PROJECT #2: Equivalence Checking with Formality

ECE 582

Abram Fouts

Mike Tian

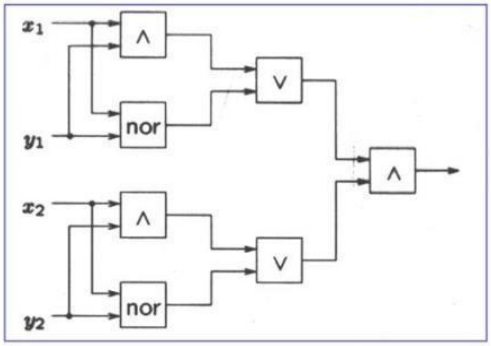
February 4th, 2021

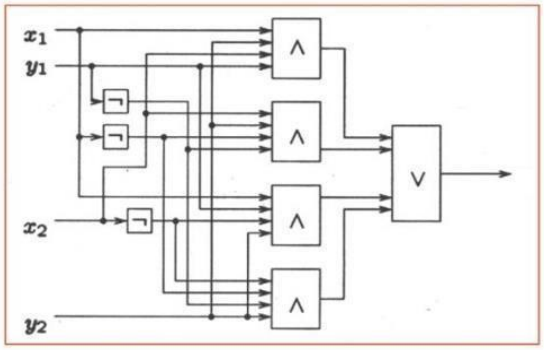
Background

Using system verilog / verilog code compiled in modelSim was used to create design files for Formality. Formality is used to check for the equivalence between circuits, netlists, libraries, and many other items.

Task 1: Describe the following two circuits C1 and C2 in Verilog. Use the formality to check if they are equivalent. You need to use the same names on the corresponding inputs and outputs.

Task 1.1:





.sv code

```
module C1(X1, Y1, X2, Y2, OUT);
input X1, Y1, X2, Y2;
output OUT;

wire AND1, NOR1, OR1;
wire AND2, NOR2, OR2;

assign AND1 = X1 & Y1;
assign NOR1 = ~(X1 | Y1);

assign AND2 = X2 & Y2;
assign NOR2 = ~(X2 | Y2);

assign OR1 = AND1 | NOR1;
assign OR2 = AND2 | NOR2;

assign OUT = OR1 & OR2;

endmodule
```

```
module C2(X1, Y1, X2, Y2, OUT);
input X1, Y1, X2, Y2;
output OUT;

wire AND1, AND2, AND3, AND4;

assign AND1 = X1 & X2 & Y1 & Y2;
assign AND2 = ~X1 & X2 & ~Y1 & Y2;
assign AND3 = X1 & ~X2 & Y1 & Y2;
assign AND4 = ~X1 & ~X2 & ~Y1 & Y2;

assign OUT = AND1 | AND2 | AND3 | AND4;

endmodule
```

Formality Screenshots

Read Verilog - loading reference design

```
Build: 4755953
Hostname: mo.ece.pdx.edu
Current time: Sun Feb  7 11:34:43 2021

Loading db file '/pkgs/synopsys/2017/formality/libraries/syn/gtech.db'
fm_shell (setup)> read_verilog -r task1.sv
No target library specified, default is WORK
Loading verilog file '/u/abfouts/582_project2/task1.sv'
Current container set to 'r'
1
fm_shell (setup)> set_top C1
Setting top design to 'r:/WORK/C1'
Status:   Elaborating design C1   ...
Status:  Implementing inferred operators...
Top design successfully set to 'r:/WORK/C1'
Reference design set to 'r:/WORK/C1'
1
```

Read Verilog - loading implementation design

```
fm_shell (setup)> read_verilog -i task1.sv
No target library specified, default is WORK
Loading verilog file '/u/abfouts/582_project2/task1.sv'
Current container set to 'i'
1
fm_shell (setup)> set_top C2
Setting top design to 'i:/WORK/C2'
Status:   Elaborating design C2   ...
Status:  Implementing inferred operators...
Top design successfully set to 'i:/WORK/C2'
Implementation design set to 'i:/WORK/C2'
1
```

Match - running Formality's matching algorithms

```
fm_shell (setup)> match
Reference design is 'r:/WORK/C1'
Implementation design is 'i:/WORK/C2'
Status:  Checking designs...
Status:  Building verification models...
Status:  Matching...

********************************** Matching Results **********************************
 1 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 4 Matched primary inputs, black-box outputs
 0(0) Unmatched reference(implementation) compare points
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
**************************************************************************************

1
```

Verify - running Formality's verification algorithms

```
fm_shell (match)> verify
Reference design is 'r:/WORK/C1'
Implementation design is 'i:/WORK/C2'

********************************** Matching Results **********************************
 1 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 4 Matched primary inputs, black-box outputs
 0(0) Unmatched reference(implementation) compare points
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
**************************************************************************************

Status:  Verifying...
    Compare point OUT failed (is not equivalent)

******************************** Verification Results ********************************
Verification FAILED
-------------------
 Reference design: r:/WORK/C1
 Implementation design: i:/WORK/C2
 0 Passing compare points
 1 Failing compare points
 0 Aborted compare points
 0 Unverified compare points
--------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
--------------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0       0       0       0       0
Failing (not equivalent)       0       0       0       0       1       0       0       1
**************************************************************************************
Info:  Try the analyze_points command to see if Formality can determine potential
causes, or suggest next steps for a FAILED or INCONCLUSIVE verification.
See the man page for analyze_points usage and options.
Info:  Formality Guide Files (SVF) can improve verification success by automating setup.
0
```

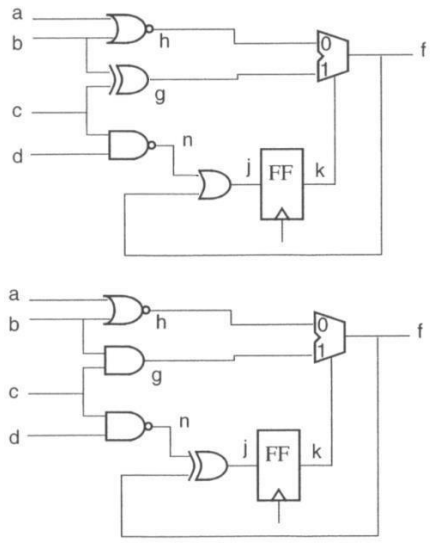
Formality Verification Process:

- Reading the designs - setting reference and implementation design
    - fm_shell (setup) > read_verilog - r
    - fm_shell (setup) > read_verilog - i
- Setup - set up constants / disable scan logic or other typical setup steps
- Match - run Formality's matching algorithms to compare points
    - fm_shell (setup) > match
- Verify - run Formality's verification algorithms
    - fm_shell (setup) > verify

Verification FAILED was returned. The implementation design C2 is not equivalent to the reference design C1. The circuits C1 and C2 are not equivalent.

Task 2: Use the formality to check if two sequential circuits are equivalent.

Task 2.1: Perform the equivalence check on the following two sequential circuits with formality.



.sv code

```
module C1(a, b, c, d, f, clk);
input a, b, c, d, clk;
output f;

wire h, g, n;
reg k, j;

assign h = ~(a | b);
assign g = (b ^ c);
assign n = ~(c & d);

assign j = (n | f);
assign f = (k) ? g : h;

always@(posedge clk) begin
#2    k <= j;
end

endmodule
```

```
module C2(a, b, c, d, f, clk);
input a, b, c, d, clk;
output f;

wire h, g, n;
reg k, j;

assign h = ~(a | b);
assign g = (b & c);
assign n = ~(c & d);

assign j = (n ^ f);
assign f = (k) ? g : h;

always@(posedge clk) begin
#2    k <= j;
end

endmodule
```

Screenshots

Read Verilog

```
Build: 4755953
Hostname: mo.ece.pdx.edu
Current time: Sun Feb  7 11:57:51 2021

Loading db file '/pkgs/synopsys/2017/formality/libraries/syn/gtech.db'
fm_shell (setup)> read_verilog -r task2.sv
No target library specified, default is WORK
Loading verilog file '/u/abfouts/582_project2/task2.sv'
Current container set to 'r'
1
fm_shell (setup)> set_top C1
Setting top design to 'r:/WORK/C1'
Status:   Elaborating design C1   ...
Status:  Implementing inferred operators...
Top design successfully set to 'r:/WORK/C1'
Reference design set to 'r:/WORK/C1'
1
```

Read Implementation

```
fm_shell (setup)> read_verilog -i task2.sv
No target library specified, default is WORK
Loading verilog file '/u/abfouts/582_project2/task2.sv'
Current container set to 'i'
1
fm_shell (setup)> set_top C2
Setting top design to 'i:/WORK/C2'
Status:   Elaborating design C2   ...
Status:  Implementing inferred operators...
Top design successfully set to 'i:/WORK/C2'
Implementation design set to 'i:/WORK/C2'
1
```

Match

```
fm_shell (setup)> match
Reference design is 'r:/WORK/C1'
Implementation design is 'i:/WORK/C2'
Status:  Checking designs...
Status:  Building verification models...
Status:  Matching...

************************************ Matching Results ************************************
 2 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 5 Matched primary inputs, black-box outputs
 0(0) Unmatched reference(implementation) compare points
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
******************************************************************************************

1
```

Verify

```
fm_shell (match)> verify
Reference design is 'r:/WORK/C1'
Implementation design is 'i:/WORK/C2'

************************************ Matching Results ************************************
 2 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 5 Matched primary inputs, black-box outputs
 0(0) Unmatched reference(implementation) compare points
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
******************************************************************************************

Status:  Verifying...
    Compare point k_reg failed (is not equivalent)
    Compare point f failed (is not equivalent)

********************************** Verification Results **********************************
Verification FAILED
-------------------
 Reference design: r:/WORK/C1
 Implementation design: i:/WORK/C2
 0 Passing compare points
 2 Failing compare points
 0 Aborted compare points
 0 Unverified compare points
------------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
------------------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0       0       0       0       0
Failing (not equivalent)       0       0       0       0       1       1       0       2
******************************************************************************************
Info:  Try the analyze_points command to see if Formality can determine potential
causes, or suggest next steps for a FAILED or INCONCLUSIVE verification.
See the man page for analyze_points usage and options.
Info:  Formality Guide Files (SVF) can improve verification success by automating setup.
0
```

Process

Based off of the description in the tutorial when Formality returns a *Verification: FAILED* or *Verification: SUCCEEDED* determines whether or not the circuits are equivalent or not. The process was to create individual modules of both *C1* and *C2* circuits, and implement both circuits into the reference and implementation sections of Formality. The .sv file is sent via the command line arguments:

1. fm_shell
2. read_verilog -r {file.sv}
3. set_top {module (C1)}
4. read_verilog -i {file.sv}
5. set_top {module (C2)}
6. match
7. verify

Each command was described in the tutorial to complete a basic verification. Since each module is stored in a single .sv file *-auto* was not used during the first *set_top*, so Formality knew to use the correct module.