The Majority Logic Optimization Paradigm

Luca Amarù
Integrated Systems Laboratory (LSI), EPFL, Switzerland.
Why Majority Logic?

- Majority logic is a powerful generalization of AND/ORs.
- $\text{MAJ}(x_1, x_2, x_3, \ldots, x_n) = 1$ if more than $n/2$ inputs are 1.
- $\text{MAJ}(a, b, c) = ab + ac + bc$. $\text{MAJ}(a, b, 1) = a + b$. $\text{MAJ}(a, b, 0) = ab$.
- More compact as compared to AND-OR logic:
How Powerful is Majority?

- Majority logic vs. AND/OR logic in representing arithmetic circuits.
- Consider small depth representations, target 4/5 logic levels.

Set of all functions

Unate

Monotone

Threshold

Majority

Self-dual

AA. Sherstov, Separating AC 0 from depth-2 majority circuits, Proc. STOC, 2007


Exploiting Majority Logic

There is an exponential gap between the expressive power of traditional AND/OR circuits and MAJ circuits when considering arithmetic.

So, why not exploiting the majority logic representation expressiveness when synthesizing circuits?

In order to manipulate majority logic we define a homogenous data structure.

We call it Majority-Inverter Graph.
Definition: An MIG is a logic network consisting of 3-input majority nodes and regular/complemented edges.
MIG Properties

AOIGs $\rightarrow$ MIGs

MIGs include AOIGs include AIGs
Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$

Majority: if $x = y$, $M(x, y, z) = x = y$
if $x = y'$, $M(x, y, z) = z$

Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$

Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$

Inverter Propagation: $M'(x, y, z) = M(x', y', z')$

Theorem: $(B, M', 0, 1)$ subject to axiom in $\Omega$ is a Boolean algebra
1- **Commutativity:** \( M(x, y, z) = M(y, x, z) = M(z, y, x) \)
2- **Majority:** if \( x = y \), \( M(x, y, z) = x = y \)  
    if \( x = y' \), \( M(x, y, z) = z \)
3- **Associativity:** \( M(x, u, M(y, u, z)) = M(z, u, M(y, u, x)) \)
4- **Distributivity:** \( M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z) \)
5- **Inverter Propagation:** \( M'(x, y, z) = M(x', y', z') \)
MIG Boolean Algebra

1- Commutativity: \( M(x, y, z) = M(y, x, z) = M(z, y, x) \)

2- Majority: if \( x = y \), \( M(x, y, z) = x = y \)
   
   if \( x = y' \), \( M(x, y, z) = z \)

3- Associativity: \( M(x, u, M(y, u, z)) = M(z, u, M(y, u, x)) \)

4- Distributivity: \( M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z) \)

5- Inverter Propagation: \( M'(x, y, z) = M(x', y', z') \)
1- **Commutativity:** \( M(x, y, z) = M(y, x, z) = M(z, y, x) \)

2- **Majority:** 
   - if \( x = y \), \( M(x, y, z) = x = y \)
   - if \( x = y' \), \( M(x, y, z) = z \)

3- **Associativity:** \( M(x, u, M(y, u, z)) = M(z, u, M(y, u, x)) \)

4- **Distributivity:** \( M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z) \)

5- **Inverter Propagation:** \( M'(x, y, z) = M(x', y', z') \)
1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$
2- Majority: if $x = y$, $M(x, y, z) = x = y$
   if $x = y'$, $M(x, y, z) = z$
3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$
4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$
5- Inverter Propagation: $M'(x, y, z) = M(x', y', z')$
MIG Boolean Algebra

1- Commutativity: \( M(x, y, z) = M(y, x, z) = M(z, y, x) \)
2- Majority: if \( x = y \), \( M(x, y, z) = x \) = \( y \)
    \[ \text{if}(x = y'), \ M(x, y, z) = z \]
3- Associativity: \( M(x, u, M(y, u, z)) = M(z, u, M(y, u, x)) \)
4- Distributivity: \( M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z) \)
5- Inverter Propagation: \( M'(x, y, z) = M(x', y', z') \)
1- **Commutativity**: \( M(x, y, z) = M(y, x, z) = M(z, y, x) \)

2- **Majority**: if \( x = y \), \( M(x, y, z) = x = y \)  
    if \( x = y' \), \( M(x, y, z) = z \)

3- **Associativity**: \( M(x, u, M(y, u, z)) = M(z, u, M(y, u, x)) \)

4- **Distributivity**: \( M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z) \)

5- **Inverter Propagation**: \( M'(x, y, z) = M(x', y', z') \)

\( \bowtie \) is the basis for more elaborated optimization transformations.

\( \bowtie \) For instance, it is possible to extend associativity:

\( \bowtie \) **Complementary Associativity**:

\( M(x, u, M(y, u', z)) = M(x, u, M(y, x, z)) \)

**Theorem**: MIG Boolean algebra is sound and complete
1- **Commutativity:** \( M(x, y, z) = M(y, x, z) = M(z, y, x) \)

2- **Majority:**
   - if \( x = y \), \( M(x, y, z) = x = y \)
   - if \( x = y' \), \( M(x, y, z) = z \)

3- **Associativity:** \( M(x, u, M(y, u, z)) = M(z, u, M(y, u, x)) \)

4- **Distributivity:** \( M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z) \)

5- **Inverter Propagation:** \( M'(x, y, z) = M(x', y', z') \)

By using \( \Omega \) transformations we want to optimize an MIG

What do we care about?

- **Area**  ➔  MIG size (details in TCAD’15)
- **Delay**  ➔  MIG depth – discussed in this presentation
- **Power**  ➔  MIG SW Activity (details in TCAD’15)
1- Commutativity: $M(x, y, z) = M(y, x, z) = M(z, y, x)$

2- Majority: if $x = y$, $M(x, y, z) = x$
   if $x \neq y$, $M(x, y, z) = z$

3- Associativity: $M(x, u, M(y, u, z)) = M(z, u, M(y, u, x))$

4- Distributivity: $M(x, y, M(u, v, z)) = M(M(x, y, u), M(x, y, v), z)$

5- Inverter Propagation: $M'(x, y, z) = M(x, y, z')$

How to reduce the depth of an MIG?

Let’s see what comes handy from $\Omega$:
MIG Depth Optimization

- **Rationale:** move critical variables closer to the outputs via associativity, distributivity and majority rules

- **Reshaping the MIG with other Ω rules**

\[ f = x(y+uv) \]

**Module optDC**
```verilog
module optDC ( pi01, pi02, pi03, pi04, po0 );
    input pi01, pi02, pi03, pi04;
    output po0;
    wire n5, n6, n7, n8;
    INV_X8 U6 (.A(pi03), .Y(n7));
    INV_X8 U7 (.A(pi01), .Y(n6));
    NOR2_X1 U8 (.A(n6), .B(n7), .Y(n5));
    NAND2_X1 U9 (.A(pi04), .B(n5), .Y(n9));
    NAND2_X1 U10 (.A(pi01), .B(pi02), .Y(n8));
    NAND2_X1 U11 (.A(n9), .B(n8), .Y(po0));
endmodule
```

**Area = 1.68 \text{um}^2**  
Levels of logic = 4  
Delay = 40 ps

**Module optMIG**
```verilog
module optMIG ( pi01, pi02, pi03, pi04, po0 );
    input pi01, pi02, pi03, pi04;
    output po0;
    wire n1, n2, n3;
    INV_X8 U1 (.A(pi01), .Y(n1));
    NAND2_X1 U2 (.A(pi04), .B(pi03), .Y(n2));
    NAND2_X1 U3 (.A(pi01), .B(pi02), .Y(n3));
    MIN3_X1 U4 (.A(n2), .B(n3), .C(n1), .Y(po0));
endmodule
```

**Area = 1.19 \text{um}^2**  
Levels of logic = 2  
Delay = 30 ps
Experimental Results

Methodology

- We implemented in C language a MIG optimizer, called Mighty.
- We target high speed results with bounded area and power overhead.
- Pure logic optimization experiments against AIG-optimization.
- Case study on adder optimization.
- Complete ASIC design experiments against commercial tools.
- Results verified with Synopsys Formality.
- Nanotechnology design experiments.
Logic Optimization Experiments:
Adders Case Study

<table>
<thead>
<tr>
<th>Adder type</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Original AIC Size</th>
<th>Depth</th>
<th>Optimized MIG Size</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-op 32 bit</td>
<td>64</td>
<td>33</td>
<td>352</td>
<td>96</td>
<td>610</td>
<td>12</td>
</tr>
<tr>
<td>2-op 64 bit</td>
<td>128</td>
<td>65</td>
<td>704</td>
<td>192</td>
<td>1159</td>
<td>11</td>
</tr>
<tr>
<td>2-op 128 bit</td>
<td>256</td>
<td>129</td>
<td>1408</td>
<td>384</td>
<td>14672</td>
<td>19</td>
</tr>
<tr>
<td>2-op 256 bit</td>
<td>512</td>
<td>257</td>
<td>2816</td>
<td>768</td>
<td>7650</td>
<td>16</td>
</tr>
<tr>
<td>3-op 32 bit</td>
<td>96</td>
<td>32</td>
<td>760</td>
<td>68</td>
<td>1938</td>
<td>16</td>
</tr>
<tr>
<td>4-op 64 bit</td>
<td>256</td>
<td>66</td>
<td>1336</td>
<td>136</td>
<td>2212</td>
<td>18</td>
</tr>
</tbody>
</table>
ASIC Design Experiments

Advanced 22nm CMOS

MIG as front end to LS & RD

Well-established 90nm CMOS

MIG as front end to LS & RD

27 bits in and 31 bits out

Behavioral

```
module div32 (a, b, quotient_uns, quotient_tc, remainder_uns, 
remainder_tc);

parameter width = 32;

input [width-1:0] a,
output [width-1:0] b;
output signed [width-1:0] quotient;
output signed [width-1:0] quotient_tc;
output signed [width-1:0] remainder;
output signed [width-1:0] remainder_tc;

// operators for quotient
assign quotient = $signed(a) / $signed(b);
assign quotient_uns = a % b;
assign $signed(quotient_uns) = $signed(quotient);
assign quotient_tc = $signed(quotient) + $signed(b);
assign $signed(quotient_tc) = $signed(quotient) + $signed(b);
assign $signed(quotient_tc) = $signed(quotient) + $signed(b);

// operators for remainder
assign remainder = a % b;
assign remainder_uns = a % b;
assign remainder_tc = $signed(remainder) + $signed(b);
assign $signed(remainder_tc) = $signed(remainder) + $signed(b);
assign $signed(remainder_tc) = $signed(remainder) + $signed(b);

endmodule
```

Area: 0.21 mm²
Delay: 11.22 ns
GC: 37k

MIG

```
module div32 (a, b, quotient_uns, quotient_tc, remainder_uns, 
remainder_tc);

parameter width = 32;

input [width-1:0] a,
output [width-1:0] b;
output signed [width-1:0] quotient;
output signed [width-1:0] quotient_tc;
output signed [width-1:0] remainder;
output signed [width-1:0] remainder_tc;

// operators for quotient
assign quotient = $signed(a) / $signed(b);
assign quotient_uns = a % b;
assign $signed(quotient_uns) = $signed(quotient);
assign quotient_tc = $signed(quotient) + $signed(b);
assign $signed(quotient_tc) = $signed(quotient) + $signed(b);
assign $signed(quotient_tc) = $signed(quotient) + $signed(b);

// operators for remainder
assign remainder = a % b;
assign remainder_uns = a % b;
assign remainder_tc = $signed(remainder) + $signed(b);
assign $signed(remainder_tc) = $signed(remainder) + $signed(b);
assign $signed(remainder_tc) = $signed(remainder) + $signed(b);

endmodule
```

Area: 0.18 mm²
Delay: 10.10 ns
GC: 24k
## Nanotechnology Design

### Spin Wave Device

**feature size 24 nm**

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>I/O</th>
<th>SWD technology - MIG</th>
<th>SWD technology - AIG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A ($\mu m^2$)</td>
<td>D (ns)</td>
</tr>
<tr>
<td>bigkey</td>
<td>487/421</td>
<td>152.50</td>
<td>3.14</td>
</tr>
<tr>
<td>my_adder</td>
<td>33/17</td>
<td>9.42</td>
<td>6.11</td>
</tr>
<tr>
<td>cla</td>
<td>129/65</td>
<td>36.57</td>
<td>7.60</td>
</tr>
<tr>
<td>dalu</td>
<td>75/16</td>
<td>50.47</td>
<td>6.71</td>
</tr>
<tr>
<td>b9</td>
<td>41/21</td>
<td>5.60</td>
<td>2.24</td>
</tr>
<tr>
<td>count</td>
<td>35/16</td>
<td>6.36</td>
<td>2.54</td>
</tr>
<tr>
<td>alu4</td>
<td>14/8</td>
<td>47.81</td>
<td>4.62</td>
</tr>
<tr>
<td>clma</td>
<td>416/115</td>
<td>433.59</td>
<td>12.96</td>
</tr>
<tr>
<td>mm30a</td>
<td>124/120</td>
<td>41.57</td>
<td>30.52</td>
</tr>
<tr>
<td>s38417</td>
<td>1494/1571</td>
<td>319.86</td>
<td>7.01</td>
</tr>
<tr>
<td>misex3</td>
<td>14/14</td>
<td>45.84</td>
<td>4.33</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>212/176</td>
<td>90.02</td>
<td>9.07</td>
</tr>
</tbody>
</table>
Conclusions

 Majority-Inverter Graphs support optimization techniques.

 The expressive power of MIG Boolean algebra axioms, such as distributivity and inverter propagation, permits more agile logic manipulation.

 MIG optimization show promising results.

  - Strong optimization of arithmetic, e.g., adders: automatic ripple-carry to carry-look-ahead transformation.
  - At the design level, we showed reductions in delay, area and power as compared to a modern commercial ASIC flow.
  - Efficient design of circuits in nanotechnologies where majority is the primitive gate for logic computation.
Questions?

Thank you for your attention!
Backup slides
Pushing MIG Optimization to the Limits

- Good, we have an algebraic framework for MIG optimization.

- But…

- There exist practical cases where algebraic optimization heuristics fail to produce improved results.

- … so how to attain further optimization?

- Exploit the Boolean properties of MIG data structure.

MIG Boolean Optimization!
MIG Boolean Optimization

- Exploit the Boolean nature of MIGs.

- An MIG is hierarchical majority voting system.
- Majority voting can correct various types of bit-errors.
MIG Boolean Optimization

- MIG voting resilience allows us to insert logic errors.

Original MIG functionality

- Logic errors smartly placed can heavily simplify a logic network.
- Not all types of logic errors can be inserted.
- To be safe, errors $f^A$, $f^B$ and $f^C$ must be pairwise orthogonal.
## IWLS’05+HDL Benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>I/O</th>
<th>Opt. MIG</th>
<th>MIGHTy</th>
<th>ABC</th>
<th>Opt. AIG</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open Cores IWLS’05</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSP</td>
<td>4223/3953</td>
<td>4017</td>
<td>34</td>
<td>11077</td>
<td>11</td>
<td>7.98</td>
</tr>
<tr>
<td>acl7_ctrl</td>
<td>2255/2250</td>
<td>10745</td>
<td>8</td>
<td>2917</td>
<td>3</td>
<td>6.52</td>
</tr>
<tr>
<td>aes_core</td>
<td>789/668</td>
<td>20947</td>
<td>18</td>
<td>3902</td>
<td>4</td>
<td>11.78</td>
</tr>
<tr>
<td>des_area</td>
<td>386/72</td>
<td>4186</td>
<td>22</td>
<td>735</td>
<td>6</td>
<td>1.04</td>
</tr>
<tr>
<td>des_perf</td>
<td>9042/9038</td>
<td>67194</td>
<td>15</td>
<td>12796</td>
<td>3</td>
<td>34.22</td>
</tr>
<tr>
<td>eternet</td>
<td>18793/18625</td>
<td>75058</td>
<td>16</td>
<td>10408</td>
<td>6</td>
<td>33.38</td>
</tr>
<tr>
<td>spi</td>
<td>274/276</td>
<td>3337</td>
<td>19</td>
<td>612</td>
<td>6</td>
<td>3.71</td>
</tr>
<tr>
<td>ss_pcm</td>
<td>106/98</td>
<td>397</td>
<td>6</td>
<td>104</td>
<td>2</td>
<td>0.01</td>
</tr>
<tr>
<td>systemcaes</td>
<td>930/819</td>
<td>9547</td>
<td>25</td>
<td>1845</td>
<td>7</td>
<td>5.26</td>
</tr>
<tr>
<td>systemcdes</td>
<td>314/258</td>
<td>2453</td>
<td>19</td>
<td>515</td>
<td>5</td>
<td>2.21</td>
</tr>
<tr>
<td>tv80</td>
<td>373/404</td>
<td>7397</td>
<td>30</td>
<td>1980</td>
<td>11</td>
<td>6.43</td>
</tr>
<tr>
<td>usb_func</td>
<td>1860/1846</td>
<td>12995</td>
<td>19</td>
<td>3333</td>
<td>5</td>
<td>13.45</td>
</tr>
<tr>
<td>usb_phy</td>
<td>113/111</td>
<td>372</td>
<td>8</td>
<td>136</td>
<td>2</td>
<td>0.11</td>
</tr>
<tr>
<td><strong>IWLS’05 total</strong></td>
<td></td>
<td>66791</td>
<td>96</td>
<td>120.38</td>
<td>277618</td>
<td></td>
</tr>
<tr>
<td><strong>Arithmetic HDL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUL32</td>
<td>64/64</td>
<td>9096</td>
<td>36</td>
<td>1852</td>
<td>10</td>
<td>2.90</td>
</tr>
<tr>
<td>lsw32</td>
<td>32/36</td>
<td>2171</td>
<td>164</td>
<td>544</td>
<td>54</td>
<td>1.07</td>
</tr>
<tr>
<td>mules</td>
<td>128/128</td>
<td>25773</td>
<td>109</td>
<td>6537</td>
<td>41</td>
<td>1.38</td>
</tr>
<tr>
<td>max</td>
<td>512/130</td>
<td>4210</td>
<td>29</td>
<td>1023</td>
<td>12</td>
<td>1.67</td>
</tr>
<tr>
<td>square</td>
<td>64/127</td>
<td>17559</td>
<td>40</td>
<td>4393</td>
<td>13</td>
<td>18.66</td>
</tr>
<tr>
<td>log2</td>
<td>32/32</td>
<td>31328</td>
<td>261</td>
<td>8809</td>
<td>59</td>
<td>23.32</td>
</tr>
<tr>
<td><strong>Arithmetic total</strong></td>
<td></td>
<td>149727</td>
<td>1222</td>
<td>555</td>
<td>155.62</td>
<td>154436</td>
</tr>
</tbody>
</table>

### Results:

**IWLS’05:**
- **-17.98% depth**
- **-12.65% size**
- **-10.00% SW act.**

**Arithmetic HDL:**
- **-26.69% depth**
- **-7.7% size**
- **-0.1% SW act.**