Model Extraction for designs based on Switches for Formal Verification

Amar Patel, Mentor Graphics, India
Naman Jain, Mentor Graphics, India
Agenda

• Introduction
• Model Extraction
• Model Properties
• Results
• Future Work
Introduction

• SV LRM IEEE1800-2009 - Section 28 defines 12 switches
  – MOS Switches
    • Cmos, pmos, nmos, rcmos, rpmos, rnmos
  – Bidirectional switches
    • Tran, tranif0, tranif1, rtran, rtranif0, rtranif1

• Why Switches?
  – Switches and Gates provide a much closer one-to-one mapping between the actual circuit and the model
  – Propagation Delays can be specified
  – There is no continuous assignment equivalent to the bidirectional transfer gate
Use of Switches

\[ f = (xy)' \]

```verbatim
module nand2 (x, y, f);
    input x, y;
    output f;
    supply1 vdd;
    supply0 gnd;
    wire a;

    pmos p1 (f, vdd, x);
    pmos p2 (f, vdd, y);
    nmos n1 (f, a, x);
    nmos n2 (a, gnd, y);
endmodule
```
Expanding Scope of Formal Verification

• Formal verification is no longer confined to block level runs.
• Use of specialized Formal Applications at SOC levels have started exposing Formal to complete SOC designs.

• Thus formal is being exposed to
  – Gate level/IP blocks containing switches
Switches in Formal flow

- Unresolved
-Incorrect results

RTL
Gate Level

Formal Compilation

Unsynthesizable

Library

Pessimistic analysis

Unresolved
Incorrect results
Why switches impose a challenge

• MOS switches are simple and are handled by majority of the tools

• While conditional bidirectional switches like tranif0/tranif1 impose a challenge.
  • There is no continuous assignment equivalent to the bidirectional transfer gate.
  • If there is a chain of such switches, then impact of all inputs in the chain needs to be conditionally applied to each port.
Chain of Bidirectional Switches

- Here each signal is impacted by all other inputs in the chain.
- Control for this impact is function of individual control signals.
- P1 is driven by P2 and P3 under some function of C1 and C2.
- Similarly P3 is driven by P2 and P1 under some condition dependent on C1 and C2.
- This dependency graph explodes as the number of elements in the chain increases.
So how do we handle bidirectional switches?

• Functional model for instance of tran switch should ensure
  – Tran I1 (p1,p2)

• Readers of p1 get the resolved value of drivers of p1 and p2

• Readers of p2 get the resolved value of drivers of p1 and p2

MDRF = Multiple Driver Resolution Function
Modelling Bidirectional Switches

- tran
  - tran T (A, B)
    - A = A_drivers MDRF B_drivers
    - B = B_drivers MDRF A_drivers

- tranif0
  - tranif0 T0 (A, B, control)
    - A = A_drivers MDRF ((control === 1) ? z : B_drivers)
    - B = B_drivers MDRF ((control === 1) ? z : A_drivers)

- tranif1
  - tranif1 T1 (A, B, control)
    - A = A_drivers MDRF ((control === 0) ? z : B_drivers)
    - B = B_drivers MDRF ((control === 0) ? z : A_drivers)
Handling Chain of Bidirectional Switches

\[ \text{tranif1 } \text{I1 (p1, p2, c1)} \]
\[ \text{tranif1 } \text{I2 (p1, p3, c2)} \]

\[ p1 = p1_{\text{drivers}} \]
\[ \text{MDRF} \]
\[ (c1 \equiv 0) ? z : p2_{\text{drivers}} \]
\[ \text{MDRF} \]
\[ (c2 \equiv 0) ? z : p3_{\text{drivers}} \]

\[ p2 = p2_{\text{drivers}} \]
\[ \text{MDRF} \]
\[ (c1 \equiv 0) ? z : p1_{\text{drivers}} \]
\[ \text{MDRF} \]
\[ (((c1 \equiv 0) \&\& (c2 \equiv 0)) ? z : p3_{\text{drivers}} \]

\[ p3 = p3_{\text{drivers}} \]
\[ \text{MDRF} \]
\[ (c1 \equiv 0) ? z : p1_{\text{drivers}} \]
\[ \text{MDRF} \]
\[ (((c1 \equiv 0) \&\& (c2 \equiv 0)) ? z : p2_{\text{drivers}} \]
Chain of Bidirectional Switches

- As the chain of transition increases, the dependency of each signal on all other inputs increases.
- The control function becomes increasingly complex.

```plaintext
tranif1  I1  (p1, p2, c1)
tranif1  I2  (p1, p3, c2)
tranif1  I3  (p2, p4, c3)

p1 = p1_drivers
MDRF
(c1 === 0) ? z : p2_drivers
MDRF
(c2 === 0) ? z : p3_drivers
MDRF
((c1 === 0) && (c3 === 0)) ? z : p4_drivers

p3 = p3_drivers
MDRF
(c2 === 0) ? z : p1_drivers
MDRF
((c2 === 0) && (c1 === 0)) ? z : p2_drivers
MDRF
(c2 === 0) && (c1 === 0) && (c3 === 0)) ? z : p4_drivers
```
Optimizations

Both inputs constants
supply0 p1;
supply1 p2;
tran t1 (p1, p2);

Action taken
Remove the tran switch

One input constants
supply0 p1;
wire p2;
tran t1 (p1, p2);

Action taken
p2=p1

One input undriven
Input in1;
wire p1,p2;
assign p1 = in1;
tran t1 (p1, p2)

Action taken
p2=p1

Merging switches
tranif0 t1 (p1, p2, c1) ;
tranif0 t2 (p1, p2, c2);

Action taken
Merged to tranif0 t (p1, p2, (c1 || c2))
MOS Switches

• Buffer between input and output pin

• pmos

\[ \text{pmos} \ P \ (\text{out, data, control}) \]

\[ \text{out} = (\text{control} === 1) \ ? \ z \ : \ data \]

• nmos

\[ \text{nmos} \ N \ (\text{out, data, control}) \]

\[ \text{out} = (\text{control} === 0) \ ? \ z \ : \ data \]

• cmos

\[ \text{cmos} \ C \ (\text{out, data, nControl, pControl}) \]

\[ \text{out} = ((\text{nControl} === 0) \ \&\& \ (\text{pControl} === 1)) \ ? \ z \ : \ data \]
Impact of Modelling

Fired

Compile Warning: Module is treated as a black box. Module 'NOT_SWITCH' contains unsupported 'tran' construct.

Formal Warning: Firing with Warning: A counter example to the assertion exists. But, the firing involves a formal control point that is not a primary input, uninitialized register or cut point.

Proven
Other Model Properties

• Generic
• Formal Friendly
• Functionally Accurate
• Debug Friendly
## Results

<table>
<thead>
<tr>
<th>Design</th>
<th>Without Modelling</th>
<th>With Modelling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td>No. of bidirectional switches</td>
</tr>
<tr>
<td></td>
<td>Design1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design</th>
<th>Without Modelling</th>
<th>With Modelling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td>No. of bidirectional switches</td>
</tr>
<tr>
<td></td>
<td>Design2</td>
<td>3529</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design</th>
<th>Time taken in Modelling (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Design3</td>
</tr>
</tbody>
</table>
Future Work

• Other Sources
  – pullup/pulldown
  – tri0, tri1

• MDRF Enhancements

• Strength and Delays
Questions
For a chain of bidirectional switches -
Let N signals are connected with a chain of M switches.

For each bidirectional switch (A, B, control) in M.
{
    expr_A = MDRF(values_on_A)
    expr_B = MDRF(values_on_B)
    propagate(A, expr_B, control)
    propagate(B, expr_A, control)
}

propagate(target, source_expr, control)
{
    cache - Cache of nodes in current path
    Insert(cache, target)
    Assign(target, source_expr, control)
    propagate_recurse(target, source_expr, control)
    Remove(cache, target)
}

For non-conditional bidirectional switches control can be assumed as 1
Primitive Cell Optimization

• Identifying Primary Cells
  – Heuristics

module nand2 (out, a1, a2);
  output out;
  supply1 vpwr;
  supply0 vgnd;
  input a1, a2;
  tranif1 I1 (net1, vgnd, a1);
  tranif1 I2 (out, net1, a2);
  tranif0 I3 (out, vpwr, a1);
  tranif0 I4 (out, vpwr, a2);
endmodule

Applying optimization 4 on I3 and I4
  tranif0 I3_MERGE_I4 (out, vpwr, a1 || a2)

Applying optimization 1 on I1 and I3_MERGE_I4
  net1 = (a1 === 0) ? z : vgnd                         (1)
  out = ((a1 === 1) && (a2 === 1)) ? z : vpwr  (2)

Applying optimization 5 on I2
  out = (a2 === 0) ? z : net1;                          (3)

Merging (1) and (3)
  out = (a2 === 0) ? z : (a1 === 0) ? z : vgnd  (4)

Now the cell is optimized to:
  out = (((a1 === 1) && (a2 === 1)) ? z : vpwr)
       MDRF
       ((a2 === 0) ? z : (a1 === 0) ? z : vgnd)
Functional Accuracy

• MOS Switches
  – Comparison with LRM truth table

\[
\text{out} = (\text{control} === 1) \ ? \ 'z' : \text{data}
\]

Symbol L represents a result that has value 0 or z
Symbol H represents a result that has value 1 or z
Functional Accuracy

• Bidirectional Switches
  – Comparing Simulator Results
  – Properties with equivalent functionality
    • Run Formal Engine