Jordan Bayles
April 16, 2014

ECE 474: VLSI HW #1

fadder.sv

```
//------------------------------------------------------------------------
//one bit full adder
//------------------------------------------------------------------------
module fadder(
  input a,          //data in a
  input b,          //data in b
  input cin,        //carry in
  output sum_out,   //sum output
  output c_out      //carry output
  );
  wire c1, c2, c3; //wiring needed

  assign sum_out = a ^ b ^ cin;    //half adder (XOR gate)
  assign c1      = a * cin;        //carry condition 1
  assign c2      = b * cin;        //carry condition 2
  assign c3      = a * b;          //carry condition 3
  assign c_out   = (c1 + c2 + c3);

endmodule
```

adder8.sv

```
//------------------------------------------------------------------------
//8-bit ripple carry adder program
// author: Jordan Bayles
// module: adder8 (8-bit adder)
// course: ECE 474 (VLSI), HW #1
//   date: 4/16/2014
//------------------------------------------------------------------------

module adder8(
  input  [7:0]  a,        //data in a
  input  [7:0]  b,        //data in b
  output [7:0] sum_out,   //sum output
  output c_out            //carry output
  );

  wire [6:0] cx;

  fadder a1(
  .a          (a[0]),
  .b          (b[0]),
  .sum_out    (sum_out[0]),
  .c_out      (cx[0])
  );

```

```
 fadder a2(
.a          (a[1]),
.b          (b[1]),
.cin        (cx[0]),
.sum_out    (sum_out[1]),
.c_out      (cx[1])
);

fadder a3(
.a          (a[2]),
.b          (b[2]),
.cin        (cx[1]),
.sum_out    (sum_out[2]),
.c_out      (cx[2])
);

fadder a4(
.a          (a[3]),
.b          (b[3]),
.cin        (cx[2]),
.sum_out    (sum_out[3]),
.c_out      (cx[3])
);

fadder a5(
.a          (a[4]),
.b          (b[4]),
.cin        (cx[3]),
.sum_out    (sum_out[4]),
.c_out      (cx[4])
);

fadder a6(
.a          (a[5]),
.b          (b[5]),
.cin        (cx[4]),
.sum_out    (sum_out[5]),
.c_out      (cx[4])
);

fadder a7(
.a          (a[6]),
.b          (b[6]),
.cin        (cx[5]),
.sum_out    (sum_out[6]),
.c_out      (cx[6])
);

fadder a8(
.a          (a[7]),
.b          (b[7]),
.cin        (cx[6]),
.sum_out    (sum_out[7]),
.c_out      (c_out)
```

```
    );
80
endmodule
```

adder8.do

```
1  // Jordan Bayles
   // VLSI HW #1
3  // 4/16/2014

5  add list -nodelta
   configure list -strobestart {9 ns}  -strobeperiod {10 ns}
7  configure list -usestrobe 1

9  add list -notrigger -hex -width 4  -label a          a
   add list -notrigger -hex -width 4  -label b          b
11 add list -notrigger -hex -width 10 -label sum_out    sum_out
   add list -notrigger -hex -width 8  -label c_out      c_out
13
   // test cases
15
   //both zero
17 force a  x"0"
   force b  x"0"
19 run 10 ns

21 // one zero
   force a  x"0"
23 force b  x"1"
   run 10 ns
25
   //neither zero
27 force a  x"1"
   force b  x"1"
29 run 10 ns

31 //both positive: multiple cases
   force a  x"1"
33 force b  x"10"
   run 10 ns
35
   force a  x"2"
37 force b  x"20"
   run 10 ns
39
   force a  x"10"
41 force b  x"1"
   run 10 ns
43
   force a  x"20"
45 force b  x"2"
   run 10 ns
47
   force a  x"30"
```

```
49 force b  x"3"
   run 10 ns
51
   force a  x"40"
53 force b  x"4"
   run 10 ns
55
   force a  x"4"
57 force b  x"40"
   run 10 ns
59
   force a  x"3"
61 force b  x"30"
   run 10 ns
63
   force a  x"50"
65 force b  x"5"
   run 10 ns
67
   force a  x"5"
69 force b  x"50"
   run 10 ns
71
   force a  x"60"
73 force b  x"6"
   run 10 ns
75
   force a  x"6"
77 force b  x"60"
   run 10 ns
79
   force a  x"70"
81 force b  x"7"
   run 10 ns
83
   force a  x"7"
85 force b  x"70"
   run 10 ns
87
   force a  x"80"
89 force b  x"8"
   run 10 ns
91
   force a  x"8"
93 force b  x"80"
   run 10 ns
95
   force a  x"5"
97 force b  x"5"
   run 10 ns
99
   force a  x"50"
101 force b  x"50"
   run 10 ns
```

```
103
force a  x"2"
105 force b  x"22"
run 10 ns
107
force a  x"22"
109 force b  x"22"
run 10 ns
111
write list adder8.list
```

syn_adder8

```
read_sverilog adder8.sv
2 compile
report_timing
4 report_area
write -format verilog -hierarchy -output adder8.gate.v
6 quit!
```

doit.sh

```
#!/bin/bash
2 # Jordan Bayles
# VLSI HW #1
4 # 4/16/2014

6 # create the work directory if it doesn't exist
if [ ! -d "work" ] ; then
8   echo "work does not exist, making it"
    vlib work
10 fi

12 # compile fadder.sv and adder8.sv if they exist
if [ -s "fadder.sv" ] ; then
14   vlog -novopt fadder.sv
fi
16
if [ -s "adder8.sv" ] ; then
18   vlog -novopt adder8.sv
fi
20
# first run of the simulation
22 if [ -s "adder.do" ] ; then
    vsim adder8 -do adder8.do -quiet -c -t 1ps
24 fi


26
# synthesize adder8
28 if [ -s "syn_adder8" ] ; then
    dc_shell-xg-t -f syn_adder8
30 fi

32 # compile the gate library if it hasn't been done yet
```

```
if [ -s `grep adder8 work/_info/*` ] ; then
34
fi
36
# compile the .gate.v file
38 if [ -s "adder8.gate.v" ] then
    vlog -novopt adder8.gate.v
40 fi

42 # second run of the simulation
vsim adder8 -do adder8.do -quiet -c -t 1ps
44
# compare results of the two runs of simulation
46
# user message indicating if comparison failed
```