Latch-Based Performance Optimizations For FPGAs

Bill Teng and Jason Anderson
Motivation & Introduction

• Programmability of FPGAs comes with a performance penalty
• FPGAs are 3-4x slower than standard cell implementations [1]
• **Need** better performance to close the gap between FPGAs with ASICs

Motivation & Introduction

• Clock skew and latch-based optimizations are used to improve the performance of today’s CPUs

• Xilinx FPGAs can configure any edge-triggered FF to be a latch, and can generate clocks with different phases and duty cycles

• **Goal of this work:** Improve performance of FPGAs without power or hardware penalty!
Outline

• Sequential Circuits: Timing Analysis
• Clock skew
• Latch Timing Analysis
• Pulsed Latches
  – Advantages
  – Our Approach
• Methodology & Results
• Ongoing & Future Work
Sequential Circuits: Timing Analysis

• Longest Path between any pair of FFs determines the clock period

Clock Period: 8 ns
Clock Skew

• Delaying the clock can give the critical path more time to complete
• Also known as time borrowing or cycle stealing

Clock Period: 6 ns
Clock Skew

 clk

FF1

FF2

1  8 ns  2  4 ns  3

2 ns

6 ns

2 ns
Multiple Skewed Clocks

- Clock skew set automatically picked to optimize performance
- ~20% gain with 4 clock lines
- Extra clock lines => power overhead

Programmable Delay Elements

• reduces glitch power and improves circuit performance via clock skew
• ~15% improvement in circuit speed
• Hardware overhead

Transparent Latch Basics

Q tracks D when clk is active
Latch Timing Analysis

- F1: Min = 3 ns, Max = 8 ns
- L2: Min = 3 ns, Max = 4 ns
- F3: 

Clock cycle: 6 ns

Signals:
- FF1
- L2
- FF3
Pulsed Latches

- FF1
  - Min = 3 ns
  - Max = 8 ns
- L2
  - Min = 3 ns
  - Max = 4 ns
- F3

Clk
- 6 ns
- 2 ns

FF1

L2

FF3
Latches versus Skewed Clocks

- Latches “mimic” the effect of having multiple skewed clocks
Pulsed Latches in FPGAs

• Modern FPGAs already can implement circuits with pulsed latches:
  – Storage elements can be edge-triggered or latch
  – Clock management circuitry can synthesize clocks with different duty cycles
Our Approach: Basic Idea

• **Input:** routed design with flip-flops
• **Output:** performance-optimized design with latches
• Automatically transform certain flip-flops to be latches
• Choose a duty cycle for the latches to avoid hold-time violations.
• Current work: single clock clock designs.
Optimal Clock Period
Optimal Clock Period

Clock Period: \( \frac{10 + 3 + 5}{3} = 6\text{ns} \)
Handling Hold-Time Violations

Pulse Width: 1ns
Handling Hold-Time Violations

Pulse Width: 2ns
Handling Hold-Time Violations

Diagram with nodes and edges labeled with numbers. Arrows indicate the direction of data flow. Diagram includes a pulse width of 3 ns.
Handling Hold-Time Violations

Clock Period: 7

Pulse Width: 3ns
FFs: A, C
Latches: D
Overall Algorithm

Compute optimal clock period without hold-times
Compute all hold-time register-to-register paths
Sort paths in ascending order
Iterate over paths until
Overall Algorithm

Compute optimal clock period without hold-times
Compute all hold-time register-to-register paths
Sort paths in ascending order
Iterate over paths until
Set pulse width = 
Force elements with paths < pulse width to be FFs
Re-calculate clock period
Methodology

• Work done in VPR 5.0
• Performance optimization framework built using C++ BOOST library.
• 27 Circuits
  – All sequential MCNC circuits.
  – VPR 5.0 circuits.
• Optimization performed after place and route
Minimum Delay Assumptions

• VPR doesn’t model minimum delays
  – Only models maximum (worst-case) delays

• Assume minimum delays are a fraction (%) of computed short path delays

• Consider three different scenarios:
  – 60% (pessimistic) [more hold violations]
  – 70% (nominal)
  – 80% (optimistic) [less hold violations]
Results

<table>
<thead>
<tr>
<th>Condition</th>
<th>Clock Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical Path</td>
<td>1</td>
</tr>
<tr>
<td>Optimal (None)</td>
<td>0.71</td>
</tr>
<tr>
<td>Pulsed Latch (70%)</td>
<td>0.949</td>
</tr>
<tr>
<td>Clock Skew (70%)</td>
<td>0.903</td>
</tr>
<tr>
<td>Pulsed Latch (60%)</td>
<td>0.976</td>
</tr>
<tr>
<td>Clock Skew (60%)</td>
<td>0.907</td>
</tr>
</tbody>
</table>
What limits our gains?

• Self-loops
  ▪ Paths that start and end at the same latch
  ▪ Only affects latches

• Extremely short intra-CLB paths
  ▪ Constant delay

• Fully-connected CLB architecture
Ongoing & Future Work

• Current work: alter router make some connections “longer” (more delay):
  – Bring intra-CLB connections outside
  – Take more circuitous routes

• Integrate into placement, routing and clustering:
  – Make place, route and clustering aware of this optimization in their criticality metrics
  – Optimize *critical cycle* instead of critical path
Summary

• Modern FPGAs can already implement pulsed latched circuits.
• Much of the benefit of multiple skewed clocks can be achieved with *single* clock + pulsed latches.
• Under 70% minimum delay assumptions: 5% speed improvement can be achieved for *free*
Questions?

• More details:
Potential for Delay Padding

![Bar chart showing the potential for delay padding with clock periods of different percentages of connections fixed.](chart.png)

- **None**: 1.005
- **3%**: 0.999
- **5%**: 0.988
- **10%**: 0.966

Clock Period

% of Connections Fixed