Application of Specific Delay Window Routing for Timing Optimization in FPGA Designs

Evan Wegley, Qinhai Zhang
Lattice Semiconductor Corporation

23rd ACM / SIGDA International Symposium on FPGAs
FPGA Routing

- FPGA routers must optimize for competing goals
  - Routability
  - Timing performance

- Timing constraints come in many forms
  - Setup timing
  - Hold timing
  - Other timing constraints
    - Maximum skew, for example
Setup Timing

- Constrains data paths between registers to arrive **before** the next clock cycle starts
  - Produces an **upper** bound on delay of the data path

\[
0.1 \text{ ns} \leq \text{Delay} \leq 3.0 \text{ ns}
\]
Hold Timing

- Constrains data paths between registers to arrive after the previous cycle has been captured
  - Produces a lower bound on delay of the data path
  - At routing phase, we can correct hold timing violations by rerouting the data path to have additional delay

\[ 0.1 \text{ ns} \leq \text{Delay} \leq 3.0 \text{ ns} \]
More Complex Hold Timing

- Example: constrain the data paths to 2.0 ns for maximum setup time and 0.4 ns for minimum hold time with respect to clock.
More Complex Hold Timing

- Example: constrain the data paths to **2.0 ns** for maximum setup time and **0.4 ns** for minimum hold time with respect to clock.

\[
\text{Slack}_{A, \text{Hold}} = (1.0 + 0.3 - 1.2) - 0.4 = -0.3 \text{ ns}
\]

Hold violation!

Note: ignoring LUT logic delays
More Complex Hold Timing

- Example: constrain the data paths to **2.0 ns** for maximum setup time and **0.4 ns** for minimum hold time with respect to clock

\[
\text{Slack}_{\text{A, Hold}} = (1.0 + 0.3 - 1.2) - 0.4 = -0.3 \text{ ns}
\]

Hold violation!

\[
\text{Slack}_{\text{B, Setup}} = 2.0 - (1.0 + 2.1 - 1.2) = 0.1 \text{ ns}
\]

Close to setup threshold

Note: ignoring LUT logic delays
More Complex Hold Timing

- Need awareness of both setup and hold requirements for each connection
- We can do so by setting lower and upper bounds on delay for each connection: slack allocation

![Diagram showing connections and delays](image-url)
Slack Allocation

- Process of distributing slack to all connections of a circuit
Slack Allocation

• Process of distributing slack to all connections of a circuit (Youssef 1990, Frankle 1992, Fung 2008)
  • Allocate slack on setup timing to get upper bound

Note: ignoring LUT logic delays
Slack Allocation

  - Allocate slack on **setup** timing to get upper bound
  - Allocate slack on **hold** timing to get lower bound

![Diagram showing slack allocation process](image)

Note: ignoring LUT logic delays
Slack Allocation

  - Allocate slack on *setup* timing to get upper bound
  - Allocate slack on *hold* timing to get lower bound

Note: ignoring LUT logic delays
Maximum Skew

- Constrains the range of delays on the loads of some net(s) or buses
- Example: constrain the net to have a maximum skew of 0.5 ns
Maximum Skew

• Constrains the range of delays on the loads of some net(s) or buses
• Example: constrain the net to have a maximum skew of 0.5 ns

Initial skew is $1.0 - 0.2 = 0.8$ ns
• Violates constraint of 0.5 ns maximum skew by 0.3 ns
Maximum Skew

- Setting delay bounds on each connection
  - Step 1: Use the largest delay value as the upper bound

![Graph showing delay bounds]
Maximum Skew

- Setting delay bounds on each connection
  - Step 1: Use the largest delay value as the upper bound
  - Step 2: Find the lower bound by subtracting the constraint value

![Diagram showing delay bounds and points (s, t₁), (s, t₂), (s, t₃) with a shaded area representing the delay range [0.50, 1.00].]
Maximum Skew

• Setting delay bounds on each connection
  • Step 1: Use the largest delay value as the upper bound
  • Step 2: Find the lower bound by subtracting the constraint value
  • Step 3: Reroute any connections falling outside the bounds
Specific Delay Window Routing

- Definition: routing a connection with a target delay between some lower and upper bounds
  - Lower and upper delay bounds form a “window”

- Allows us to optimize for various timing constraints constituting both lower and upper bounds on delay
Current FPGA Routing Technology

- **PathFinder based (McMurchie 1995)**
  - Basis of VPR router (Betz 1997) and many other academic and commercial FPGA routers
  - Effective at balancing routability and timing performance

\[ \text{Cost}_n = A_{ij} d_n + (1 - A_{ij}) c_n \]

- Delay cost is the total delay of the connection
- Traditional single-wave search: total delay contains estimation
Single-Wave Expansion

- One approach to performing Specific Delay Window Routing
  - Single-wave expansion using delay estimation to direct search towards the target delay window

\[ \text{Total Delay} = \text{Known Delay} + \text{Estimated Delay} \]
Single-Wave Expansion

- One approach to performing Specific Delay Window Routing
  - Single-wave expansion using delay estimation to direct search towards the target delay window

Total Delay = Known Delay + Estimated Delay
Single-Wave Expansion

- One approach to performing Specific Delay Window Routing
  - Single-wave expansion using delay estimation to direct search towards the target delay window

\[
\text{Total Delay} = \text{Known Delay} + \text{Estimated Delay}
\]
Accuracy Issues

• Estimation can be inaccurate

  • Sparse crossbar
  • Manhattan distance: 4
  • Estimate: 2 “X2” wires
  • Switch between “X2” wires is missing!
Accuracy Issues

- Estimation can be inaccurate
  - Sparse crossbar
  - Swappable pins
    - LUT input pins have different delays
    - Pins are logically equivalent
    - Actual target pin not known during estimation
Accuracy Issues

- Estimation can be inaccurate
  - Sparse crossbar
  - Swappable pins
  - Congestion adds variability
Accuracy Issues

- Estimation can be inaccurate
  - Sparse crossbar
  - Swappables pins
  - Congestion adds variability
Accuracy Issues

- Estimation can be inaccurate
  - Sparse crossbar
  - Swappable pins
  - Congestion adds variability
Dual-Wave Expansion

• To address these accuracy issues, we propose to use dual-wave search
  • Instead of directing one search towards the target, we can expand from both the source and target
  • Each time the waves intersect, we check if the resulting path meets the target delay window
  • This eliminates estimation from the selection process
Dual-Wave Expansion

Wave from both source and target
Total Delay = **Known Delay** + **Known Delay**

**Intersection:**
Total delay does not meet target window
Dual-Wave Expansion

Intersection:
Total delay does meet target window

Total Delay = Known Delay + Known Delay
Routing Flow

- Global Routing
  - Clock routing, other architecture-specific routing
- Detail Routing
  - PathFinder-based

Specific Delay Window Routing

- Skew Optimization
  - Calculate delay windows for constrained connections
  - Perform specific delay window routing on connections in violation of skew constraint
- Hold Timing Optimization
  - Calculate delay windows using slack allocation
  - Perform specific delay window routing on connections in violation of hold timing
Experimental Methodology

- Compared two generations of commercial tools
  - Older one uses single-wave expansion with delay estimation
  - Newer one uses dual-wave expansion without delay estimation
- Ten designs each tested for hold timing and skew correction
  - Customer designs with known violations
  - Designs with strict skew constraints

<table>
<thead>
<tr>
<th>Design</th>
<th>Device</th>
<th>Family</th>
<th>LUT4s</th>
<th>Utilization</th>
<th>Device</th>
<th>Family</th>
<th>LUT4s</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>N0</td>
<td>MachXO2</td>
<td>4K</td>
<td>70%</td>
<td>H0</td>
<td>LatticeEC</td>
<td>33K</td>
<td>9%</td>
<td></td>
</tr>
<tr>
<td>N1</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>11%</td>
<td>H1</td>
<td>LatticeEC</td>
<td>33K</td>
<td>9%</td>
<td></td>
</tr>
<tr>
<td>N2</td>
<td>LatticeECP2</td>
<td>20K</td>
<td>45%</td>
<td>H2</td>
<td>MachXO</td>
<td>2K</td>
<td>85%</td>
<td></td>
</tr>
<tr>
<td>N3</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>66%</td>
<td>H3</td>
<td>MachXO</td>
<td>2K</td>
<td>85%</td>
<td></td>
</tr>
<tr>
<td>N4</td>
<td>ECP5</td>
<td>85K</td>
<td>67%</td>
<td>H4</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>83%</td>
<td></td>
</tr>
<tr>
<td>B0</td>
<td>MachXO2</td>
<td>2K</td>
<td>67%</td>
<td>H5</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>14%</td>
<td></td>
</tr>
<tr>
<td>B1</td>
<td>LatticeECP2</td>
<td>20K</td>
<td>74%</td>
<td>H6</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>82%</td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>LatticeECP3</td>
<td>70K</td>
<td>41%</td>
<td>H7</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>87%</td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>28%</td>
<td>H8</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>76%</td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>ECP5</td>
<td>45K</td>
<td>23%</td>
<td>H9</td>
<td>LatticeECP3</td>
<td>150K</td>
<td>84%</td>
<td></td>
</tr>
</tbody>
</table>
Experimental Results: Skew Correction

- Varied the skew constraint values from 4.0 ns (loose) to 0.0625 ns (very strict) and compared ability to find a solution.
Experimental Results: Skew Correction

- Runtime comparison between dual-wave and single-wave approaches

**Graph Description:**
- **X-axis:** Constraint Value (ns)
- **Y-axis:** Runtime Ratio (dual-wave runtime / single-wave runtime)
- **Graph Elements:**
  - Individual Designs
  - Mean
Experimental Results: Hold Correction

- Compared ability to solve hold timing violations in 10 designs
  - Single-wave approach corrected only 5 of 10
  - Dual-wave approach corrected all 10 of 10
Experimental Results: Hold Correction

- Runtime not significantly increased using dual-wave approach
  - Older tool generation used heuristic method for setup-awareness
  - Newer tool generation uses slack allocation, which uses more time

![Runtime Ratio Graph]

<table>
<thead>
<tr>
<th>H0</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>H4</th>
<th>H5</th>
<th>H6</th>
<th>H7</th>
<th>H8</th>
<th>H9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dual-wave runtime / single-wave runtime
Conclusion

- Presented specific delay window routing as a generic framework to optimize for constraints constituting lower and upper bounds on delay

- Proposed dual-wave expansion to address accuracy issues with single-wave expansion when targeting a specific delay window

- Future work
  - Applying specific delay window routing to more constraints
    - Clock-to-output
    - Cycle stealing
Multiple Speed Grades

• We consider multiple speed grades for slack allocation
  • For setup timing, we use the design speed grade
  • For hold timing, we use the fastest speed grade

• Inverted delay windows
  • If the lower bound exceeds the upper bound, then we cannot satisfy the setup and hold requirements for the connection
Slack Allocation

LOOP {

1. Update timing

2. Compute slack on each connection

   \[ \text{slack}(c) = \min(\text{slack}(p)) \]

   where \( p \) is any path containing \( c \)

   \textbf{Stop if the cumulative slack is near zero}

3. Compute the weight for each connection

   \[ \text{weight}(c) = \frac{\text{delay}(c)}{\max(\text{delay}(p))} \]

   where \( p \) is any path containing \( c \)

4. Allocate slack for each connection

   \[ \text{delay}(c) = \text{delay}(c) + \text{weight}(c) \times \text{slack}(c) \]

}
Slack Allocation

Slack allocation on setup timing for the circled connection

![Diagram showing setup slack and connection delay](image)
Results Analysis

- Why does design “B0” take so much longer with dual-wave expansion?
  - Both single-wave and dual-wave approaches fail for this design with a 0.125 ns constraint

- Architectural corner case involving three connections
  - Stuck in a scenario where 2 have equal skew, but another is faster
Results Analysis

• Why does design “B0” take so much longer with dual-wave expansion?
  • Both single-wave and dual-wave approaches fail for this design with a 0.125 ns constraint

• Architectural corner case involving three connections
  • Stuck in a scenario where 2 have equal skew, but another is faster
Results Analysis

• Why does design “B0” take so much longer with dual-wave expansion?
  • Both single-wave and dual-wave approaches fail for this design with a 0.125 ns constraint

• Architectural corner case involving three connections
  • Stuck in a scenario where 2 have equal skew, but another is faster

![Diagram showing connections: S to t1, S to t2, S to t3 (t3 is too fast)]
Results Analysis

• Why does design “B0” take so much longer with dual-wave expansion?
  • Both single-wave and dual-wave approaches fail for this design with a 0.125 ns constraint

• Architectural corner case involving three connections
  • Stuck in a scenario where 2 have equal skew, but another is faster
Future Work

• Clock-to-output constraint
  • Relative timing constraint between paths
  • Example: $0.5 \text{ ns} < \text{clock-to-out path} - \text{clock-out path} < 4 \text{ ns}
Future Work

• Cycle stealing
  • Add extra delay to clock path to alleviate setup violations
Future Work

- Cycle stealing
  - Add extra delay to clock path to alleviate setup violations