delay constraints not well supported
   - Treated as “hold time fixing”
   - Create arbitrarily large delays
     - Degrades performance
     - Required matching max-delay constraint to bound delay
3. Poor job of optimizing competing constraints
4. Placement can be substantially improved
RT Physical Design Problems

Simple experiment with inverters with endpoints mapping either to module pin or library gate pin:

```
Path   Result | Iterations | Type     | Result | type
A → E  Yes    | 5          | buffers  | No     | –
A → F  Yes    | 5          | buffers  | No     | –
B → E  Yes    | 1          | Dly Elts | No     | –
B → F  Yes    | 1          | Dly Elts | Yes    | Dly Elts
C → E  Yes    | 1          | Dly Elts | No     | –
C → F  Yes    | 1          | Dly Elts | Yes    | Dly Elts
D → E  No     | –          | –        | No     | –
D → F  No     | –          | –        | No     | –
```

Paths use both max and min delay constraints
RT Physical Design Problems

Min-delay constraints get dropped, even in relatively small design!

<table>
<thead>
<tr>
<th>Model</th>
<th>Design Compiler</th>
<th>SoC</th>
<th>SoC - timing closure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#iter</td>
<td>cyc. time</td>
<td>#iter</td>
</tr>
<tr>
<td>wl0.5</td>
<td>9</td>
<td>738ps</td>
<td>1</td>
</tr>
<tr>
<td>wl0</td>
<td>7</td>
<td>666ps</td>
<td>1</td>
</tr>
</tbody>
</table>

27 March 2013 UofU and GMT
RT Physical Design Potential

1. Low hanging fruit for performance improvements
2. Force directed algorithms
   - Combine power/placement optimizations
   - Drive cell clustering
   - Drive pipeline/repeater placement and wire optimization
3. Tool performance: Convergence and run-time
Multi-Synchronous Advantage

1. Efficiency in power and performance is new game in town

2. Multi-synchronous design provides optimization opportunity

3. New (asynchronous) timing model is one excellent path

4. Produces average $10 \times e \tau^2$ improvement
   - Pentium: $e \tau^2 = 17.5 \times$
   - FFT: $e \tau^2 = 16.9 \times$

5. But ... *need improved physical design support*

<table>
<thead>
<tr>
<th>Design</th>
<th>Energy</th>
<th>Area</th>
<th>Freq.</th>
<th>Latency</th>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium F.E.</td>
<td>2.05</td>
<td>0.85</td>
<td>2.92</td>
<td>2.38</td>
<td>12.11 ×</td>
</tr>
<tr>
<td>64-pt FFT</td>
<td>3.95</td>
<td>2.83</td>
<td>2.07</td>
<td>3.37</td>
<td>77.98 ×</td>
</tr>
</tbody>
</table>