Hasson Qayum

CMPE 100L

Lab Section 3

3 June 2016

Lab 7 Report

Description: In this lab, we were to make a game with the VGA port of the BASYS 2 board. The game we created is a slug (a blue rectangular bar) that moves left and right within the active region of a computer screen. It toggles off the walls of the borders and moves the other direction. The way to control the slug is using the BTN2 push button on the BASYS 2 board. When the game is in it's Idle state, you will see the blue slug moving left and right, bouncing off the walls. We create a yellow border around the screen that is 8 pixels wide to show that the slug bounces a pixel off from the border, and to see the meteors fall into and off the screen. There are two meteors that we must create, a red colored one with a pattern and a green one. In Idle state, you will see the border, slug moving and a red meteor in the center of the screen. The meteor is shaped like a diamond. It is 128 pixels wide (64 pixels wide from the center to the corners of the diamond). Once we start the game, we must dodge the meteors. Every meteor dodges increments our score by one. When we dodge 10 meteors, we increase in level. The way you know u went up a level correctly is by seeing the game stop, the slug flash magenta, then flash blue and get bigger and the level score increments one. We pick how many levels by using the switches. When the level matches the switches, the slug and meteor flash saying you won the game (Victory Dance). You lose when u hit a meteor. When you lose, your slug flashes as the game is stopped. The way to return to Idle from winning or losing is by pressing BTN1 to restart the

game, or by pressing BTN3 which is a hard reset and resets the entire game, then goes to Idle, which takes more time. From this information, you can play the game correctly.

Methods: The way I went on with this lab was by first making my yellow border. I set the bounds of the active region and HS and VS according to the lab manual. After that, I tried to create my slug and have it bounce which was really hard. After two days of working on that, and from getting help from some friends, I figured that I must use a toggle flip flop and edge detect my push button. I used a counter to move the slug left and right (1 frame per second), and to toggle the slug, I used the toggle flip flop and I set the bounds at which the slug is suppose to bounce at. After that, I started working on the meteors. In piazza, the professor uploaded this equation to make the diamond shape. The equation was: | U - X | + | V - Y | < 64. I remember her saying that it sometimes gets messed up if you subtract in verilog so I went threw the arithmetic symbols and found an adder and subtracter. The way I did this was use my horizontal counter (U) and subtracting that with a constant. Same with the vertical counter (V). Those outputs went into my sign changer which made their values positive. Then I added their values. To get the design, I used the COMPM16 comparator and set bounds with the less than symbol and made it start in the center bright and it gets lighter as it goes out. From this, I got my red and green diamond and their pattern, now I have to get them to fall and in random places. I made an LFSR (same as lab 5) and had that go into the register. I took off the constants from the subtractor symbols and plugged in the output of the register into (X) to get the (U - X) part. I used a counter to have it fall down and that output was used as the (Y) in the (V - Y) part. In order to keep the meteors in the screen, I had to create a verilog module that keeps it in bounds, and I had another verilog

module that makes the meteor fall at 2 frames per second. The last verilog module just tells the counter when the diamond when it falls off the screen so it can reset. The only difference between my red and green meteor is that my red meteor has a 2 to 1 mux connected to the counter. When the game is in Idle, the counter puts the position of the red meteor in the center of the screen, other than that, it falls from -64 (FFC0). Then I started my Finite State Machine. I had to make 3 of them to get the right one. I used LEDs to find what state and outputs were working. I made the time counter similar to older labs, same with the hex7seg, sign changer, full adder, LFSR, Ring Counter, Edge detector, and Synchronizer. The score counter was new to me because we used the CD counter but I got it down after a while. The thing I didn't know was what my CE would be but I found out it was the AND of my four second timer and when my FSM is at it's level up state. I made my collision logic with and gates so when they were both 1, the FSM would know there was a collision. I had a schematic called Color Logic which dealt with all the logic for the solid and flashing colors. All that stuff was outputted into my TOP TOP and ibuf/obufed their.

Results: My design had a lot of errors which delayed me from finishing. One thing I kept doing was misspelling things so when I thought something was connected, it wasn't because of the difference in spelling. This messed me up a lot because I thought it was connected so I thought my logic was wrong. That was really frustrating. Other than that, my design was nice. I connected all the things needed to where they go and compiled and it worked. The way I tested my FSM was with LEDs. I tested my symbols by just playing the game even though the score or anything wasn't connected. The only errors I got from testing was FSM errors. My LEDs weren't

lighting when I wanted them to so I tried to debug but it was hard to find the error, so I thought it was my state diagram so I redrew that. The second one had similar problems because the diagrams were really similar, so then I just remade it one more time and that one worked well. The major components in this lab in my opinion were the borders verilog, slug logic, and meteor logic. I feel like that was the meat of this assignment. They all interested though the color logic schematic. The VGA signals played the role in connecting all these components together. The pins I used in this lab were:

Input	**Pins**
PB0	G12
PB1	C11
PB2	M4
CA	L14
CB	H12
CC	N14
CD	N11
CE	P12
CF	L13
CG	M12
SW0	P11
SW1	L3
SW2	K3
SW3	B4
HS	J14
VS	K13
RED0	C14
RED1	D13
RED2	F13

GREEN0	F14
GREEN1	G13
GREEN2	G14
BLUE0	H13
BLUE1	J13
DP	N13
AN0	F12
AN1	J12
AN2	M13
AN3	K14
SW7	N3
PB3	A7
LED7	G1

Conclusion: From this lab, I learned how a lot about the VGA outputs and how to use logic to control pixels on a connected computer through BASYS 2 VGA port. The logic and verilog was basically the same, the only thing that was different was accounting for the 3 red, 3 green, and 2 blue colors, as well as the HSync and VSync. Other than that, I felt like everything was the same. I had a lot of difficulties in this lab. Some that I explained earlier like to get the slug to toggle, or to making the meteors fall within the borders randomly was truly a pain. Another error I had that was very strange was with my selector. I tried showing the TAs/tutors on Tuesday this week in lab but nobody knew how to solve this error. I had a 4 bit bus of an M4_1E mux. I fed it the inputs and everything and when I compiled, it would show no errors. When I tried to generate the bit file, it would pass synthesize and Implement design, but once it got to trying to generate the bit file, the error would pop up. I posted the error on piazza on Tuesday when the tutors told me

too cause they didn’t know what was wrong. I also sent a message to the Professor because I was in lab the whole day and couldn’t get help from anybody on how to fix it. The next day I met with the professor and we fixed the problem and it worked. From asking the professor, the error was that an input was grounded and that would mess the m4_1E mux up. She made me create the 4 to 1 mux by using three 2 to 1 muxes and then it worked. The error didn’t explain well what was wrong and that is why I think me or the tutors couldn’t find the error. That took me like 2 days to solve. Plus the toggle of the slug and making the meteors fall randomly in the borders was the problems I dealt with the longest. If I could do this lab over again, I would make sure to name stuff easier so I know what output is going to what input, and to make sure the names were correct. A problem I had was having the slug to go every level. I checked my logic and verilog and I was sure it was right. After some hours of looking and trying to find why it wasn’t working, I found that the inputs name was misspelled from the output I needed. Once I fixed this, it worked so that was a pain in the butt. I now know to make sure all my names are correct and spelled correctly, and to sometimes fix warning because they may lead to future errors. I believe my components are optimized but I am not fully sure about that. I didn’t ask students, TAs, tutors, or the teacher so I am not sure. In the end, this lab was great and I am happy I was able to create it, but it was extremely stressful and the most time consuming thing I have ever done. I am so happy it is over with and I don't have to stress about this anymore, but I am happy we did this lab because it is really cool knowing that we made a game like this.

Appendix:

BTN0 - G12

BTN1 - C11

BTN2 - M4

10 VGA obufs

CA - L14
CB - H12
CC - N14
CD - N11
CE - P12
CF - L13
CG - M12

SW0 - P11
SW1 - L3
SW2 - K3
SW3 - B4

HS - J14

VS - K13

LAB 7

*DP - N13
AN0 - F12
AN1 - J12
*AN2 - M13
AN3 - K14

Idle - Meter back and forth / Meteor Not in motion

BTN0 - Start game

BTN2 - Change direction

Every meteor dodged is 1 point

Every 10 point AN3 goes up 1 (level), Slug gets bigger by 16 pixels. Flasher Magenta then goes back to blue.

Slug moves 1 pixel per frame

Meteor moves 2 pixels per frame.

When Meteor hits slug, meteors freeze and slug flashes until BTN1 is pressed, then back to Idle.

When win game, flash meteor and slug, until BTN1 pressed, then back to Idle.

Switches correspond to levels, if 2 switches up, 2 levels then game is over.

RED0 - C14
RED1 - D13
RED2 - F13

GO - F14
G1 - G13
G2 - G14

BO - H13
B1 - J13

Once half meteor gone, other meteor falls.

& 6 bit LFSR determines horizontal coordinates of meteor.

Increment when meteor is all off from bottom.

$(R_2, R_1, R_0, G_2, G_1, G_0, B_1, B_0)$

x-axis 1) →

y-axis 2) ↓

HS → 0 to 655, 750 - 799

VS → 0 to 489, 490 - 524

Red-green = Yellow

Red - Blue = Magenta

$799 = 0000001100011111$ → H - high

$524 = 0000001000001100$ → V - high

$479 = 0000000111011111$ → AR-V

$639 = 0000001001111111$ → AR_H

Slug initial = 16 x 64

H high = frame = 1 p/s

320 (dec) → 0140 (hex)

240 (dec) → F0 (hex)

272 → 0000000100010000

16x lev AN2

15 — 9 L

L(7) high

16

L(7) high L

~~How to move? Sign Change for meteor?~~

How to move Slug and meteor? ✓

How to make meteor? ✓

How to bounce Slug? ✓

64 frame ≈ 1 second

−64 → FFC0

How to make meteor move? ✓

Alternate from Red to Green ✓

Fall, reset, change color, fall (meteor falls after last m half m)

Center pixel (x, y)

$|U - x| + |V - y| < 64$

Counter +2

111 → ~~C04~~ C16 (0)

110

101 → C32 (1)

100

011 → C48 (2)

010

001 → C64 (3)

512 → (9)

240

480

525

0

-64

608

Text Box

From

480

Alternate Red & green ✓

Meteor movement ✓

Random Meteors

Collision detection

1 2 3

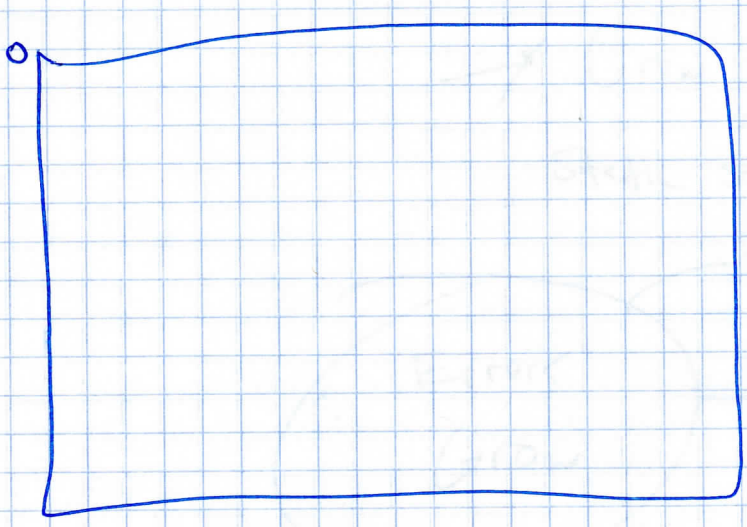
3 2 1

302

1 2 3

3 2 1 0

-64



pix_cont

Random Stuf

Random Meteors

Collision Detection

Slug Flash

Meteor Flash

Green Met

Met speed

Test Bench

Not growing. 1066

toggle Red Green → Center

Stay in Bounder ✓

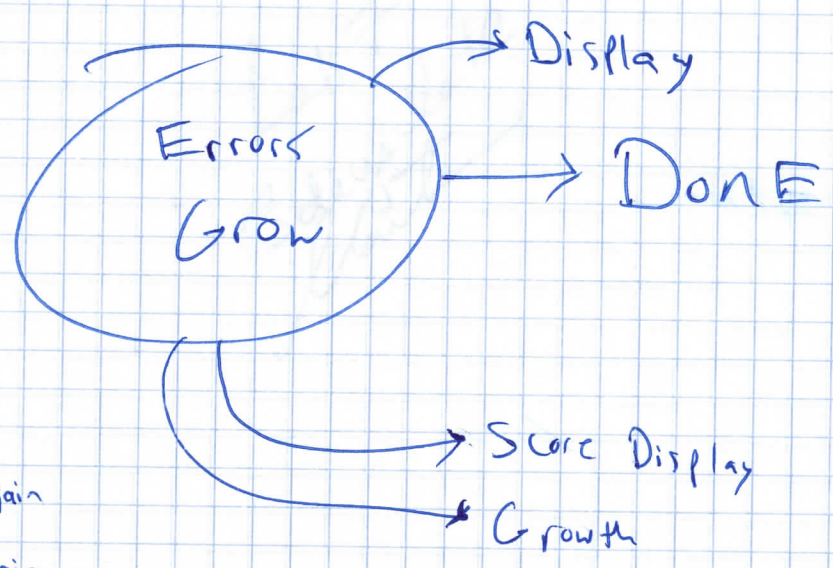
LFSR ✓

Error
→ Grow

640 → 320

480 ↓ 240

Static start ✓



Error ✓
Grow
Center

Slug length → Check again

T_ Count → Check again

Selector

Score Display

flash

Error fixed

	AN3	2	1	0	A	B
Score (7:4)	0	0	0	1	0	0
Score (3:0)	0	0	1	0	0	1
gD	0	1	0	0	1	0
Level (3:0)	1	0	0	0	1	1

Inc Score

Poor_time

1 00

level starts off at 1

ever in Idle

Lab 7 Stuff

Inputs:

PB0 (start game)

Four Secs

PB1 (return to idle after lose)

Collision (meteor & slug)

SW Level (1 - F)

Diamond_Gone

Level_Up

Match (sw & level)

Start

Restart

Four Sec1

Collision

Dia_In

Meteor_TEN, Match

Outputs:

Local (Random Meteors)

Level_Inc (start at 1) (ever

Flash_Slug (every 10 magenta)

Increase_Slug (blue plus 16 pix

Score_Inc

Win (Flash Slug and Meteor (Vict

Lose (Flash Slug)

Reset Time

Reset

Victory Dance

Magenta

Score_Inc

Slug_Stop

Flas

RED_M

Dia_

level starts at 1

every meteor gone off is Inc

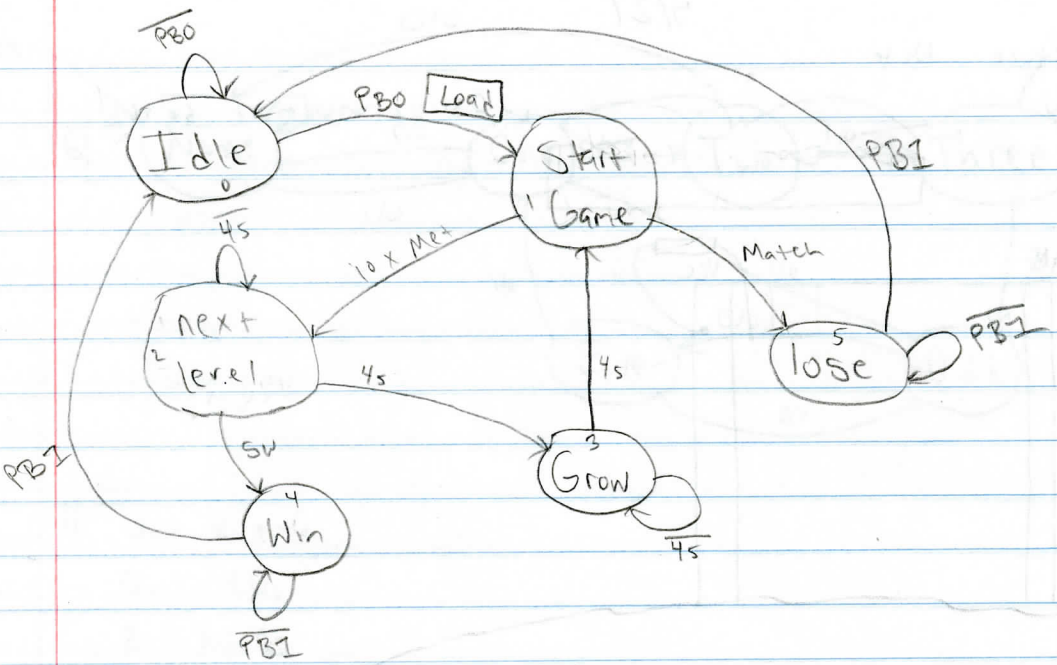
every 10 is Inc level, flash magenta, grow & Inc level & flash sl

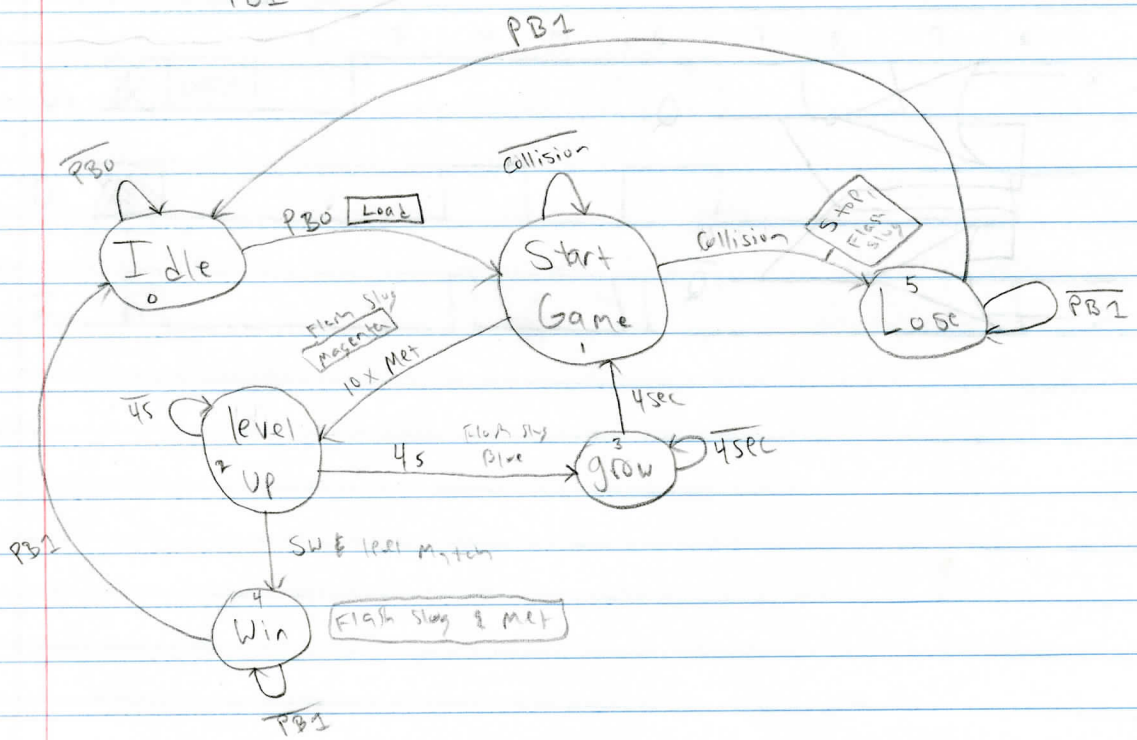
Keep playing after last score.

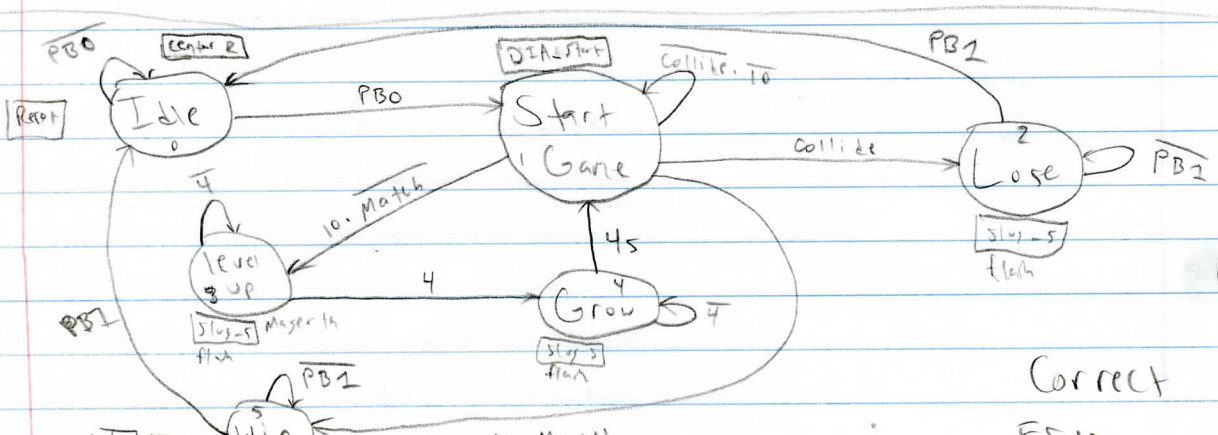
When lose, meteor stop and slug reached, PB1 to return to Idle

When either match level, flash meteor/Slug, PB1 to return to Idle,

grow → 16 pixels

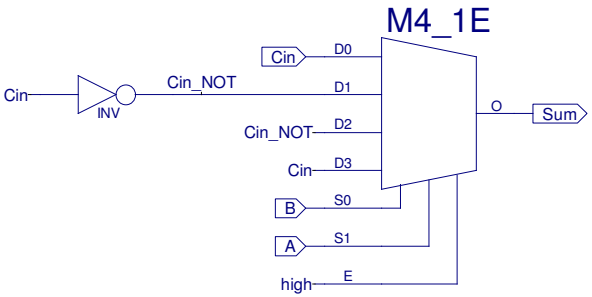


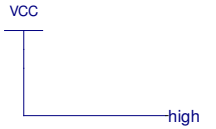


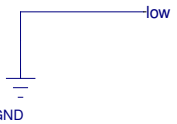


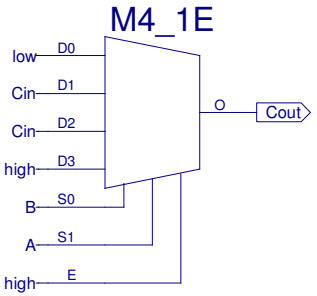
Correct

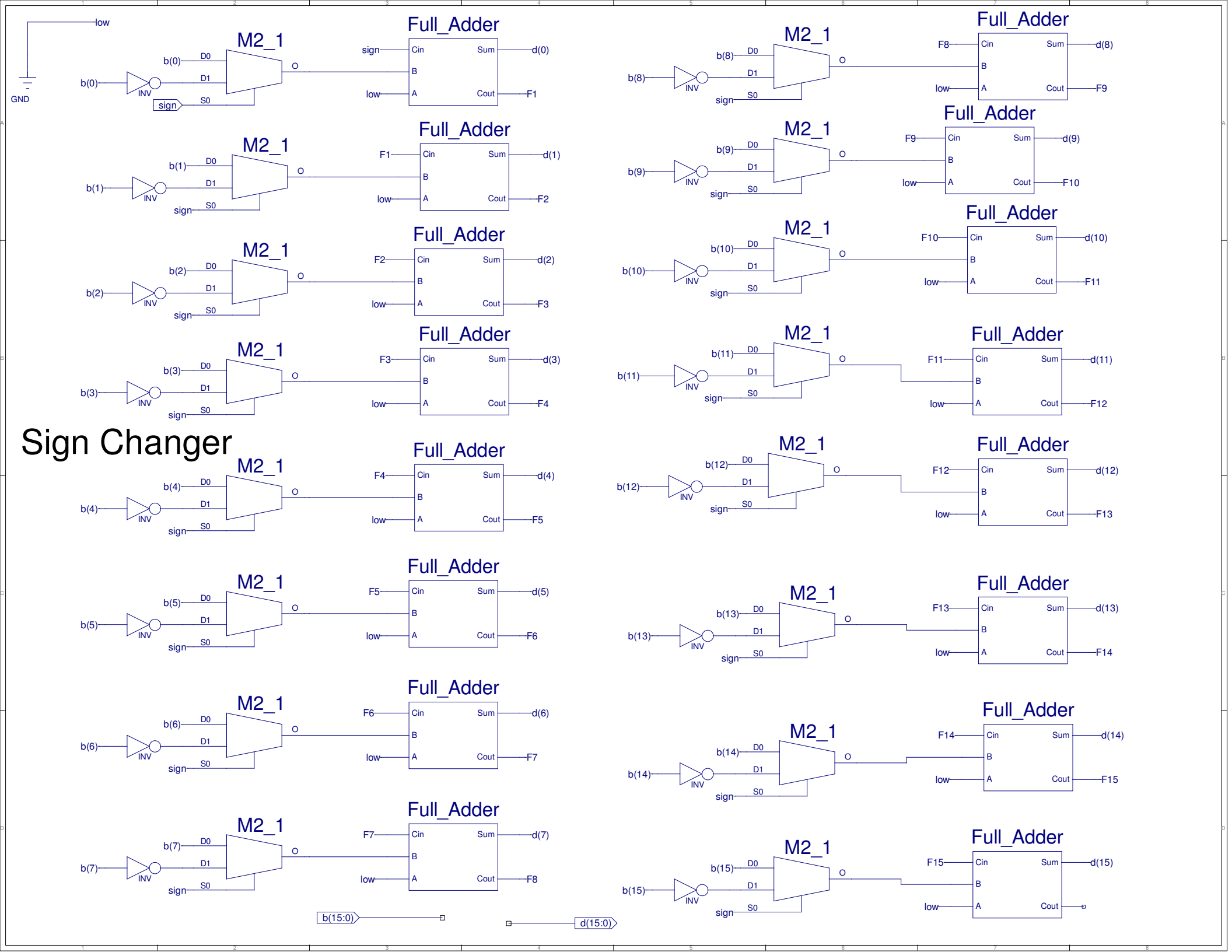
Full Adder

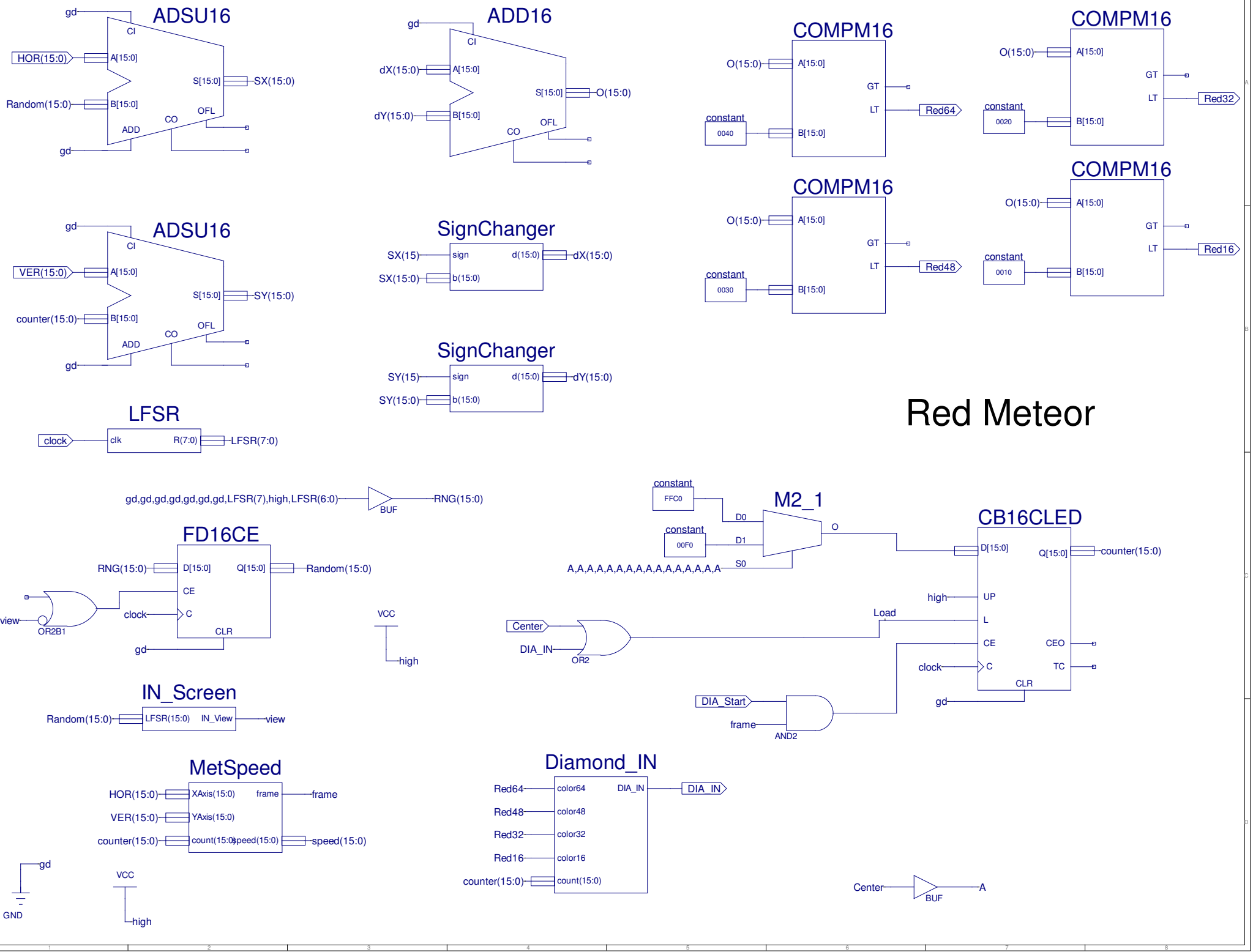

M4_1E
Cin
Cin_NOT
INV
D0
D1
D2
D3
S0
S1
E
O
Sum
Cin_NOT
Cin
B
A
high

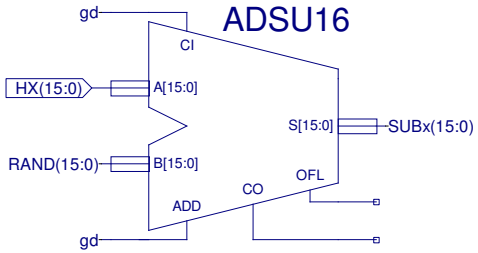

VCC
high

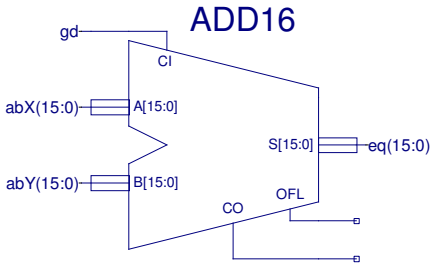

low
GND

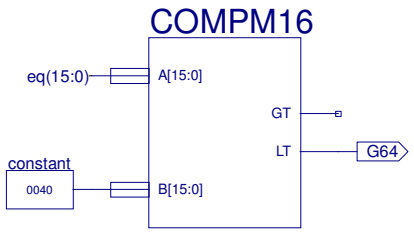

M4_1E
low
Cin
Cin
high
B
A
high
D0
D1
D2
D3
S0
S1
E
O
Cout

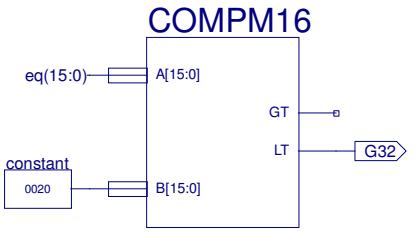

Sign Changer
low
GND
M2_1
Full_Adder
b(0)
b(1)
b(2)
b(3)
b(4)
b(5)
b(6)
b(7)
b(8)
b(9)
b(10)
b(11)
b(12)
b(13)
b(14)
b(15)
D0
D1
S0
O
INV
sign
Cin
Sum
B
A
Cout
low
d(0)
d(1)
d(2)
d(3)
d(4)
d(5)
d(6)
d(7)
d(8)
d(9)
d(10)
d(11)
d(12)
d(13)
d(14)
d(15)
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15
b(15:0)
d(15:0)

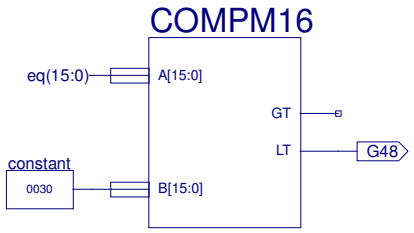

ADSU16
gd
CI
HOR(15:0)
A[15:0]
S[15:0]
SX(15:0)
Random(15:0)
B[15:0]
OFL
CO
ADD
gd
ADD16
gd
CI
dX(15:0)
A[15:0]
S[15:0]
O(15:0)
dY(15:0)
B[15:0]
OFL
CO
COMPM16
O(15:0)
A[15:0]
GT
LT
Red64
constant
0040
B[15:0]
COMPM16
O(15:0)
A[15:0]
GT
LT
Red32
constant
0020
B[15:0]
ADSU16
gd
CI
VER(15:0)
A[15:0]
S[15:0]
SY(15:0)
counter(15:0)
B[15:0]
OFL
CO
ADD
gd
SignChanger
SX(15)
sign
d(15:0)
dX(15:0)
SX(15:0)
b(15:0)
SignChanger
SY(15)
sign
d(15:0)
dY(15:0)
SY(15:0)
b(15:0)
COMPM16
O(15:0)
A[15:0]
GT
LT
Red48
constant
0030
B[15:0]
COMPM16
O(15:0)
A[15:0]
GT
LT
Red16
constant
0010
B[15:0]
LFSR
clock
clk
R(7:0)
LFSR(7:0)
Red Meteor
gd,gd,gd,gd,gd,gd,gd,LFSR(7),high,LFSR(6:0)
BUF
RNG(15:0)
constant
FFC0
M2_1
D0
O
constant
00F0
D1
A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A
S0
CB16CLED
D[15:0]
Q[15:0]
counter(15:0)
high
UP
Load
L
Center
DIA_IN
OR2
CE
CEO
clock
C
TC
CLR
gd
DIA_Start
frame
AND2
FD16CE
RNG(15:0)
D[15:0]
Q[15:0]
Random(15:0)
CE
view
clock
C
OR2B1
CLR
gd
VCC
high
IN_Screen
Random(15:0)
LFSR(15:0)
IN_View
view
MetSpeed
HOR(15:0)
XAxis(15:0)
frame
frame
VER(15:0)
YAxis(15:0)
counter(15:0)
count(15:0)
speed(15:0)
speed(15:0)
gd
GND
VCC
high
Diamond_IN
Red64
color64
DIA_IN
DIA_IN
Red48
color48
Red32
color32
Red16
color16
counter(15:0)
count(15:0)
Center
BUF
A

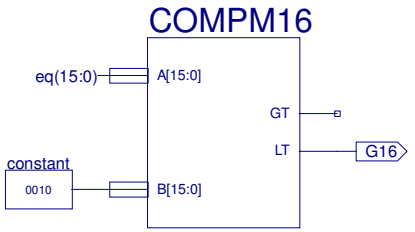


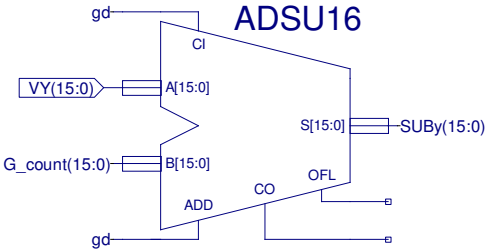


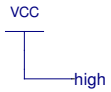


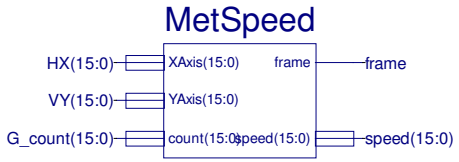


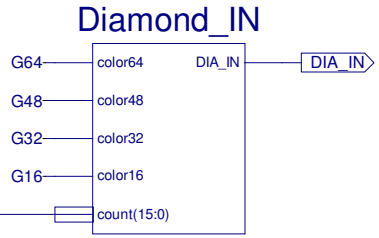


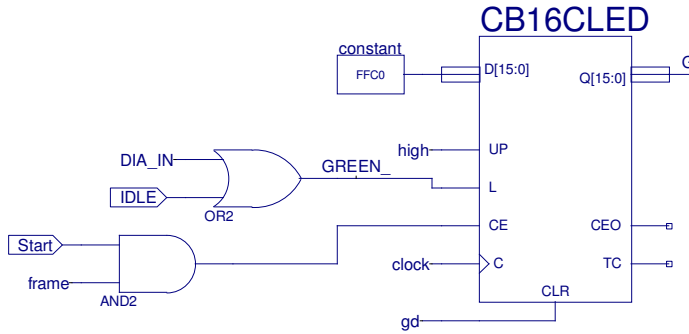


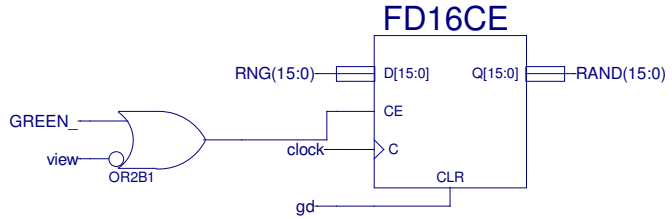












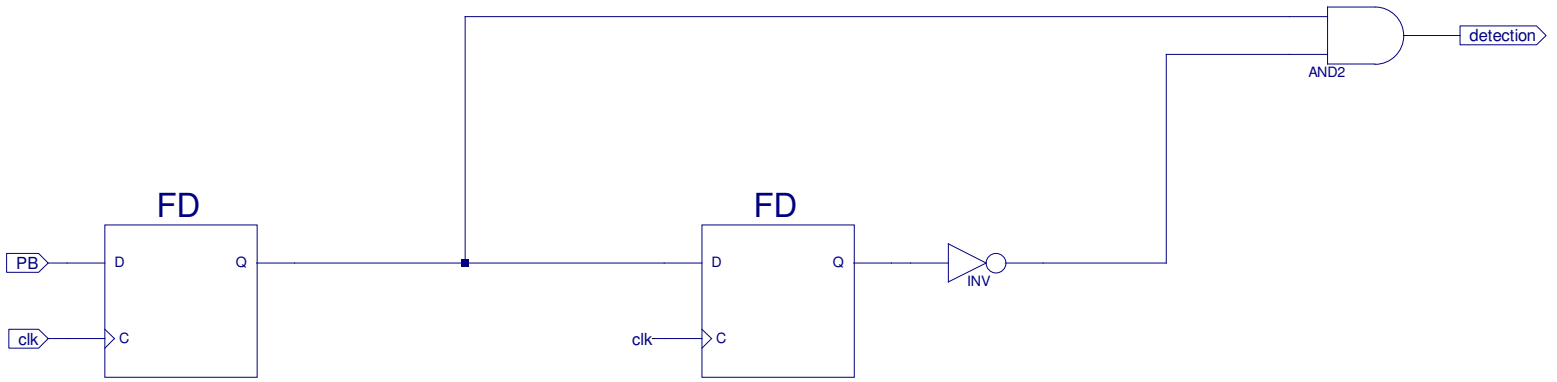
Green Meteor

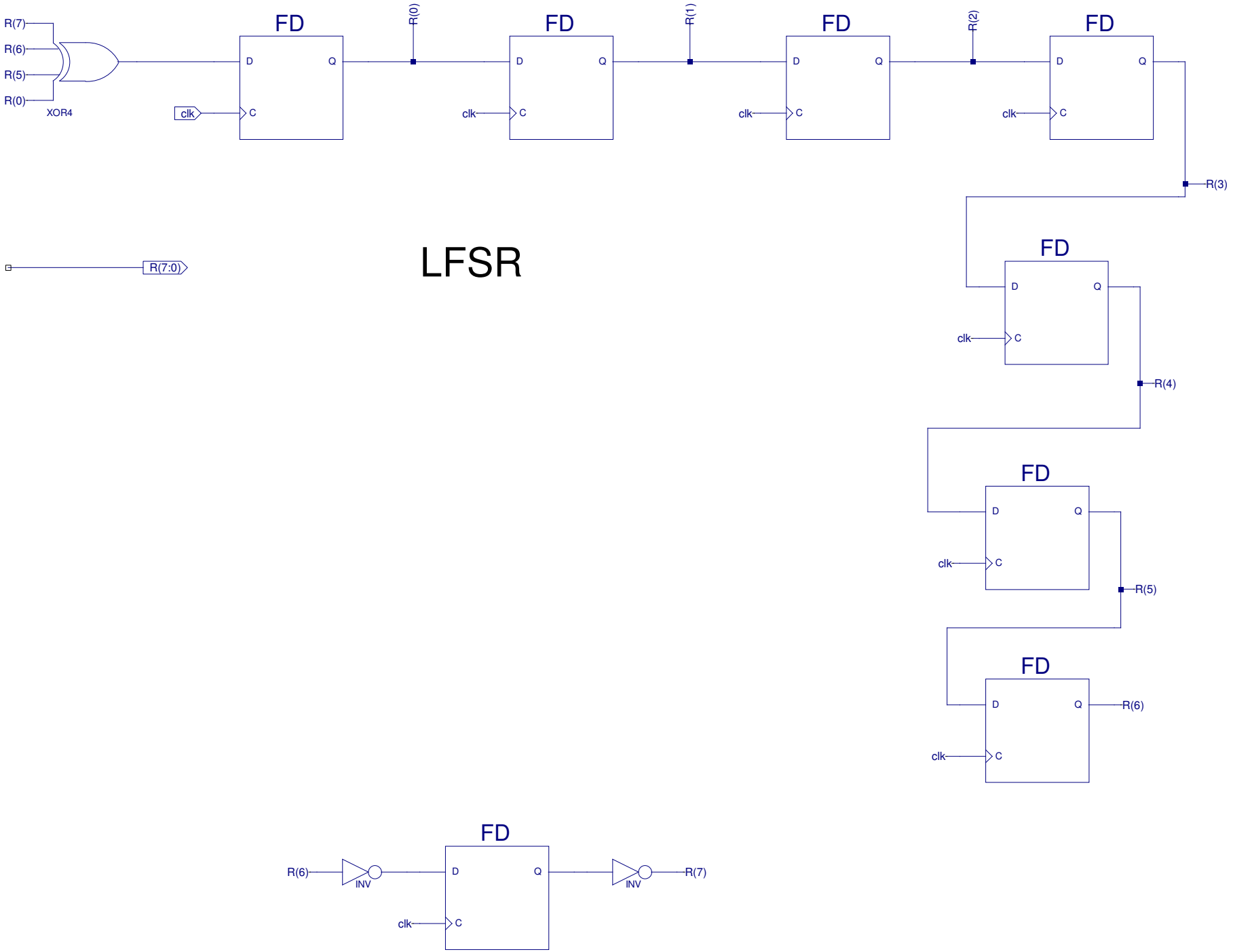
```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date:    19:38:42 05/30/2016
7   // Design Name:
8   // Module Name:    IN_Screen
9   // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21  module IN_Screen(LFSR, IN_View);
22
23  input [15:0] LFSR;
24
25  output IN_View;
26
27     //assign IN_View = (520 >= LFSR >= 150);
28     assign IN_View = ((LFSR <= 520) & (LFSR >= 150));
29
30  endmodule
31
```

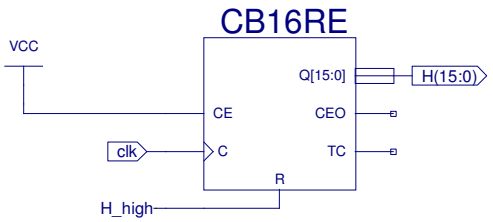
```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
    /////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date:    19:32:05 05/30/2016
7   // Design Name:
8   // Module Name:    Diamond_IN
9   // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////
    /////////////////
21  module Diamond_IN(count, color64, color48, color32, color16,
    DIA_IN);
22
23  input [15:0] count;
24  input color64;
25  input color48;
26  input color32;
27  input color16;
28
29  output DIA_IN;
30
31  assign DIA_IN = (color64 & color48 & color32 & color16 & (count
    ==530));
32
33
34  endmodule
35
```

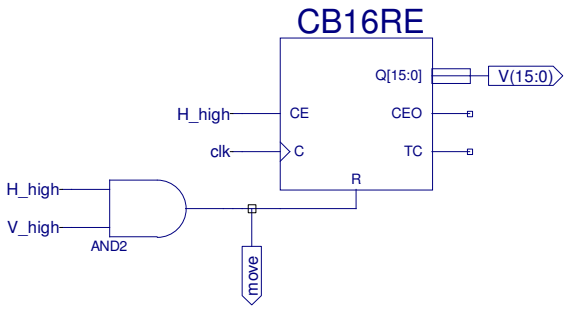
```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
    /////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date:    19:16:56 05/28/2016
7   // Design Name:
8   // Module Name:    MetSpeed
9   // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////
    /////////////////
21  module MetSpeed(XAxis, YAxis, count, frame, speed);
22
23     input [15:0] XAxis;
24     input [15:0] YAxis;
25     input [15:0] count;
26
27     output frame;
28     output [15:0] speed;
29
30     assign frame = ((XAxis==480 & YAxis==480) | (XAxis==480 &
    YAxis==481));
31     assign speed = (count + count);
32
33  endmodule
34
```

Edge Detector

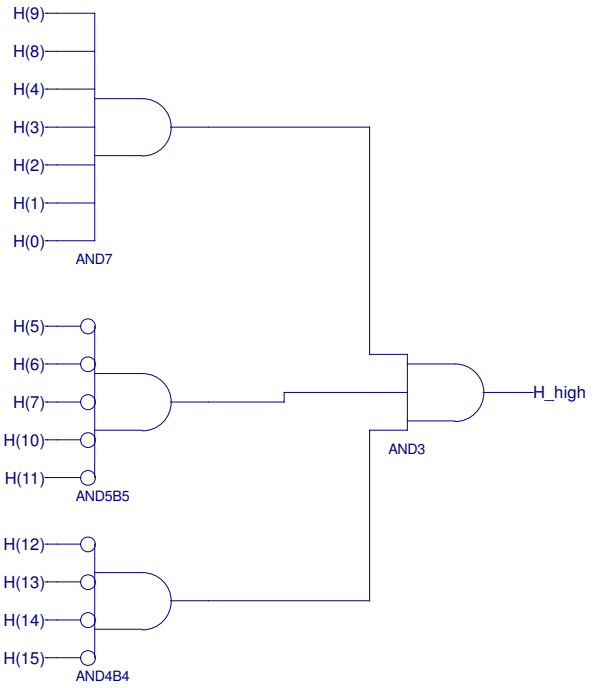


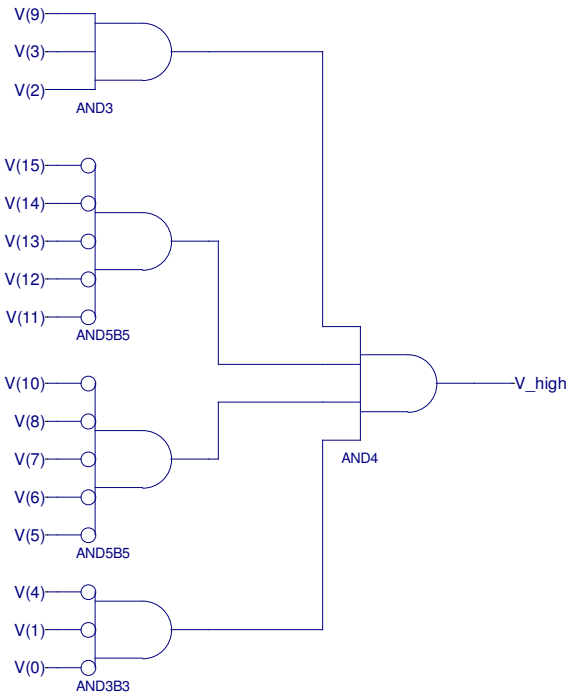

R(7)
R(6)
R(5)
R(0)
XOR4
FD
D
Q
C
clk
R(0)
R(1)
R(2)
R(3)
R(4)
R(5)
R(6)
R(7)
R(7:0)
LFSR
INV

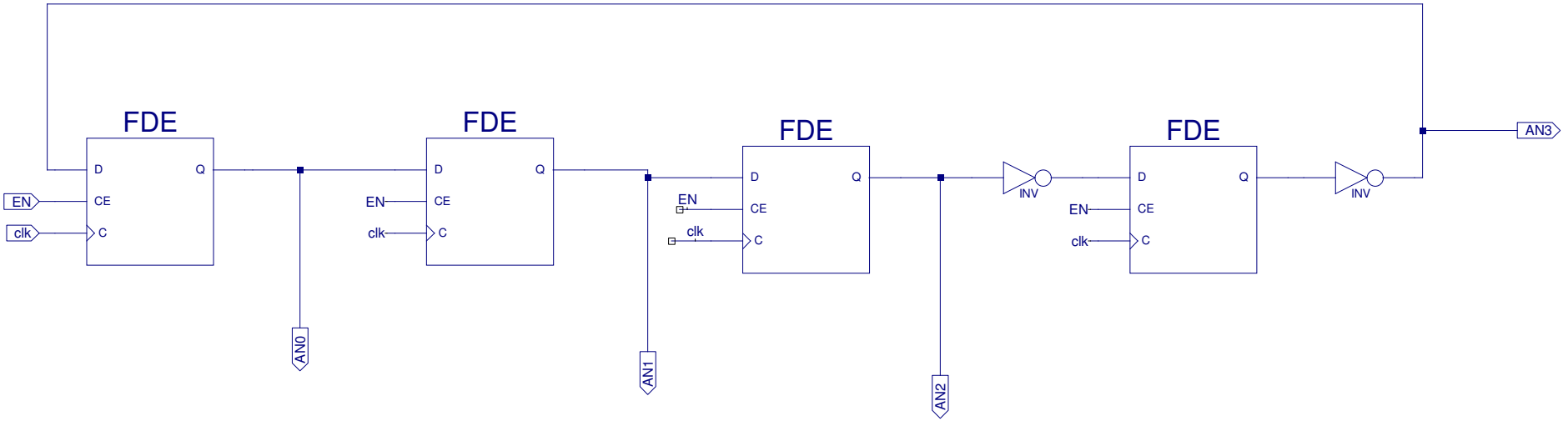




Pixal Counter

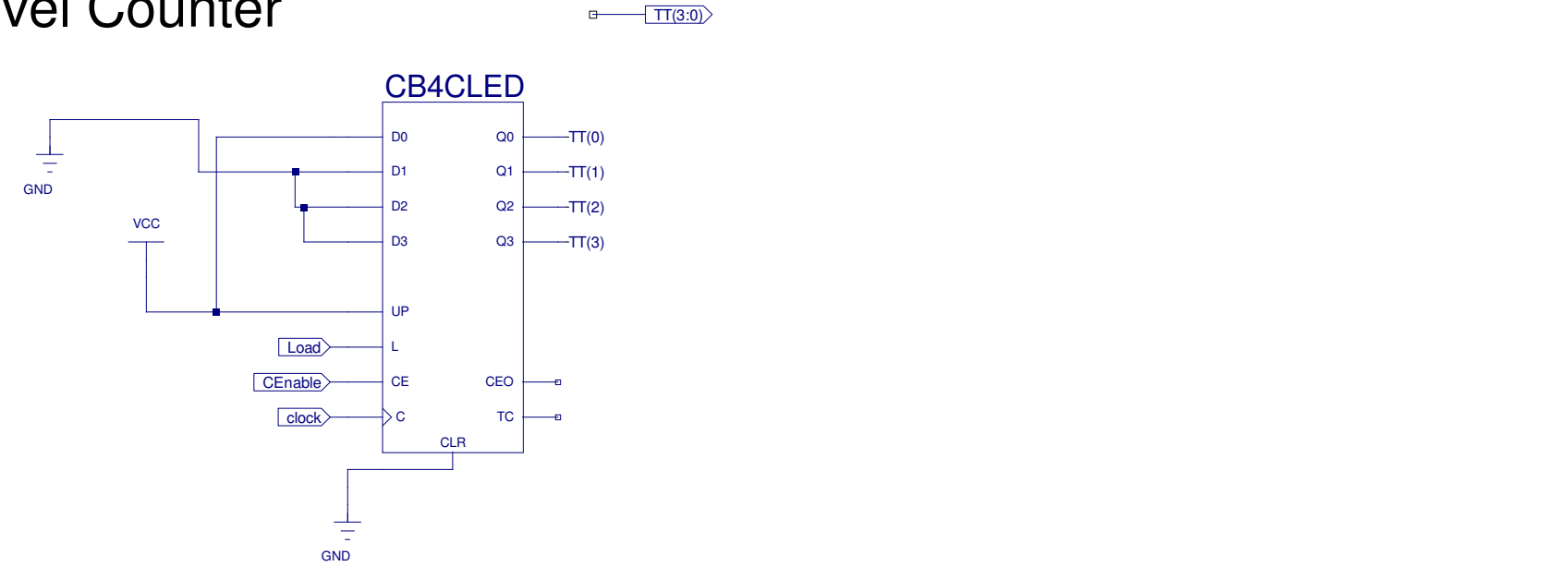






Ring Counter

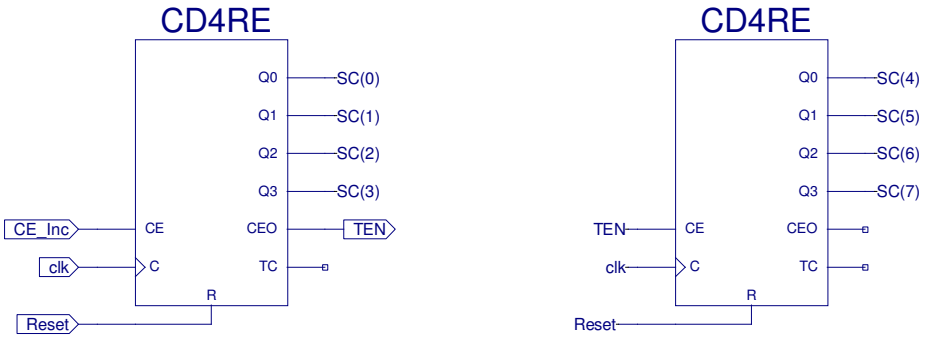
Level Counter



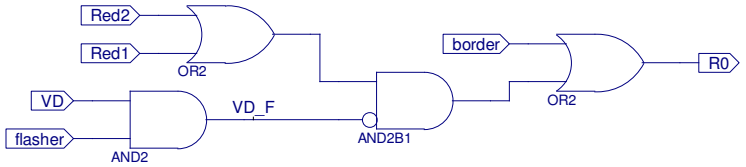
```
 1  `timescale 1ns / 1ps
 2  //////////////////////////////////////////////////////////////////
    /////////////////
 3  // Company:
 4  // Engineer:
 5  //
 6  // Create Date:    12:41:30 05/19/2016
 7  // Design Name:
 8  // Module Name:    hex7seg
 9  // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////
    /////////////////
21  module hex7seg(s3, s2, s1, s0, CA, CB, CC, CD, CE, CF, CG);
22  input s3, s2, s1, s0;
23  output CA, CB, CC, CD, CE, CF, CG;
24
25  assign CA = (s3|s2|s1|~s0)&(s3|~s2|s1|s0)&(~s3|s2|~s1|~s0)&(~s3
    |~s2|s1|~s0);
26  assign CB = (s3|~s2|s1|~s0)&(s3|~s2|~s1|s0)&(~s3|s2|~s1|~s0)&(~
    s3|~s2|s1|s0)&(~s3|~s2|~s1|s0)&(~s3|~s2|~s1|~s0);
27  assign CC = (s3|s2|~s1|s0)&(~s3|~s2|s1|s0)&(~s3|~s2|~s1|s0)&(~s3
    |~s2|~s1|~s0);
28  assign CD = (s3|s2|s1|~s0)&(s3|~s2|s1|s0)&(s3|~s2|~s1|~s0)&(~s3|
    s2|~s1|s0)&(~s3|~s2|~s1|~s0);
29  assign CE = (s3|s2|s1|~s0)&(s3|s2|~s1|~s0)&(s3|~s2|s1|s0)&(s3|~
    s2|s1|~s0)&(s3|~s2|~s1|~s0)&(~s3|s2|s1|~s0);
30  assign CF = (s3|s2|s1|~s0)&(s3|s2|~s1|s0)&(s3|s2|~s1|~s0)&(s3|~
    s2|~s1|~s0)&(~s3|~s2|s1|~s0);
31  assign CG = (s3|s2|s1|s0)&(s3|s2|s1|~s0)&(s3|~s2|~s1|~s0)&(~s3|~
    s2|s1|s0);
32
33  endmodule
```

SC(7:0)

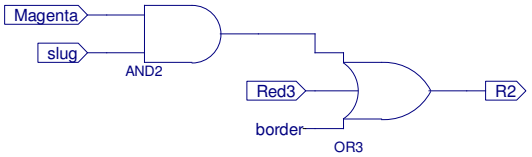
Score Counter

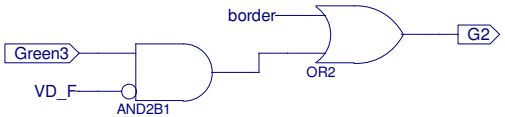


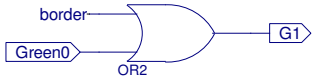
Color Logic

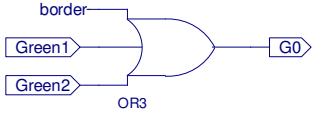

Red2
Red1
OR2
VD
flasher
AND2
VD_F
AND2B1
border
OR2
R0


border
Red0
OR2
R1

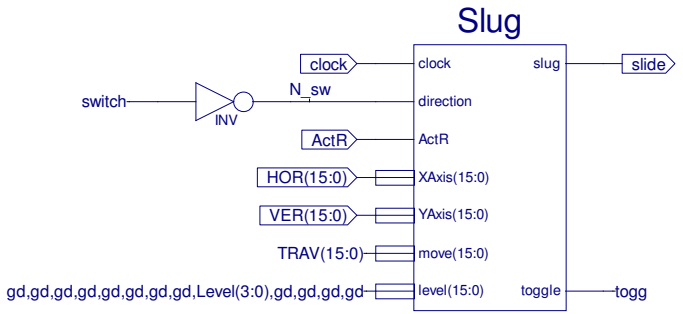

Magenta
slug
AND2
Red3
border
OR3
R2

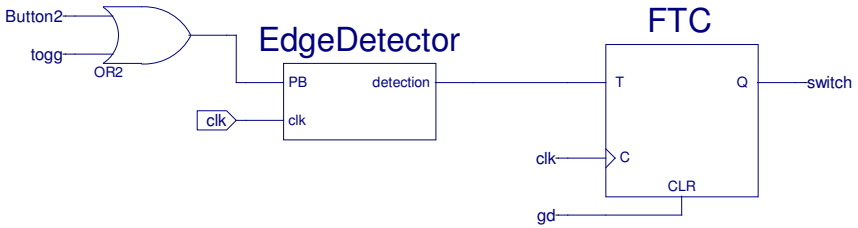

border
Green3
VD_F
AND2B1
OR2
G2

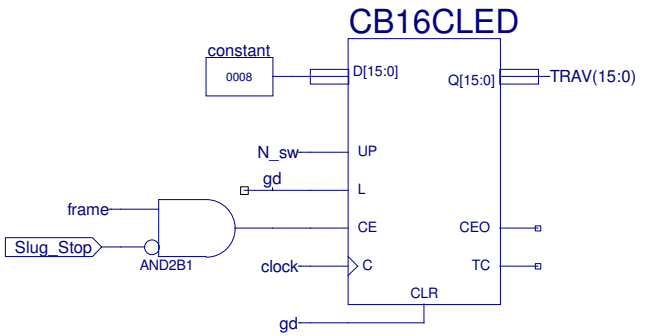

border
Green0
OR2
G1

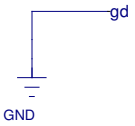

border
Green1
Green2
OR3
G0

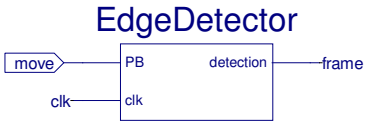
Slug Logic


Slug
clock
switch
INV
N_sw
ActR
HOR(15:0)
VER(15:0)
TRAV(15:0)
gd,gd,gd,gd,gd,gd,gd,gd,Level(3:0),gd,gd,gd,gd
clock
direction
ActR
XAxis(15:0)
YAxis(15:0)
move(15:0)
level(15:0)
slug
toggle
slide
togg

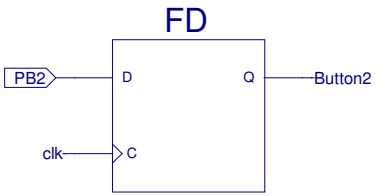

Button2
togg
OR2
EdgeDetector
PB
clk
detection
clk
FTC
T
Q
C
CLR
switch
clk
gd

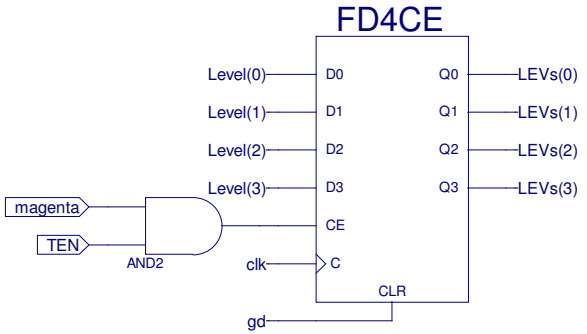

CB16CLED
constant
0008
D[15:0]
Q[15:0]
TRAV(15:0)
N_sw
gd
frame
Slug_Stop
AND2B1
UP
L
CE
CEO
clock
C
TC
CLR
gd


gd
GND


EdgeDetector
move
clk
PB
clk
detection
frame


LEVs(3:0)


FD
PB2
clk
D
C
Q
Button2


FD4CE
Level(0)
Level(1)
Level(2)
Level(3)
magenta
TEN
AND2
clk
D0
D1
D2
D3
CE
C
CLR
gd
Q0
Q1
Q2
Q3
LEVs(0)
LEVs(1)
LEVs(2)
LEVs(3)


Level(3:0)

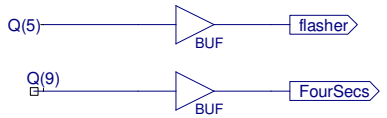
```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
    /////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date:    09:45:51 05/19/2016
7   // Design Name:
8   // Module Name:    Borders
9   // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////
    /////////////////
21  module Borders(XAxis, YAxis, Clock, HS, VS, border, ActR);
22
23  input [15:0]XAxis;
24  input [15:0]YAxis;
25  input Clock;
26
27  output HS;
28  output VS;
29  output border;
30  output ActR;
31
32
33  assign HS = (XAxis < 655) | (XAxis > 750);
34  assign VS = (YAxis < 489) | (YAxis > 490);
35  assign ActR = (XAxis >= 0 & XAxis <= 639) & (YAxis >= 0 & YAxis
    <= 479);
36  //assign border = ((XAxis >=0 & XAxis <= 7) | (YAxis >= 0 &
    YAxis <= 7) | (XAxis >= 632 & XAxis <= 639) | (YAxis >= 472 &
    YAxis <= 479));
37  assign topborder = (YAxis >= 0 & YAxis <= 7);
38  assign bottomborder = (YAxis >= 472 & YAxis <= 479);
39  assign leftborder = (XAxis >=0 & XAxis <= 7);
40  assign rightborder = (XAxis >= 632 & XAxis <= 639);
41  assign border = ActR & (leftborder | rightborder | topborder |
```

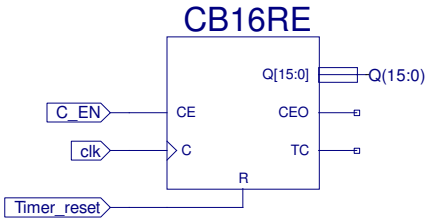
```
      bottomborder);// & ActR;
42
43
44    endmodule
45
```

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////
    /////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date:    22:40:08 05/23/2016
7   // Design Name:
8   // Module Name:    Slug
9   // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////
    /////////////////
21  module Slug(clock, XAxis, YAxis, move, level, direction, ActR,
    slug, toggle);
22
23     input clock;
24     input [15:0]XAxis;
25     input [15:0]YAxis;
26     input [15:0]move;
27     input [15:0]level;
28     input direction;
29     input ActR;
30
31     output slug;
32     output toggle;
33  //output slug_leftbound;
34  //output slug_rightbound;
35
36  //wire slug_leftbound;
37  //wire slug_rightbound;
38
39  //assign XAxis_MAX = 799;
40  //assign XAxis_MIN = 0;
41  //assign slug_LE = (XAxis >= (7 + move));
42  //assign slug_RE = (XAxis <= (71 + move));
43     assign slug = (ActR & (((XAxis >= (8 + move ) & XAxis <= (56
```

```
      + move + level) & (YAxis >= 404 & YAxis <= 420)))));
44    //assign slug = (slug_LE & slug_RE & (420 >= YAxis >= 404));
45    //assign travel = (slug > XAxis + 7) & (slug + 568 < XAxis);
46    //assign toggle = (slug == border);
47    //assign slug_leftbound = (move + 7);
48    //assign slug_rightbound = (move + 71);
49    //assign toggle = (slug_leftbound == 8 | slug_rightbound == 631);
50    //assign toggle = (slug == 7 | slug == 632);
51       assign toggle = ((56 + move + level >= 631) & direction) | (
      move<=0 & ~direction);
52    //assign toggle = (slug_LE == slug_leftbound) | (slug_RE ==
      slug_rightbound);
53
54
55    endmodule
56
```

Time Counter



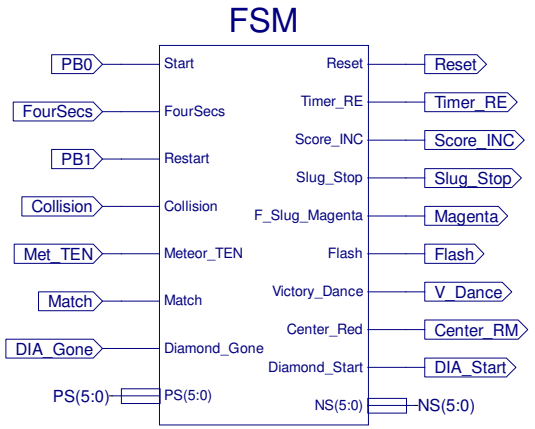


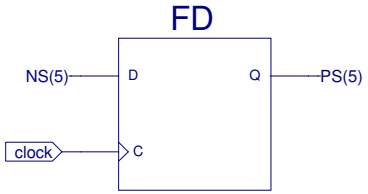
```
 1  `timescale 1ns / 1ps
 2  //////////////////////////////////////////////////////////////////////////////////
 3  // Company:
 4  // Engineer:
 5  //
 6  // Create Date:    20:04:35 05/28/2016
 7  // Design Name:
 8  // Module Name:    FSM
 9  // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21  module FSM(Start, FourSecs, Restart, Collision, Meteor_TEN,
    Match, Diamond_Gone, PS, NS, Reset, Timer_RE, Score_INC,
    Slug_Stop, F_Slug_Magenta, Flash, Victory_Dance, Center_Red,
    Diamond_Start);
22
23      input Start;
24      input FourSecs;
25      input Restart;
26      input Collision;
27      input Meteor_TEN;
28      input Match;
29      input Diamond_Gone;
30      input [5:0] PS;
31
32      output [5:0] NS;
33      output Reset;
34      output Timer_RE;
35      output Score_INC;
36      output Slug_Stop;
37      output F_Slug_Magenta;
38      output Flash;
39      output Victory_Dance;
40      output Center_Red;
41      output Diamond_Start;
```

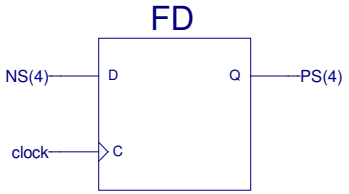
```
42
43
44    wire Idle, Start_Game, Lose, Level_Up, Grow, Win;
45    wire Next_Idle, Next_Start_Game, Next_Lose, Next_Level_Up,
      Next_Grow, Next_Win;
46
47    assign Idle = PS[0];
48    assign Start_Game = PS[1];
49    assign Lose = PS[2];
50    assign Level_Up = PS[3];
51    assign Grow = PS[4];
52    assign Win = PS[5];
53
54    assign NS[0] = Next_Idle;
55    assign NS[1] = Next_Start_Game;
56    assign NS[2] = Next_Lose;
57    assign NS[3] = Next_Level_Up;
58    assign NS[4] = Next_Grow;
59    assign NS[5] = Next_Win;
60
61
62       assign Next_Idle = (Idle & ~Start) | (Win & Restart) | (Lose
      & Restart);
63       assign Next_Start_Game = (Start_Game & ~Collision & ~
      Meteor_TEN) | (Idle & Start) | (Grow & FourSecs);
64       assign Next_Lose = (Lose & ~Restart) | (Start_Game &
      Collision);
65       assign Next_Level_Up = (Level_Up & ~FourSecs) | (Start_Game &
       Meteor_TEN & ~Match);
66       assign Next_Grow = (Grow & ~FourSecs) | (Level_Up & FourSecs);
67       assign Next_Win = (Win & ~Restart) | (Start_Game & Match &
      Meteor_TEN);
68
69    // assign Meteor_TEN = (MET_reset + MET_reset + MET_reset +
      MET_reset + MET_reset + MET_reset + MET_reset + MET_reset +
      MET_reset + MET_reset);
70
71    // assign Load = (Idle & Start) | (Grow & FourSecs);
72       assign Reset = Idle;
73       assign Timer_RE = ((Grow & FourSecs) | (Level_Up & FourSecs)
      | (Start_Game & Meteor_TEN));
74       assign Score_INC = (Start_Game & Diamond_Gone);
75       assign Slug_Stop = (Lose | Level_Up | Grow | Win);
76    // assign Level_Inc = Meteor_TEN;
77       assign F_Slug_Magenta = Level_Up;
78       assign Flash = (Lose | Level_Up | Grow);
```

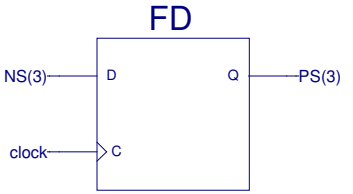
```
79      assign Victory_Dance = Win;
80      assign Center_Red = Idle;
81      assign Diamond_Start = Start_Game;
82
83
84
85  endmodule
86
```

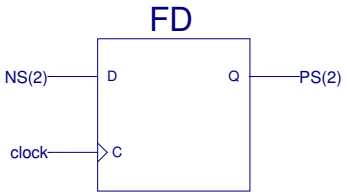
FSM Flip Flops

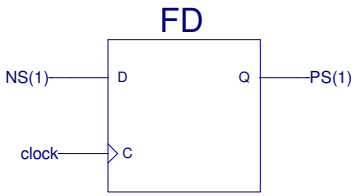


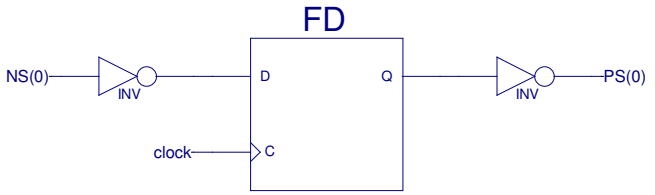




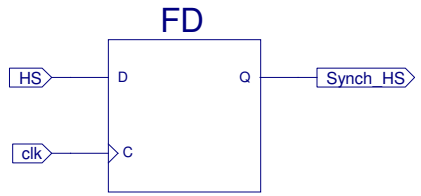


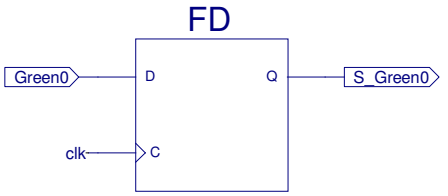


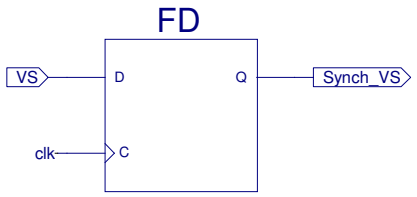


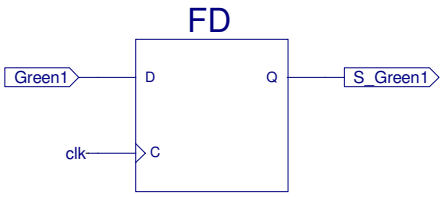


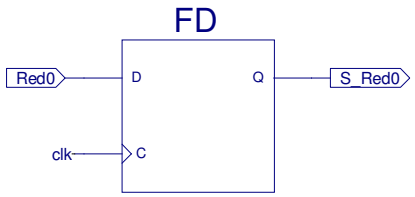
Synchronizer

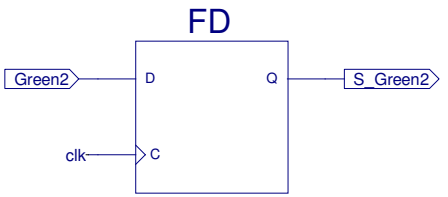


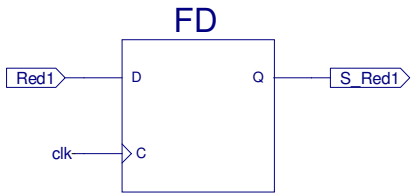


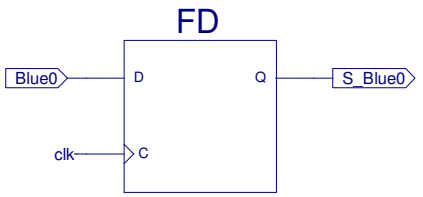


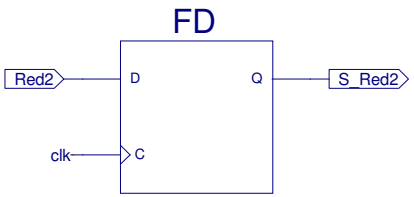


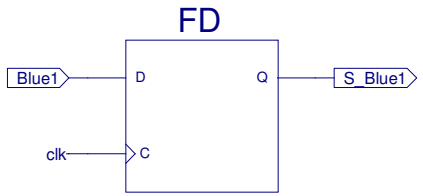




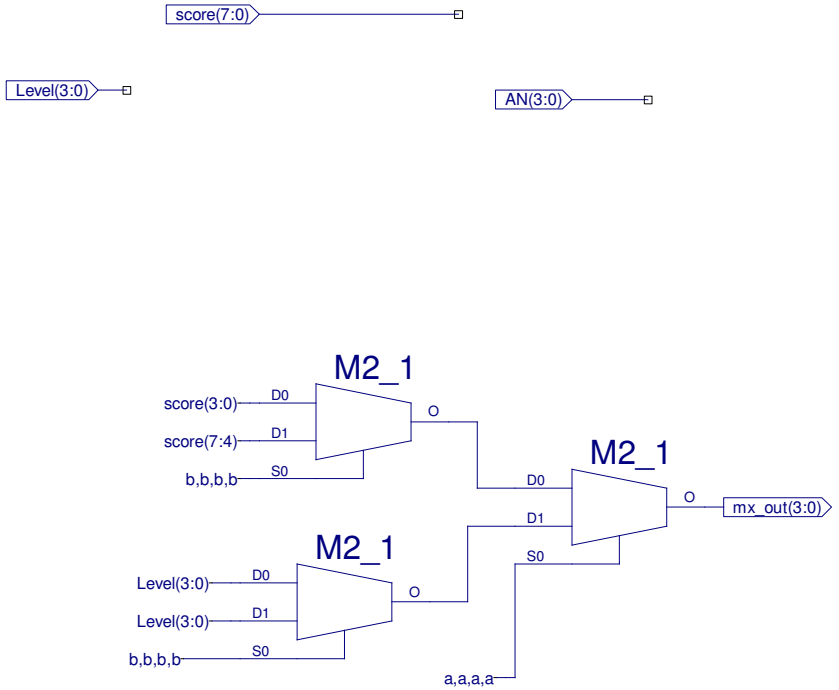


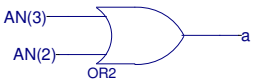


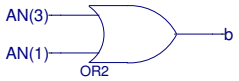


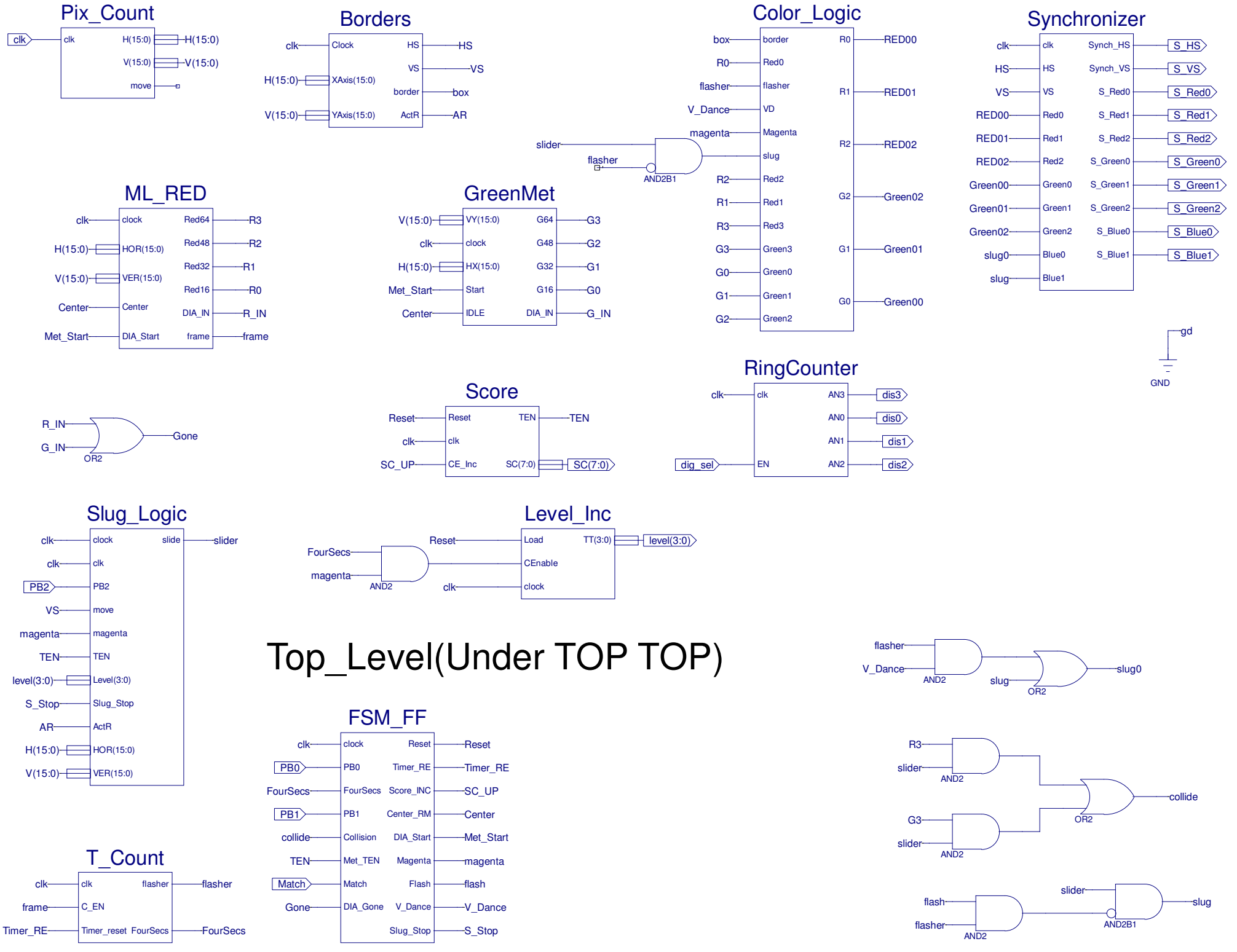


Selector

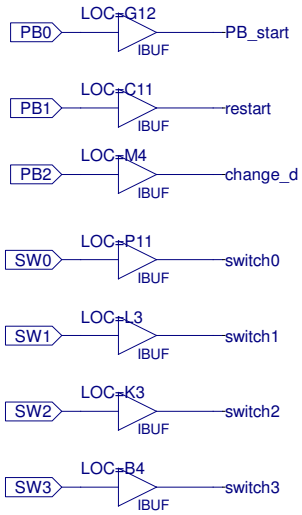

score(7:0)
Level(3:0)
AN(3:0)
M2_1
score(3:0)
D0
score(7:4)
D1
b,b,b,b
S0
O
M2_1
D0
D1
S0
O
mx_out(3:0)
M2_1
Level(3:0)
D0
Level(3:0)
D1
b,b,b,b
S0
O
a,a,a,a


AN(3)
AN(2)
OR2
a

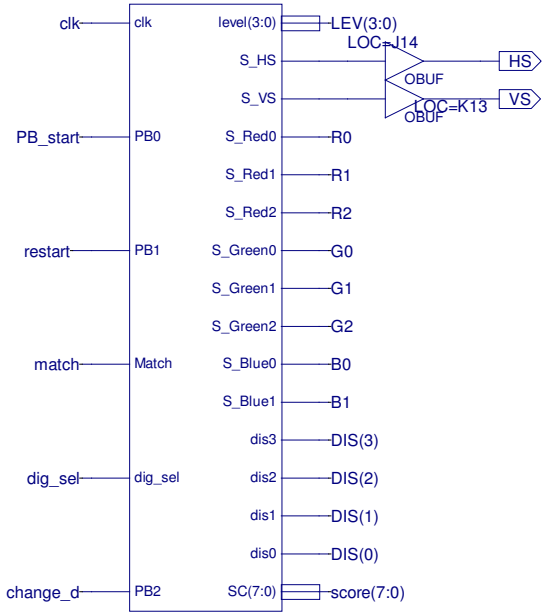

AN(3)
AN(1)
OR2
b


Pix_Count
clk
clk
H(15:0)
V(15:0)
move
H(15:0)
V(15:0)
Borders
clk
H(15:0)
V(15:0)
Clock
XAxis(15:0)
YAxis(15:0)
HS
VS
border
ActR
HS
VS
box
AR
Color_Logic
box
R0
flasher
V_Dance
magenta
slider
flasher
AND2B1
border
Red0
flasher
VD
Magenta
slug
Red2
Red1
Red3
Green3
Green0
Green1
Green2
R2
R1
R3
G3
G0
G1
G2
R0
R1
R2
G2
G1
G0
RED00
RED01
RED02
Green02
Green01
Green00
Synchronizer
clk
HS
VS
RED00
RED01
RED02
Green00
Green01
Green02
slug0
slug
clk
HS
VS
Red0
Red1
Red2
Green0
Green1
Green2
Blue0
Blue1
Synch_HS
Synch_VS
S_Red0
S_Red1
S_Red2
S_Green0
S_Green1
S_Green2
S_Blue0
S_Blue1
S_HS
S_VS
S_Red0
S_Red1
S_Red2
S_Green0
S_Green1
S_Green2
S_Blue0
S_Blue1
gd
GND
ML_RED
clk
H(15:0)
V(15:0)
Center
Met_Start
clock
HOR(15:0)
VER(15:0)
Center
DIA_Start
Red64
Red48
Red32
Red16
DIA_IN
frame
R3
R2
R1
R0
R_IN
frame
GreenMet
V(15:0)
clk
H(15:0)
Met_Start
Center
VY(15:0)
clock
HX(15:0)
Start
IDLE
G64
G48
G32
G16
DIA_IN
G3
G2
G1
G0
G_IN
R_IN
G_IN
OR2
Gone
Score
Reset
clk
SC_UP
Reset
clk
CE_Inc
TEN
SC(7:0)
TEN
SC(7:0)
RingCounter
clk
dig_sel
clk
EN
AN3
AN0
AN1
AN2
dis3
dis0
dis1
dis2
Slug_Logic
clk
clk
PB2
VS
magenta
TEN
level(3:0)
S_Stop
AR
H(15:0)
V(15:0)
clock
clk
PB2
move
magenta
TEN
Level(3:0)
Slug_Stop
ActR
HOR(15:0)
VER(15:0)
slide
slider
Level_Inc
FourSecs
magenta
AND2
Reset
clk
Load
CEnable
clock
TT(3:0)
level(3:0)
Top_Level(Under TOP TOP)
FSM_FF
clk
PB0
FourSecs
PB1
collide
TEN
Match
Gone
clock
PB0
FourSecs
PB1
Collision
Met_TEN
Match
DIA_Gone
Reset
Timer_RE
Score_INC
Center_RM
DIA_Start
Magenta
Flash
V_Dance
Slug_Stop
Reset
Timer_RE
SC_UP
Center
Met_Start
magenta
flash
V_Dance
S_Stop
T_Count
clk
frame
Timer_RE
clk
C_EN
Timer_reset
flasher
FourSecs
flasher
FourSecs
flasher
V_Dance
AND2
slug
OR2
slug0
R3
slider
AND2
G3
slider
AND2
OR2
collide
flash
flasher
AND2
slider
AND2B1
slug

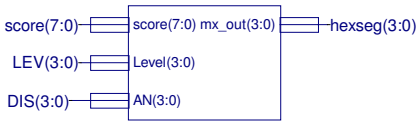
TOP_TOP

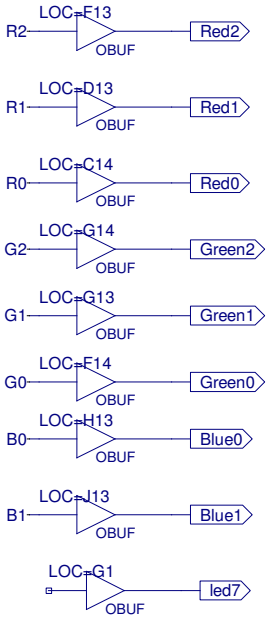


Top_Level

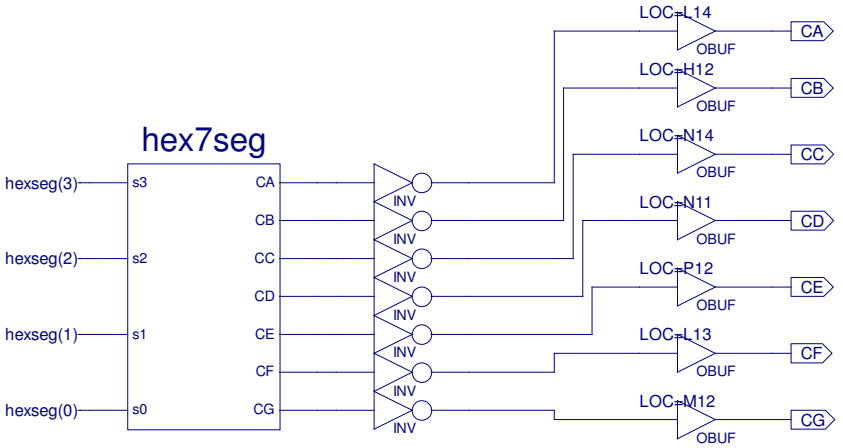


NEW_SELECT4

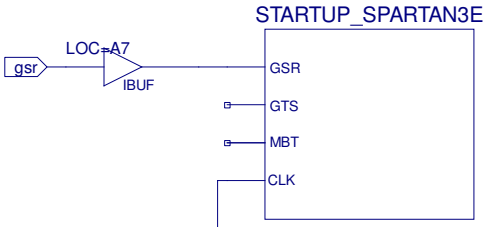




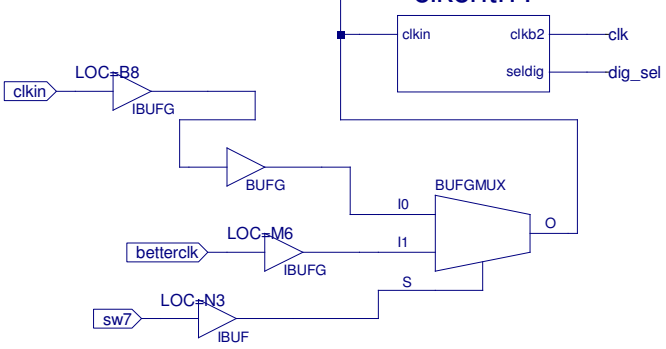
hex7seg



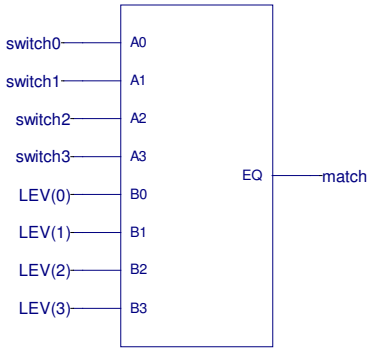
STARTUP_SPARTAN3E

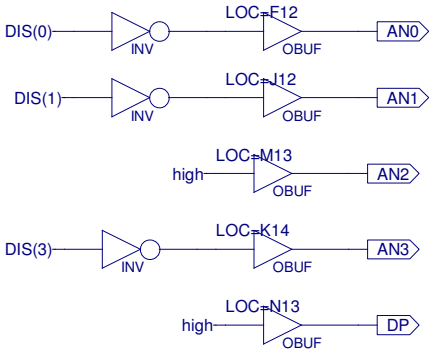


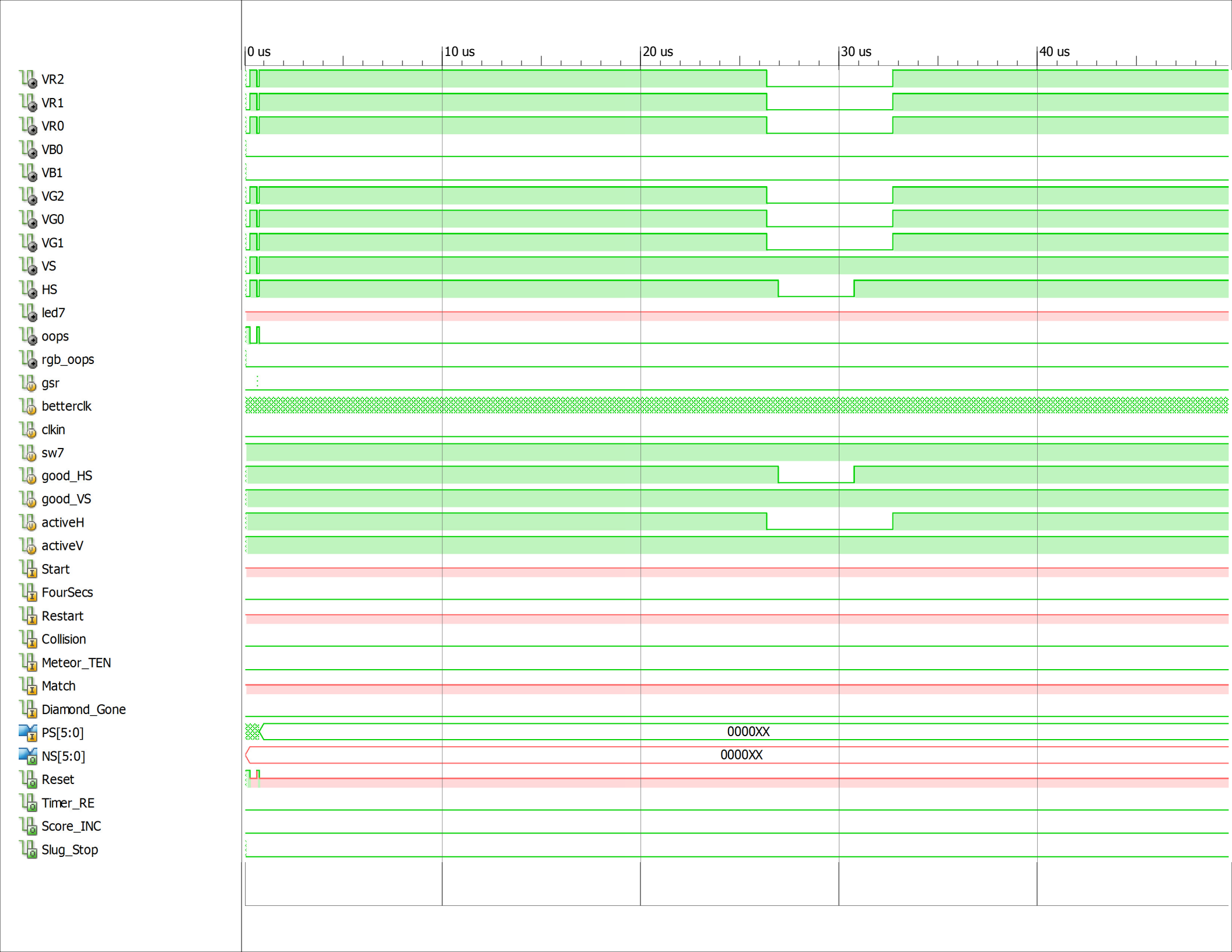
clkcntrl4

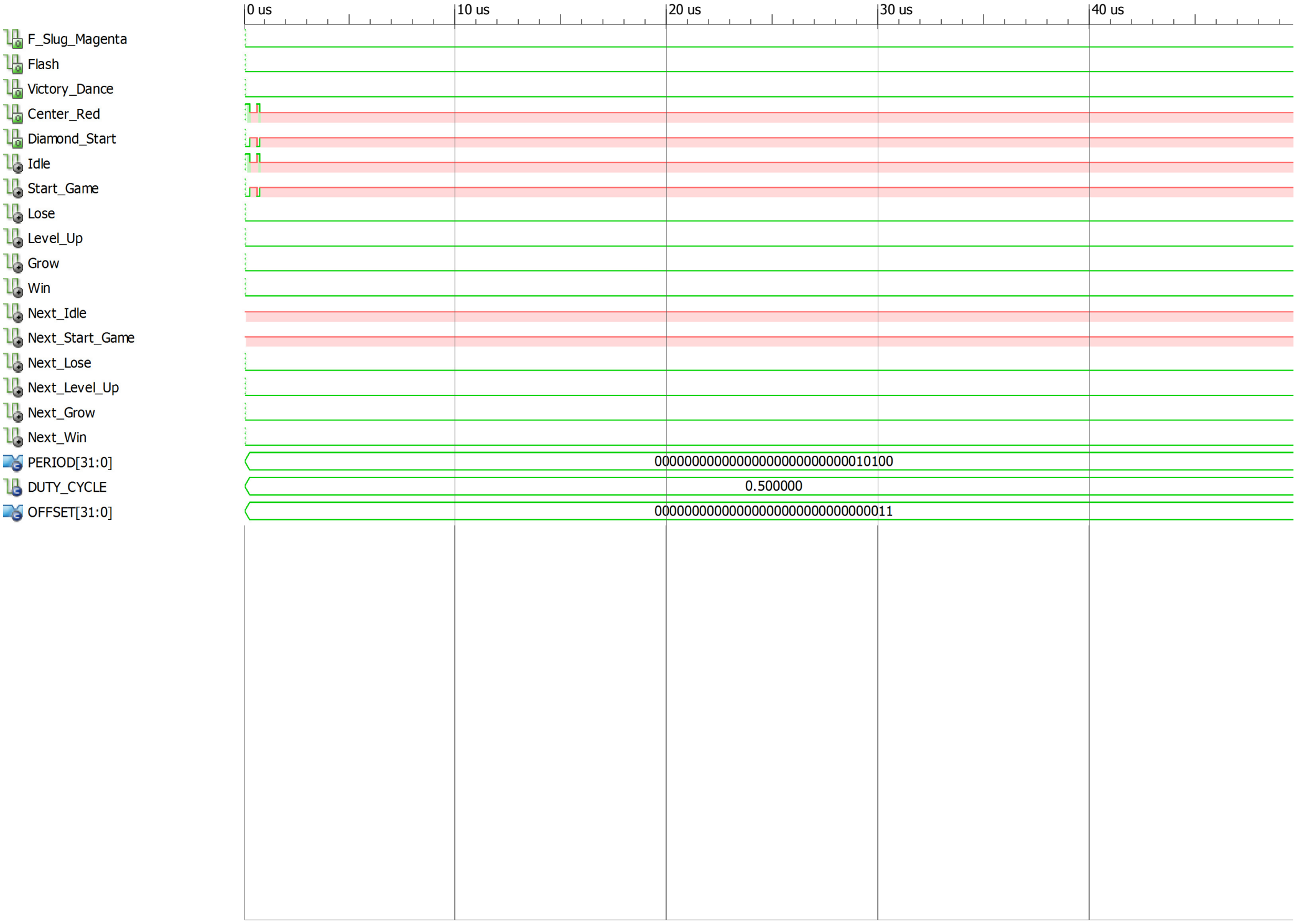


COMP4






0 us
10 us
20 us
30 us
40 us
VR2
VR1
VR0
VB0
VB1
VG2
VG0
VG1
VS
HS
led7
oops
rgb_oops
gsr
betterclk
clkin
sw7
good_HS
good_VS
activeH
activeV
Start
FourSecs
Restart
Collision
Meteor_TEN
Match
Diamond_Gone
PS[5:0]
NS[5:0]
Reset
Timer_RE
Score_INC
Slug_Stop
0000XX
0000XX


0 us
10 us
20 us
30 us
40 us
F_Slug_Magenta
Flash
Victory_Dance
Center_Red
Diamond_Start
Idle
Start_Game
Lose
Level_Up
Grow
Win
Next_Idle
Next_Start_Game
Next_Lose
Next_Level_Up
Next_Grow
Next_Win
PERIOD[31:0]
00000000000000000000000000010100
DUTY_CYCLE
0.500000
OFFSET[31:0]
00000000000000000000000000000011

```
--------------------------------------------------------------------------------
Release 14.7 Trace  (nt64)
Copyright (c) 1995-2013 Xilinx, Inc.  All rights reserved.

C:\Xilinx\14.7\ISE_DS\ISE\bin\nt64\unwrapped\trce.exe -filter
C:/Users/hqayum/Desktop/Lab7_DONE!!!/Lab7_CE100/iseconfig/filter.filter
-intstyle ise -v 3 -s 5 -n 3 -fastpaths -xml TOP_TOP.twx TOP_TOP.ncd -o
TOP_TOP.twr TOP_TOP.pcf

Design file:              TOP_TOP.ncd
Physical constraint file: TOP_TOP.pcf
Device,package,speed:     xc3s100e,cp132,-5 (PRODUCTION 1.27 2013-10-13)
Report level:             verbose report

Environment Variable      Effect
--------------------      ------
NONE                      No environment variables were set
--------------------------------------------------------------------------------

INFO:Timing:2698 - No timing constraints found, doing default enumeration.

INFO:Timing:3412 - To improve timing, see the Timing Closure User Guide (UG612).
INFO:Timing:2752 - To get complete path coverage, use the unconstrained paths
   option. All paths that are not constrained will be reported in the
   unconstrained paths section(s) of the report.
INFO:Timing:3339 - The clock-to-out numbers in this timing report are based on
   a 50 Ohm transmission line loading model.  For the details of this model,
   and for more information on accounting for different loading conditions,
   please see the device datasheet.
INFO:Timing:3390 - This architecture does not support a default System Jitter
   value, please add SYSTEM_JITTER constraint to the UCF to modify the Clock
   Uncertainty calculation.
INFO:Timing:3389 - This architecture does not support 'Discrete Jitter' and
   'Phase Error' calculations, these terms will be zero in the Clock
   Uncertainty calculation.  Please make appropriate modification to
   SYSTEM_JITTER to account for the unsupported Discrete Jitter and Phase
   Error.



Data Sheet report:
-----------------
All values displayed in nanoseconds (ns)

Setup/Hold to clock betterclk
------------+------------+------------+------------------+--------+
            |Max Setup to|Max Hold to |                  | Clock  |
Source      | clk (edge) | clk (edge) |Internal Clock(s) | Phase  |
------------+------------+------------+------------------+--------+
PB0         |    2.277(R)|   -0.156(R)|clk               |   0.000|
PB1         |    1.629(R)|    0.566(R)|clk               |   0.000|
PB2         |   -0.062(R)|    1.145(R)|clk               |   0.000|
SW0         |    4.154(R)|   -1.382(R)|clk               |   0.000|
SW1         |    5.149(R)|   -2.178(R)|clk               |   0.000|
SW2         |    4.643(R)|   -1.773(R)|clk               |   0.000|
SW3         |    3.344(R)|   -0.734(R)|clk               |   0.000|
sw7         |    1.234(R)|   -0.132(R)|XLXN_57           |   0.000|
------------+------------+------------+------------------+--------+
```

Setup/Hold to clock clkin

Source	Max Setup to clk (edge)	Max Hold to clk (edge)	Internal Clock(s)	Clock Phase
PB0	0.094(R)	2.497(R)	clk	0.000
PB1	-0.554(R)	3.219(R)	clk	0.000
PB2	-2.245(R)	3.798(R)	clk	0.000
SW0	1.971(R)	1.271(R)	clk	0.000
SW1	2.966(R)	0.475(R)	clk	0.000
SW2	2.460(R)	0.880(R)	clk	0.000
SW3	1.161(R)	1.919(R)	clk	0.000
sw7	-0.949(R)	2.521(R)	XLXN_60	0.000

Clock betterclk to Pad

Destination	clk (edge) to PAD	Internal Clock(s)	Clock Phase
AN0	7.419(R)	clk	0.000
AN1	7.012(R)	clk	0.000
AN3	6.934(R)	clk	0.000
Blue0	7.445(R)	clk	0.000
Blue1	7.758(R)	clk	0.000
CA	12.011(R)	clk	0.000
CB	12.219(R)	clk	0.000
CC	11.800(R)	clk	0.000
CD	12.246(R)	clk	0.000
CE	11.728(R)	clk	0.000
CF	11.902(R)	clk	0.000
CG	12.132(R)	clk	0.000
Green0	7.217(R)	clk	0.000
Green1	7.099(R)	clk	0.000
Green2	7.306(R)	clk	0.000
HS	7.638(R)	clk	0.000
Red0	7.490(R)	clk	0.000
Red1	7.662(R)	clk	0.000
Red2	7.201(R)	clk	0.000
VS	7.124(R)	clk	0.000

Clock clkin to Pad

Destination	clk (edge) to PAD	Internal Clock(s)	Clock Phase
AN0	10.072(R)	clk	0.000
AN1	9.665(R)	clk	0.000
AN3	9.587(R)	clk	0.000
Blue0	10.098(R)	clk	0.000
Blue1	10.411(R)	clk	0.000
CA	14.664(R)	clk	0.000
CB	14.872(R)	clk	0.000
CC	14.453(R)	clk	0.000
CD	14.899(R)	clk	0.000
CE	14.381(R)	clk	0.000

CF	14.555(R)	clk	0.000
CG	14.785(R)	clk	0.000
Green0	9.870(R)	clk	0.000
Green1	9.752(R)	clk	0.000
Green2	9.959(R)	clk	0.000
HS	10.291(R)	clk	0.000
Red0	10.143(R)	clk	0.000
Red1	10.315(R)	clk	0.000
Red2	9.854(R)	clk	0.000
VS	9.777(R)	clk	0.000

Clock to Setup on destination clock betterclk

	Src:Rise	Src:Fall	Src:Rise	Src:Fall
Source Clock	Dest:Rise	Dest:Rise	Dest:Fall	Dest:Fall
betterclk	20.668			
clkin	20.668			

Clock to Setup on destination clock clkin

	Src:Rise	Src:Fall	Src:Rise	Src:Fall
Source Clock	Dest:Rise	Dest:Rise	Dest:Fall	Dest:Fall
betterclk	20.668			
clkin	20.668			

Analysis completed Fri Jun 03 17:50:35 2016

Trace Settings:

Trace Settings

Peak Memory Usage: 160 MB