Interleaved Architectures for High-Throughput Synthesizable Synchronization FIFOs

Ameer Abdelhadi  
McGill University  
ameer.abdelhadi@gmail.com

Mark Greenstreet  
University of British Columbia  
mrg@cs.ubc.ca

May 22, 2017

• Synchronization FIFOs
• The Interleaved FIFO Architecture
• Design Flow
• Hazards
• Results
Multi-Synchronous Designs

Typical chip designs consist of

- A large number of synchronous, synthesized blocks – **this is the easy part!**
- There are many different clocks.
- Designers need easy to use, synthesizable interfaces.

The big challenges are ones of design integration.

- **This is where asynchronous methods excel.**

FIFOs are a commonly used for these interfaces.
Prior Work: Gray-Code FIFOs

- Textbook solution – familiar to many designs.
- Gray-to-binary conversion a bottleneck.
- See [Cummings+Alfke2002].
Prior Work: One-Hot FIFOs

+ Simple, fast control – very modular design.
− Per-stage synchronizers and control tends to dominate area and power.

• See [Chelcea+Nowick2004,Ono+Greenstreet2009].
Contributions

- A novel, interleaved FIFO architecture
  - Area and power efficient
  - High throughput, minimal latency.
- Synthesizable design
  - Open-source Verilog
    - Supports standard, Synopsis™ design flow.
    - You can use it in your designs!
  - Highly parameterized – synthesize the FIFO you want.
  - Provides benchmark design for comparisons for further research.
- Identify a glitch hazard in synchronization FIFOs
  - Many published designs have this hazard.
  - We present our solution.
FIFO Architecture

Similar to Gray-Code FIFOs

- Separate write and read pointers in put and get interfaces.
- Our design uses a pair of one-hot counters in each interface.
  - Avoids conversions to/from Gray code ⇒ simple and fast
  - Has a small number of synchronizers between put and get ⇒ area and power efficient.

Simple interface to sender and receiver

- Looks like flip-flop (in each domain) with standard flow control.
- Minimal exposure of internal timing details.
A Thermometer Counter

- Initialized to all zeros.
- Fills left-to-right with ones, then fills with zeros, then with ones, . . .
- Current position marked by state with transition (or 0 if all stages are the same).
Dual Thermometer Counters

Increment the horizontal counter with each wrap-around of the vertical counter.

Sequence length is the product of those for the two counters.

Avoids the $O(N)$ control-complexity that is the bane of one-hot designs.
Decode to produce full-thermometer code
Decode to produce full-thermometer code
Decode to produce full-thermometer code
Decode to produce full-thermometer code
Decode to produce full-thermometer code
Decode to produce full-thermometer code
Decode to produce full-thermometer code
Detecting Data Availability

A stage holds valid data if the value of the get-thermometer differs from the value of the put-thermometer.

An OR-tree can determine if any stage holds valid data.

But we need to synchronize.
Synchronization with Interleaving

- One synchronizer per row.
- $N_{\text{vertical}} \geq \text{Latency}_{\text{sync}}$ required for full throughput.
- Can use fewer synchronizers than Gray-Code designs – if the synchronizer latency is less than the number of address bits.
data_available is registered, we need to know if there will be data available after the next clk_get↑.

- If any row has a valid value and there isn’t a req_get on this cycle, then data will be available on the next cycle.
- If two consecutive rows have valid data (one even, the other odd), then data will be available on the next cycle.

We clear the synchronizer after removing a data word from that row.
data_available is registered, we need to know if there will be data available after the next clk_get↑.

We clear the synchronizer after removing a data word from that row.

- If there is data available in another column, we allow the first stage of the synchronizer to record that on clk_get↑.
The Data Path

- Input latch gives FIFO the same set-up and hold timing as a flip-flop.
- Output path uses even/odd interleaving:
  - We found that the data output path could not maintain GHz clock frequencies.
  - Key observation: data is available on the output of the data latches for nearly the full synchronizer latency. This ensures data validity.
  - Alternating paths provides an extra clock-cycle for the path to settle, and the timing requirements are easily satisfied.
- The output logic is to block glitches (see next slide).
A designer might (reasonably) assume that

```
assign foo = data_valid ? f(data_out, ...) : g(...);
```

is safe.
FIFOs and CDC Hazards

What synthesis can do

- A designer might (reasonably) assume that
  
  ```
  assign foo = data_valid ? f(data_out, ...) : g(...);
  ```
  
is safe.
- **BUT**, synthesis can introduce glitches the designer never imagined.
FIFOs and CDC Hazards

What synthesis can do

- A designer might (reasonably) assume that
  
  \[
  \text{assign foo} = \text{data_valid} \ ? \ f(\text{data_out}, \ldots) \ : \ g(\ldots); \]
  
  is safe.

- **BUT,** synthesis can introduce glitches the designer never imagined.
A designer might (reasonably) assume that
\[
\text{assign foo = data_valid ? f(data_out, ...) : g(...);
}\]
is safe.

**BUT**, synthesis can introduce glitches the designer never imagined.
A designer might (reasonably) assume that
\[
\text{assign foo = data_valid ? f(data_out, ...) : g(...);} \text{ is safe.}
\]

BUT, synthesis can introduce glitches the designer never imagined.

This hazard is present in many published clock-domain-crossing (CDC) FIFOs.

Our design ensures \textit{data_out} is all 0s when no valid data is available.
**Complete Synopsis™ design flow**
- Includes synthesis, place-and-route, back-annotation, simulation, performance analysis, and power estimation

**Highly parameterized**
- FIFO word-size, depth, and interleaving specified by user.
- Number of synchronizer stages
- Target clock frequency

**Open source**
- The FIFO itself is about 200 lines of Verilog
- The test bench is about 300 lines of Verilog
- Plus about 2400 lines for 10 scripts.
- You can use it, you can build on it!
- Get it from Github – details on the last slide.
### Performance

<table>
<thead>
<tr>
<th>$N$</th>
<th>$N_h$</th>
<th>$N_v$</th>
<th>$w$</th>
<th>$S$</th>
<th>freq</th>
<th>power</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>1300</td>
<td>1.2 mw</td>
<td>1745 $\mu^2$</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>1400</td>
<td>1.3 mw</td>
<td>2618 $\mu^2$</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>1500</td>
<td>1.9 mw</td>
<td>3268 $\mu^2$</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>1200</td>
<td>1.8 mw</td>
<td>4875 $\mu^2$</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>2</td>
<td>32</td>
<td>2</td>
<td>1400</td>
<td>2.2 mw</td>
<td>2979 $\mu^2$</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>4</td>
<td>32</td>
<td>2</td>
<td>1200</td>
<td>2.0 mw</td>
<td>6725 $\mu^2$</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>4</td>
<td>32</td>
<td>4</td>
<td>1100</td>
<td>2.9 mw</td>
<td>12712 $\mu^2$</td>
</tr>
</tbody>
</table>

A few examples from 144 test cases.

- Frequency dominated by “tree structures”. Clock period is roughly logarithmic in $N_h$, $N_v$, and $w$.
- Power is dominated by the control circuitry and is roughly linear in the number of control flip-flops.
- Area is dominated by data storage and is roughly linear in the number of data latches.
- Results using a 65nm commercial library.
Conclusions

- We have presented a fully synthesizable, general purpose, clock-domain crossing FIFO.
  - open source: [https://github.com/AmerAbdelhadi/Interleaved-Synthesizable-Synchronization-FIFOs](https://github.com/AmerAbdelhadi/Interleaved-Synthesizable-Synchronization-FIFOs)
  - Supports complete Synopsis ASIC design flow.
  - Highly configurable.
  - Thoroughly evaluated for a wide range of configuration parameters.
- Novel interleaved FIFO architecture
  - avoids the Gray-code conversion bottlenecks of Gray-code FIFOs.
  - smaller control logic and far fewer synchronizers than one-hot FIFOs.
- Identified a glitch hazard that is common in published designs.
  - and our design avoids it.
- Thanks: Tarik Ono, Brad Quinton, NSERC Canada.