A Pausible Bisynchronous FIFO for GALS Systems

Ben Keller*†, Matthew Fojtik*, Brucek Khailany*

* NVIDIA Corporation  † University of California, Berkeley
The Nightmare of Global Clocking

- Large chips have only a handful of clock domains
- Each clock domain is many \( \text{mm}^2 \), even in \( \leq 28\text{nm} \)
  - Made up of many *partitions*

- Tapeout signoff currently requires distributing balanced synchronous clocks to every flip-flop in a clock domain...
  - ...then closing SETUP and HOLD on all paths in the clock domain
    - Across dozens of PVT corners!
Globally Asynchronous, Locally Synchronous!

From this...
Globally Asynchronous, Locally Synchronous!

...to this.
Towards Fine-Grained GALS

- No global clock! Local ring oscillator generates clock in each partition
- No global timing closure
- Reduce timing margin
  - Tracks local PVT variation
  - May improve tracking of high-frequency noise
- But there’s a catch…
Textbook Approach: Bisynchronous FIFO

Write Pointer Logic

Read Pointer Logic

FIFO Memory (Dual Port RAM)

Data In

Data Out

"Brute Force" Synchronizers

Valid

Full

Ready

Empty

TX Interface

RX Interface

May 4, 2015
The Brute Force Synchronizer

- Add latency until metastability is most likely resolved
- +3 cycles of latency at every partition boundary!
- Need to do better...

$MTBF \propto e^t$
Metastability happens when Data_Sync transitions near the clock edge.

If data arrives when clock is high, let it through.

Mutual exclusion circuit only allows G1 OR G2 to go high (never both).
Pausible Clocking Basics

Metastability happens when Data_Sync transitions near the clock edge.

If data arrives when clock is low, delay it until after the clock edge.

Mutual exclusion circuit only allows G1 OR G2 to go high (never both).

May 4, 2015
Metastability in Pausible Clocks

If the metastability lasts long enough, the next clock edge is delayed until it resolves safely.

If

Mutex Output

Data_Sync

OK Phase

Delay Phase

metastability
Clock Pauses Are Rare

Average impact on cycle time: <0.1%

99.99% of metastability events resolve in <100ps

96% of synchronizations have no metastability

May 4, 2015

A Pausible Bisynchronous FIFO for GALS Systems
Synchronization With Pausible Clock Generators

“Pausible Clock Generator”

Mutex and pausible clock safely synchronize the signal.

Synchronized signal arrives after (an average of) just 1 clock cycle!

“Demystifying Data Driven and Pausible Clocking Schemes,” Mullins07
Pausible Clocks: Our Contribution

• Prior work in pausible clocks:
  – Proposed pausible clocking as a technique for low-latency asynchronous interface crossings (Yun96)
  – Integrated pausible clocks with asynchronous FIFOs to make GALS wrappers (Moore02, Fan09, others)

• We propose a circuit that:
  – Pairs pausible clocking techniques with standard two-ported synchronous FIFOs
  – Uses a novel flow-control scheme that enables low-latency asynchronous boundary crossings
  – Integrates easily with standard toolflows
A Pausible Bisynchronous FIFO

May 4, 2015

A Pausible Bisynchronous FIFO for GALS Systems
A Pausible Bisynchronous FIFO

FIFO Memory (Dual Port RAM)

Write Pointer Logic

Pausible Clock Generator
tx_clk

Read Pointer Logic

Pausible Clock Generator
rx_clk

Data In

Data Out

Valid

Full

Empty

Ready

TX Interface

RX Interface
Pointers are synchronized via two-phase **increment** and **acknowledge** signals.
Two-phase signals

1. Valid data is written to the FIFO.
2. Write pointer increment is toggled.
3. The write pointer increment is synchronized.
4. Valid is asserted and data is read out of the FIFO.
5. Read pointer increment is toggled.
6. After the RX clock, write pointer acknowledge is toggled.
7. Write pointer acknowledge is synchronized. The increment signal can now safely be reused.
Simulation Results: Brute Force Synchronizer

Typical latency: 4 cycles

Typical latency: 1.25 cycles
Pausible Clocking Timing Constraints

Minimum clock period:
\[ \frac{T}{2} \geq t_{r2} + t_{g2} + t_{fb} \]

Maximum insertion delay:
\[ T_{ins} \leq T - t_{fb} - t_{g2} \]

Maximum same-cycle work:
\[ T_{CL} \leq t_{fb} + t_{g2} + T_{ins} \]
Pausible Clocking Limitations: Where to Put the Mutexes?

Partition

Pausible Interface

Pausible Interface

Delay due to physical distance

Clock Generator

Pausible Interface

Pausible Interface

Clock

May 4, 2015

A Pausible Bisynchronous FIFO for GALS Systems
Pausible Clocking Limitations: Where to Put the Mutexes?

Option 1: Mutexes at the interface
– Reduces maximum clock rate

Option 2: Mutexes at the center
– Increases latency
+ Improves maximum clock rate
+ Can bake the mutex into a black box with the clock generator
Playing Nice with the Tools

- Most of the design is synchronous and works fine with synthesis tools
- FIFO is a standard dual-ported memory
- A single custom cell is needed
- Custom timing constraints at clock domain crossings
### Synchronizer Comparison

<table>
<thead>
<tr>
<th>Design</th>
<th>Average Latency (cycles)</th>
<th>Area (um²)</th>
<th>Power (mW)</th>
<th>Energy (fJ/bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Interface</td>
<td>1</td>
<td>4968</td>
<td>4.08</td>
<td>25.5</td>
</tr>
<tr>
<td>Brute Force Synchronizer</td>
<td>~4</td>
<td>5005</td>
<td>6.03</td>
<td>37.7</td>
</tr>
<tr>
<td>Pausible Synchronizer</td>
<td>~1.5</td>
<td>4808</td>
<td>5.41</td>
<td>33.8</td>
</tr>
</tbody>
</table>
Summary

• Pausible clocks have key advantages over standard synchronizers, including low latency and zero probability of failure
• We designed a pausable bisynchronous FIFO that pairs pausable clocking techniques with standard two-ported synchronous FIFOs
• Fast asynchronous interfaces that work well with standard toolflows are a key enabling technology of fine-grained GALS design
References


Questions?
Backup Slides
Mutex Circuit

- Only G1 or G2 can be high at once (never both)
- If R1 and R2 transition high simultaneously, metastability can result
- The “metastability filter” keeps both outputs low until metastability resolves
Simulation Setup

- Implemented interface in Verilog
- Allows direct comparison to brute-force synchronizers via drop-in replacement
Minimum Clock Period

Minimum clock period:

\[ \frac{T}{2} \geq t_{r2} + t_{g2} + t_{fb} \]
Requests can arrive near the clock edge!

Maximum insertion delay:

\[ T_{ins} \leq T - t_{fb} - t_{g2} \]
Combinational Logic Delay $T_{CL}$

Maximum same-cycle work:

$$T_{CL} \leq t_{fb} + t_{g2} + T_{ins}$$
Metastability Margin $t_m$

Minimum clock period:

$$\frac{T}{2} \geq t_{r2} + t_{g2} + t_{fb}$$

Metastability margin:

$$t_m = \frac{T}{2} - (t_{r2} + t_{g2} + t_{fb})$$
Average Latency

Average latency:

\[ T_L = T + T_{\text{ins}} - t_{r2} \]