Technology Mapping of Timed Asynchronous Circuits

Curtis A. Nelson
University of Utah
August 26, 2004

Technology Mapping

- Process of implementing a synthesized design.
- Utilizes technology-specific libraries.
- Combines the steps of:
  - Partitioning
  - Decomposition
  - Matching
  - Covering
Timed Asynchronous Circuits

- Timed circuits are a class of asynchronous circuits that use explicit timing information.
- Can potentially reduce required circuitry as compared with speed-independent circuits.
- Used in Intel RAPPID design which was 3 times faster than synchronous design.
- As in all asynchronous circuits, timed circuit design is complicated by hazards.

Hazards

- Conditions that may manifest as glitches.
- Caused by structure or timing of the circuit.
- May result in incorrect behavior.
- Must be detected and eliminated.
- Two types of hazards:
  - Acknowledgment.
  - Monotonicity.
Acknowledgement Hazard

Occurs when inputs to a gate change evaluation before the output stabilizes.

Acknowledgement Hazard

Occurs when inputs to a gate change evaluation before the output stabilizes.
Acknowledgement Hazard

- Occurs when inputs to a gate change evaluation before the output stabilizes.

Monotonicity Hazard

- Occurs when an internal or output node:
  - Becomes excited to change when it should stay stable, or
  - Makes a transition non-monotonically.
Monotonicity Hazard

- Occurs when an internal or output node:
  - Becomes excited to change when it should stay stable, or
  - Makes a transition non-monotonically.

Asynchronous Design Flow

- Specifications
- Logic Synthesis
- Partitioning
- Decomposition
- Matching/Covering
- Physical Design
- Layout
- Library
- Hazard Verification
- Technology Mapping
- Cost Factors
Hazard Verification

- Must check all reachable states for hazards.
- Size of state space is $O(2^{|I|} \times 2^{|O|} \times 2^{|N|})$.
- Beerel/Burch/Meng proposed using a *cube* to approximate internal signal behavior for speed-independent circuits.
- Reduces size of state space to $O(2^{|I|} \times 2^{|O|})$.
- We extend this application to timed circuits.

Hazard Verification Algorithm

- Input: *Time Petri Net* and *Gate Netlist*.
  - Check *complex gate equivalence*.
  - Determine stability of internal signals.
  - Check for acknowledgement hazards.
  - Check for monotonicity hazards.
Time Petri Net

- Specification method for timed systems.
- Used to specify:
  - Environmental behavior.
  - Expected output behavior.

Transition enabled when all input places are marked.
Transition fires after it has been enabled between [min, max] time units.
Time Petri Net

After transition fires, marking removed from input places and added to output places.

Gate Netlist

Gate-Level Netlist
Gate Netlist

Gate-Level Netlist

Complex Gate Equivalent (CGE)

Checking Equivalence

0000

abcd
Checking Equivalence

a+ [2,5] b+ [2,5]
c [2,5]
d+
b- [2,5]
c- [2,5]
a- [2,5]
d-

Checking Equivalence

0000 a+

1000 b+

+1100 c+

d+

a+ [2,5] b+ [2,5]
c [2,5]
d+
b- [2,5]
c- [2,5]
a- [2,5]
d-

Checking Equivalence

a+ [2,5] b+ [2,5]
c [2,5]
d+
b- [2,5]
c- [2,5]
a- [2,5]
d-

Checking Equivalence

a+ [2,5] b+ [2,5]
c [2,5]
d+
b- [2,5]
c- [2,5]
a- [2,5]
d-

Checking Equivalence

a+ [2,5] b+ [2,5]
c [2,5]
d+
b- [2,5]
c- [2,5]
a- [2,5]
d-
Checking Equivalence
Checking Equivalence

Checking Equivalence
Checking Equivalence

Checking Equivalence
Checking Equivalence

Checking Equivalence
Checking Equivalence
Checking Equivalence

Finding Stable States
Finding Stable States

Step 1: Boolean evaluation of node e.

Step 2: Untimed stabilization of node e for d+
Finding Stable States

Step 2: Untimed stabilization of node e for d+

Step 2: State 1101 stable high.
Finding Stable States

Step 3: Untimed stabilization of node e for d-

Rising
High
Falling
Low
Finding Stable States

Step 4: Propagate stable states.

Rising
High
Falling
Low

abcd
Checking for Hazards

Node e is hazard-free!

Alternative Gate-Level Netlist

Try a different decomposition – Replace:

With:
Finding Stable States

Step 1: Boolean evaluation of node e.

Rising
High
Falling
Low

Step 2: Untimed stabilization of node e for d+

Rising
High
Falling
Low
Finding Stable States

Step 2: Untimed stabilization of node e for d+

Finding Stable States

Step 3: Propagate stable states.
Finding Stable States

Step 3: Propagate stable states.

Step 4: Untimed stabilization of node e for d−.
ACKNOWLEDGEMENT HAZARD!

Node e may glitch.

MONOTONICITY HAZARD!

Output d may glitch.
Consider Netlist timing.

Minimum elapsed time: 2
Timed Stabilization

Minimum elapsed time: 4
Node e has stabilized low

Node e is now hazard-free!

Propagate stable states.
Asynchronous Design Flow

Partitioning

- Each output is a tree-based structure which is decomposed, matched, and covered individually.
- Makes matching and covering tractable.
- May reduce quality of final netlist because gate-sharing is eliminated between outputs.
- For our work, synthesis provides a set of equations already partitioned by primary output.
Decomposition: Background

- Decomposition performed to guarantee a solution.
- Changing the circuit structure may create hazards.
- Synchronous approach:
  - Decompose synthesized circuit to base functions, typically inverters, 2-input NANDs, and memory elements.
- Asynchronous approaches are to:
  - Decompose directly to library gates without creating hazards (Burns, Beerel, Siegel, Kondratyev, Myers, etc.).
  - Decompose without regard to hazard creation and remove hazards by inserting delay elements (Lavagno).

Decomposition: Our Approach

- Follow synchronous approach, decompose into base functions (inverters, 2-input NANDs, and C-element's).
  - Decompose without regard to hazard creation.
  - Perform gate-level hazard verification.
  - Do hazard-guided matching/covering later.
- Insert inverter pairs to increase matching options.
- Use unbalanced structure to allow for:
  - Consistent architecture.
  - Input pin re-ordering.
State-Holding Devices

- **C-elements (CEL's)**

![C-elements Diagram]

- **Generalized C-elements (gC's)**

![Generalized C-elements Diagram]
Decomposition Architectures

CGE Netlist

Balanced

Decomposition Architectures

CGE Netlist

Balanced

Unbalanced
Decomposition Architectures

Balanced

Unbalanced with inverter pairs

Input Pin Re-ordering

- Identify trigger, context signals.
- Place wisely in decomposition.

Untimed Hazardous.

Timed Hazard-free.
Input Pin Re-ordering

- Identify trigger, context signals.
- Place wisely in decomposition.

Untimed Hazardous.
Timed Hazard-free.

Untimed Hazard-free.
Timed Hazard-free.

Asynchronous Design Flow

Technology Mapping

Specifications
Logic Synthesis
Partitioning
Decomposition
Cost Factors

Hazard Verification
Matching/Covering
Physical Design
Layout

Library
Matching and Covering

- Structurally matches library elements to circuit decomposition.
- Typically guided by area, delay, or power.
- Ours must be guided by hazard-freedom.
- Proposed hazard-aware matching algorithm:
  - Encapsulates acknowledgment hazards.
  - Finds forcing side-inputs for monotonicity hazards.

Acknowledgement Hazard

- Occurs when inputs to a gate change evaluation before the output stabilizes.
- Node must be encapsulated.
Acknowledgement Hazard

- Occurs when inputs to a gate change evaluation before the output stabilizes.
- Node must be encapsulated.

Monotonicity Hazard

- Occurs when an internal or output node becomes excited to change when it should stay stable.
- Prevented by forcing side inputs.
Monotonicity Hazard

- Occurs when an internal or output node becomes excited to change when it should stay stable.
- Prevented by forcing side inputs.
- Fanin node causing hazard must not be encapsulated.
Hazard-Aware Matching

- Each match must undergo cost analysis.
- Objective is to minimize cost.
- Heuristic equation to determine hazard cost:

\[
hazcost = (W_1)_{\text{monohaz}} + (W_2)_{\text{ackhaz}}
\]

- \(W_1, W_2\) are coefficients.
- \(\text{monohaz}\) is the number of exposed monotonicity hazards.
- \(\text{ackhaz}\) is the number of exposed acknowledgment hazards.

- \(W_1, W_2 = 0\): No hazard awareness.
- \(W_1, W_2 = 1\): Encourage encapsulation.
- \(W_1, W_2 = -1\): Discourage encapsulation.
- \(W_1, W_2 = -1, 1\): Discourage monotonicity encapsulation, encourage acknowledgment encapsulation.

Hazard-Aware Matching Example
Hazard-Aware Matching Example

- $W_1, W_2 = 0$: No hazard awareness.
- $W_1, W_2 = -1$: Discourage encapsulation.
- $W_1, W_2 = 1$: Encourage encapsulation.

* Untimed synthesis, verification.

Hazard-Aware Matching Example

- $W_1, W_2 = -1,1$: Optimum.
- $W_1, W_2 = 0$: No hazard awareness.
- $W_1, W_2 = 1$: Encourage encapsulation.

1 Ack hazard.
Mono hazards on req caused by $y_0, y_1$.

2 Ack hazards.
Mono hazard on req caused by node M.
Short-Circuit Issues

- May occur when a gC is mapped to a portion of decomposition.

Example: Map a plain gC element to a netlist.
Short-Circuit Issues

- Potential short-circuit in state 00101.⇒ reject gC and replace with CEL
Short-Circuit: Example 2

State = adbcx

Reject gC.
Common-Input Matching

- Must be addressed when a library cell has a common-input on two or more leaves.
- Requires leaves to be driven by equivalent sub-networks.
- Prohibits short-circuit problems if the common-input is found in both set and reset networks of a gC.
- Can increase size of library dramatically.

Common-Inputs: Example 1

- XOR: Non-tree based

\[ u = a \overline{b} + b \overline{a} \]
Common-Inputs: Example 1

- XOR: Non-tree based
  \[ u = \overline{ab} + \overline{ba} \]
- XOR: Tree based
Common-Inputs: Example

- $gC22$ with 1 common-input ($p$)
Common-Inputs: Example

- gC22 with 1 common-input (p)

Experimental Results

- Algorithms implemented in 11,545 lines of C/C++ code within the ATACS software tool.
- Tests results presented for:
  - Gate-level verification.
  - Matching and covering.
    - Timed synthesis.
    - Untimed synthesis.
    - Hazard-aware matching.
Gate-level Verification

- Compared our verifier with:
  - KRONOS – timed automata tool, only checks conformance, not hazards.
  - Pena’s tools – conservative approximation method, stops after finding first hazard.
  - ATACS – explicit state timing verifier.

<table>
<thead>
<tr>
<th>Example</th>
<th>Gates</th>
<th>Kronos CPU Time</th>
<th>Pena CPU Time</th>
<th>ATACS CPU Time</th>
<th>New CPU Time</th>
<th>Hazards</th>
</tr>
</thead>
<tbody>
<tr>
<td>converta</td>
<td>12</td>
<td>0.19</td>
<td>12</td>
<td>0.24</td>
<td>0.11</td>
<td>2/2</td>
</tr>
<tr>
<td>alloc-outbound</td>
<td>11</td>
<td>0.09</td>
<td>3</td>
<td>0.33</td>
<td>0.09</td>
<td>0/0</td>
</tr>
<tr>
<td>chu133</td>
<td>9</td>
<td>0.63</td>
<td>1</td>
<td>0.16</td>
<td>0.11</td>
<td>1/1</td>
</tr>
<tr>
<td>ebergen</td>
<td>9</td>
<td>0.14</td>
<td>1</td>
<td>0.15</td>
<td>0.13</td>
<td>3/3</td>
</tr>
<tr>
<td>half</td>
<td>7</td>
<td>0.41</td>
<td>1</td>
<td>0.13</td>
<td>0.08</td>
<td>1/1</td>
</tr>
<tr>
<td>rpdt</td>
<td>8</td>
<td>2.93</td>
<td>2</td>
<td>0.30</td>
<td>0.10</td>
<td>1/2</td>
</tr>
<tr>
<td>sbuf-ram-write</td>
<td>17</td>
<td>31.77</td>
<td>415</td>
<td>0.32</td>
<td>0.20</td>
<td>1/2</td>
</tr>
<tr>
<td>ram-read-sbuf</td>
<td>17</td>
<td>&gt;678</td>
<td>550</td>
<td>0.34</td>
<td>0.18</td>
<td>0/0</td>
</tr>
<tr>
<td>trimos-send</td>
<td>24</td>
<td>&gt;580</td>
<td>127</td>
<td>10.7</td>
<td>4.87</td>
<td>5/5</td>
</tr>
</tbody>
</table>
### Gate-level Verification

<table>
<thead>
<tr>
<th>Example</th>
<th>Gates</th>
<th>ATACS CPU Time</th>
<th>ATACS Mem (MB)</th>
<th>New CPU Time</th>
<th>New Mem (MB)</th>
<th>Hazards</th>
</tr>
</thead>
<tbody>
<tr>
<td>converta</td>
<td>12</td>
<td>0.19</td>
<td>12</td>
<td>0.24</td>
<td>0.11</td>
<td>2/2</td>
</tr>
<tr>
<td>alloc-outbound</td>
<td>11</td>
<td>0.09</td>
<td>3</td>
<td>0.33</td>
<td>0.09</td>
<td>0/0</td>
</tr>
<tr>
<td>chu133</td>
<td>9</td>
<td>0.63</td>
<td>1</td>
<td>0.16</td>
<td>0.11</td>
<td>1/1</td>
</tr>
<tr>
<td>ebergen</td>
<td>9</td>
<td>0.14</td>
<td>1</td>
<td>0.15</td>
<td>0.13</td>
<td>3/3</td>
</tr>
<tr>
<td>half</td>
<td>7</td>
<td>0.41</td>
<td>1</td>
<td>0.13</td>
<td>0.08</td>
<td>1/1</td>
</tr>
<tr>
<td>rpdf</td>
<td>8</td>
<td>2.93</td>
<td>2</td>
<td>0.30</td>
<td>0.10</td>
<td>1/2</td>
</tr>
<tr>
<td>sbuf-ram-write</td>
<td>17</td>
<td>31.77</td>
<td>415</td>
<td>0.32</td>
<td>0.20</td>
<td>1/2</td>
</tr>
<tr>
<td>ram-read-sbuf</td>
<td>17</td>
<td>&gt;678</td>
<td>550</td>
<td>0.34</td>
<td>0.18</td>
<td>0/0</td>
</tr>
<tr>
<td>trimos-send</td>
<td>24</td>
<td>&gt;580</td>
<td>127</td>
<td>10.7</td>
<td>4.87</td>
<td>5/5</td>
</tr>
</tbody>
</table>

### Gate-level Verification

<table>
<thead>
<tr>
<th>Example</th>
<th>Gates</th>
<th>ATACS Time(s)</th>
<th>ATACS Mem (MB)</th>
<th>New Time(s)</th>
<th>New Mem (MB)</th>
<th>Hazards</th>
</tr>
</thead>
<tbody>
<tr>
<td>sci5v</td>
<td>18</td>
<td>1.35</td>
<td>7.9</td>
<td>0.13</td>
<td>1.3</td>
<td>0/0</td>
</tr>
<tr>
<td>slatch</td>
<td>29</td>
<td>33.5</td>
<td>53.4</td>
<td>0.15</td>
<td>1.8</td>
<td>0/0</td>
</tr>
<tr>
<td>lapbsv</td>
<td>37</td>
<td>20.0</td>
<td>41.5</td>
<td>0.17</td>
<td>1.3</td>
<td>0/0</td>
</tr>
<tr>
<td>elatch</td>
<td>38</td>
<td>183</td>
<td>229</td>
<td>0.28</td>
<td>1.8</td>
<td>0/0</td>
</tr>
<tr>
<td>cnt3</td>
<td>80</td>
<td>&gt;1000</td>
<td>&gt;256</td>
<td>0.24</td>
<td>1.7</td>
<td>--/15</td>
</tr>
<tr>
<td>srgate</td>
<td>85</td>
<td>&gt;1000</td>
<td>&gt;256</td>
<td>0.29</td>
<td>2.3</td>
<td>--/0</td>
</tr>
<tr>
<td>selopt</td>
<td>164</td>
<td>&gt;2000</td>
<td>&gt;256</td>
<td>0.90</td>
<td>3.3</td>
<td>--/46</td>
</tr>
<tr>
<td>cnt11</td>
<td>213</td>
<td>&gt;2000</td>
<td>&gt;256</td>
<td>1.20</td>
<td>4.8</td>
<td>--/52</td>
</tr>
</tbody>
</table>
Gate-level Verification

<table>
<thead>
<tr>
<th>Example</th>
<th>Gates</th>
<th>ATACS Time(s)</th>
<th>ATACS Mem (MB)</th>
<th>New Time(s)</th>
<th>New Mem (MB)</th>
<th>Hazards</th>
</tr>
</thead>
<tbody>
<tr>
<td>scsiSV</td>
<td>18</td>
<td>1.35</td>
<td>7.9</td>
<td>0.13</td>
<td>1.3</td>
<td>0/0</td>
</tr>
<tr>
<td>sLatch</td>
<td>29</td>
<td>33.5</td>
<td>53.4</td>
<td>0.15</td>
<td>1.8</td>
<td>0/0</td>
</tr>
<tr>
<td>lapbsv</td>
<td>37</td>
<td>20.0</td>
<td>41.5</td>
<td>0.17</td>
<td>1.3</td>
<td>0/0</td>
</tr>
<tr>
<td>elatch</td>
<td>38</td>
<td>183</td>
<td>229</td>
<td>0.28</td>
<td>1.8</td>
<td>0/0</td>
</tr>
<tr>
<td>cnt3</td>
<td>80</td>
<td>&gt;1000</td>
<td>&gt;256</td>
<td>0.24</td>
<td>1.7</td>
<td>--/15</td>
</tr>
<tr>
<td>srgate</td>
<td>85</td>
<td>&gt;1000</td>
<td>&gt;256</td>
<td>0.29</td>
<td>2.3</td>
<td>--/0</td>
</tr>
<tr>
<td>selopt</td>
<td>164</td>
<td>&gt;2000</td>
<td>&gt;256</td>
<td>0.90</td>
<td>3.3</td>
<td>--/46</td>
</tr>
<tr>
<td>cnt11</td>
<td>213</td>
<td>&gt;2000</td>
<td>&gt;256</td>
<td>1.20</td>
<td>4.8</td>
<td>--/52</td>
</tr>
</tbody>
</table>
### Matching: Timed Synthesis

<table>
<thead>
<tr>
<th>Example</th>
<th>Simple</th>
<th>Combo</th>
<th>S-Stack</th>
<th>Com-Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc-outbound</td>
<td>54/0</td>
<td>13/0</td>
<td>13/0</td>
<td>13/0</td>
</tr>
<tr>
<td>chu133</td>
<td>34/0</td>
<td>8/0</td>
<td>7/0</td>
<td>6/0</td>
</tr>
<tr>
<td>cnt3</td>
<td>80/12</td>
<td>21/0</td>
<td>21/0</td>
<td>22/0</td>
</tr>
<tr>
<td>cnt11</td>
<td>213/52</td>
<td>37/0</td>
<td>26/0</td>
<td>26/0</td>
</tr>
<tr>
<td>ebergen</td>
<td>43/7</td>
<td>9/2</td>
<td>9/2</td>
<td>6/0</td>
</tr>
<tr>
<td>ram-read-sbuf</td>
<td>70/0</td>
<td>18/0</td>
<td>17/0</td>
<td>13/0</td>
</tr>
<tr>
<td>sbuf-send-ctl</td>
<td>51/5</td>
<td>12/0</td>
<td>11/0</td>
<td>11/0</td>
</tr>
<tr>
<td>trimos-send</td>
<td>87/21</td>
<td>21/3</td>
<td>21/3</td>
<td>18/0</td>
</tr>
</tbody>
</table>

| chul33             | 0      |
| cnt3               | 12     |
| cnt11              | 52     |
| ebergen            | 7      |
| sbuf-send-ctl      | 5      |
| trimos-send        | 21     |
| selopt             | 128    |
### Matching: Timed Synthesis

<table>
<thead>
<tr>
<th>Example</th>
<th>Simple</th>
<th>Combo</th>
<th>S-Stack</th>
<th>Com-Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc-outbound</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chu133</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cnt3</td>
<td>12</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cnt11</td>
<td>52</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ebergen</td>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sbuf-send-ctl</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trimos-send</td>
<td>21</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>selopt</td>
<td>128</td>
<td>19</td>
<td></td>
<td>19</td>
</tr>
</tbody>
</table>
### Matching: Timed Synthesis

<table>
<thead>
<tr>
<th>Example</th>
<th>Simple</th>
<th>Combo</th>
<th>S-Stack</th>
<th>Com-Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc-outbound</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>chu133</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cnt3</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cnt11</td>
<td>52</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ebergen</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>sbuf-send-dtl</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>trimos-send</td>
<td>21</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>selopt</td>
<td>128</td>
<td>19</td>
<td>19</td>
<td>17</td>
</tr>
</tbody>
</table>

### Matching: Untimed Synthesis

<table>
<thead>
<tr>
<th>Example</th>
<th>Simple</th>
<th>Combo</th>
<th>S-Stack</th>
<th>Com-Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc-outbound</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chu133</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cnt3</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cnt11</td>
<td>177</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ebergen</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sbuf-send-dtl</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trimos-send</td>
<td>121</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>selopt</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Matching: Untimed Synthesis

<table>
<thead>
<tr>
<th>Example</th>
<th>Simple</th>
<th>Combo</th>
<th>S-Stack</th>
<th>Com-Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc-outbound</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>chu133</td>
<td>26</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cnt3</td>
<td>60</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cnt11</td>
<td>177</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ebergen</td>
<td>26</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sbuf-send-ctl</td>
<td>17</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trimos-send</td>
<td>121</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>selopt</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>
### Matching: Untimed Synthesis

<table>
<thead>
<tr>
<th>Example</th>
<th>Simple</th>
<th>Combo</th>
<th>S-Stack</th>
<th>Com-Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc-outbound</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>chu133</td>
<td>26</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>cnt3</td>
<td>60</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>cnt11</td>
<td>177</td>
<td>17</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>ebergen</td>
<td>26</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>sbuf-send-ctl</td>
<td>17</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>trimos-send</td>
<td>121</td>
<td>26</td>
<td>26</td>
<td>28</td>
</tr>
<tr>
<td>selopt</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>
## Hazard-Aware Matching

### Example

<table>
<thead>
<tr>
<th></th>
<th>(0,0)</th>
<th>(-1,0)</th>
<th>(0,1)</th>
<th>(1,1)</th>
<th>(-1,1)</th>
<th>(-1,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc-outbound</td>
<td>13/7</td>
<td>13/3</td>
<td>14/0</td>
<td>16/0</td>
<td>14/0</td>
<td>14/0</td>
</tr>
<tr>
<td>chu133</td>
<td>6/4</td>
<td>13/10</td>
<td>6/4</td>
<td>6/4</td>
<td>6/4</td>
<td>6/4</td>
</tr>
<tr>
<td>converta</td>
<td>15/11</td>
<td>24/16</td>
<td>16/5</td>
<td>16/2</td>
<td>16/5</td>
<td>16/5</td>
</tr>
<tr>
<td>ebergen</td>
<td>9/5</td>
<td>13/8</td>
<td>9/3</td>
<td>9/5</td>
<td>9/3</td>
<td>9/3</td>
</tr>
<tr>
<td>half</td>
<td>6/4</td>
<td>10/8</td>
<td>5/2</td>
<td>5/2</td>
<td>5/2</td>
<td>5/2</td>
</tr>
<tr>
<td>newick</td>
<td>15/4</td>
<td>18/2</td>
<td>15/2</td>
<td>15/2</td>
<td>17/2</td>
<td>17/2</td>
</tr>
<tr>
<td>ram-read-sbuf</td>
<td>17/7</td>
<td>26/12</td>
<td>18/2</td>
<td>18/2</td>
<td>18/2</td>
<td>18/2</td>
</tr>
<tr>
<td>rpdfi</td>
<td>7/4</td>
<td>10/9</td>
<td>8/4</td>
<td>8/4</td>
<td>8/5</td>
<td>8/4</td>
</tr>
<tr>
<td>sbuf-ram-write</td>
<td>13/5</td>
<td>22/13</td>
<td>13/4</td>
<td>13/5</td>
<td>13/4</td>
<td>13/4</td>
</tr>
<tr>
<td>sbuf-send-ctl</td>
<td>10/2</td>
<td>20/13</td>
<td>10/2</td>
<td>10/2</td>
<td>10/4</td>
<td>10/4</td>
</tr>
<tr>
<td>trimos-send</td>
<td>33/24</td>
<td>56/45</td>
<td>37/28</td>
<td>33/24</td>
<td>37/28</td>
<td>33/24</td>
</tr>
<tr>
<td>cntd</td>
<td>14/11</td>
<td>24/20</td>
<td>13/6</td>
<td>13/4</td>
<td>13/4</td>
<td>13/4</td>
</tr>
<tr>
<td>cntl1</td>
<td>28/20</td>
<td>51/40</td>
<td>26/15</td>
<td>28/17</td>
<td>26/15</td>
<td>26/15</td>
</tr>
</tbody>
</table>

### Test conditions for 21 circuits:
- Untimed synthesis.
- Untimed verification.
- Common-input library implementation.
Hazard-Aware Matching

- Test conditions for 21 circuits:
  - Untimed synthesis.
  - Untimed verification.
  - Common-input library implementation.
- Compared matching using various $W_1, W_2$:
  - $W_1, W_2 = 0, 0$: No hazard awareness.
  - $W_1, W_2 = 0, 1$: Ignore mono, encap ack.
  - $W_1, W_2 = -1, 1$: Discourage mono encap, encap ack.

Hazard-Aware Matching

- Results (win means fewer hazards in final netlist):
  - 0,1 compared to −1,0: 19 wins, 0 losses
  - 0,1 compared to 0,0: 13 wins, 2 losses
  - 0,1 compared to 1,1: 7 wins, 3 losses
  - 0,1 compared to −1,1: 4 wins, 1 loss
  - 0,1 compared to −1,2: 3 wins, 2 losses
Hazard-Aware Matching

- Results (win means fewer hazards in final netlist):
  - 0,1 compared to –1,0 19 wins, 0 losses
  - 0,1 compared to 0,0 13 wins, 2 losses
  - 0,1 compared to 1,1 7 wins, 3 losses
  - 0,1 compared to –1,1 4 wins, 1 loss
  - 0,1 compared to –1,2 3 wins, 2 losses

- Observations:
  - Must encapsulate acknowledgement hazards.
  - Give more emphasis to acknowledgment hazards.
Hazard-Aware Matching

Results (win means fewer hazards in final netlist):
- 0,1 compared to −1,0: 19 wins, 0 losses
- 0,1 compared to 0,0: 13 wins, 2 losses
- 0,1 compared to 1,1: 7 wins, 3 losses
- 0,1 compared to −1,1: 4 wins, 1 loss
- 0,1 compared to −1,2: 3 wins, 2 losses

Observations:
- Must encapsulate acknowledgement hazards.
- Give more emphasis to acknowledgment hazards.
- Perhaps ignore monotonicity hazards altogether.
- Try multiple solutions and pick the best.
Conclusions

- Synchronous technology mapping flow is adaptable to timed circuit technology mapping.
- Gate-level hazard verifier using timing is feasible.
  - Results show number of false hazards is small.
  - Method scales well for large examples.
- gC’s work well when short-circuit and common-input issues are considered.
- Timed synthesis and verification, hazard-aware matching, in most cases, produce hazard-free circuits.
Contributions

- Adapting synchronous technology mapping flow to timed circuits.
- Efficient gate-level hazard verification using timing.
- Hazard-aware matching and covering utilizing gC’s in circuit solutions.
- Evaluating library complexity for implementation.
Future Work

- Investigate:
  - Whether or not hazards are false.
  - Hazard reduction via input re-ordering.
  - Hazard reduction via inertial delay models.
  - Hazard-free outputs with internal hazardous behavior.
  - Circuits with internal cycles.
  - Using other specification forms to increase example suite.
  - Monotonicity hazard creation during hazard-aware matching.

Case Studies

- False Hazards.
- Non-Propagating Acknowledgment Hazard.
- Non-Optimum Covered Circuit.
False Hazards

- Mono hazard on output \( t \) caused by \( b58 \) in state \( 10001 \).

Non-Propagating Ack. Hazard
Non-Optimum Covered Circuit

Simple Example