Rolling In The Deep

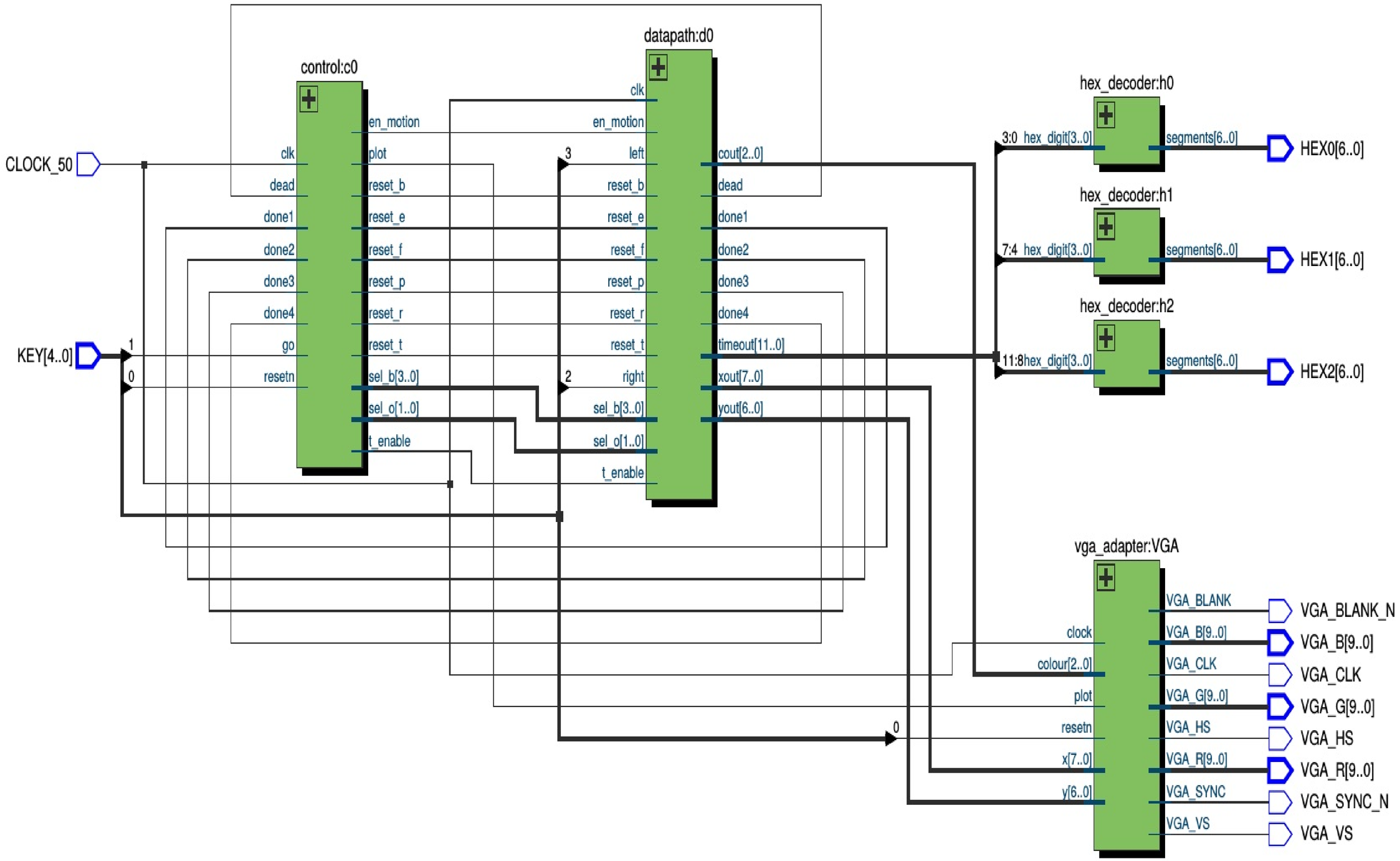
By Bin Liu and Guanxiong Liu

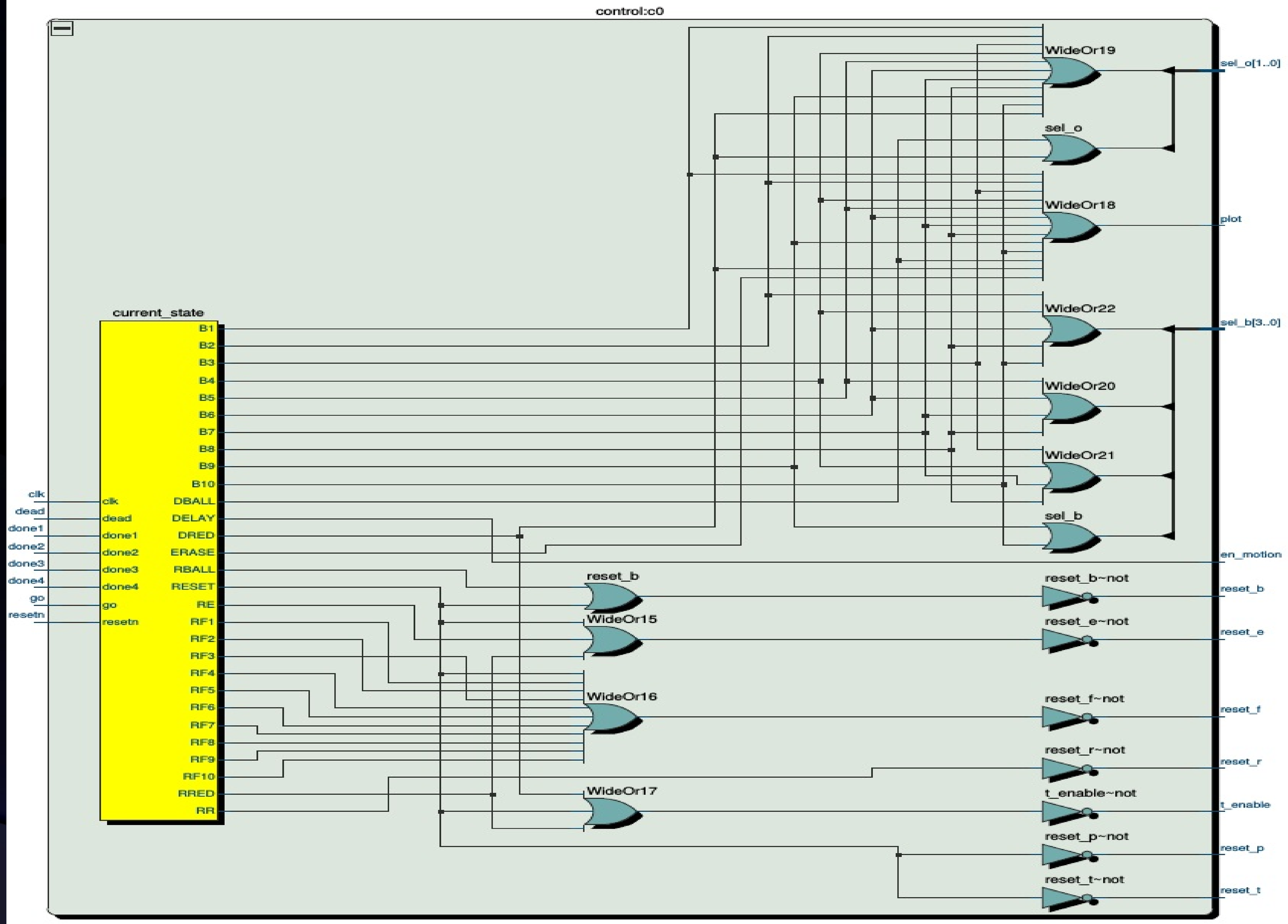
Game Description

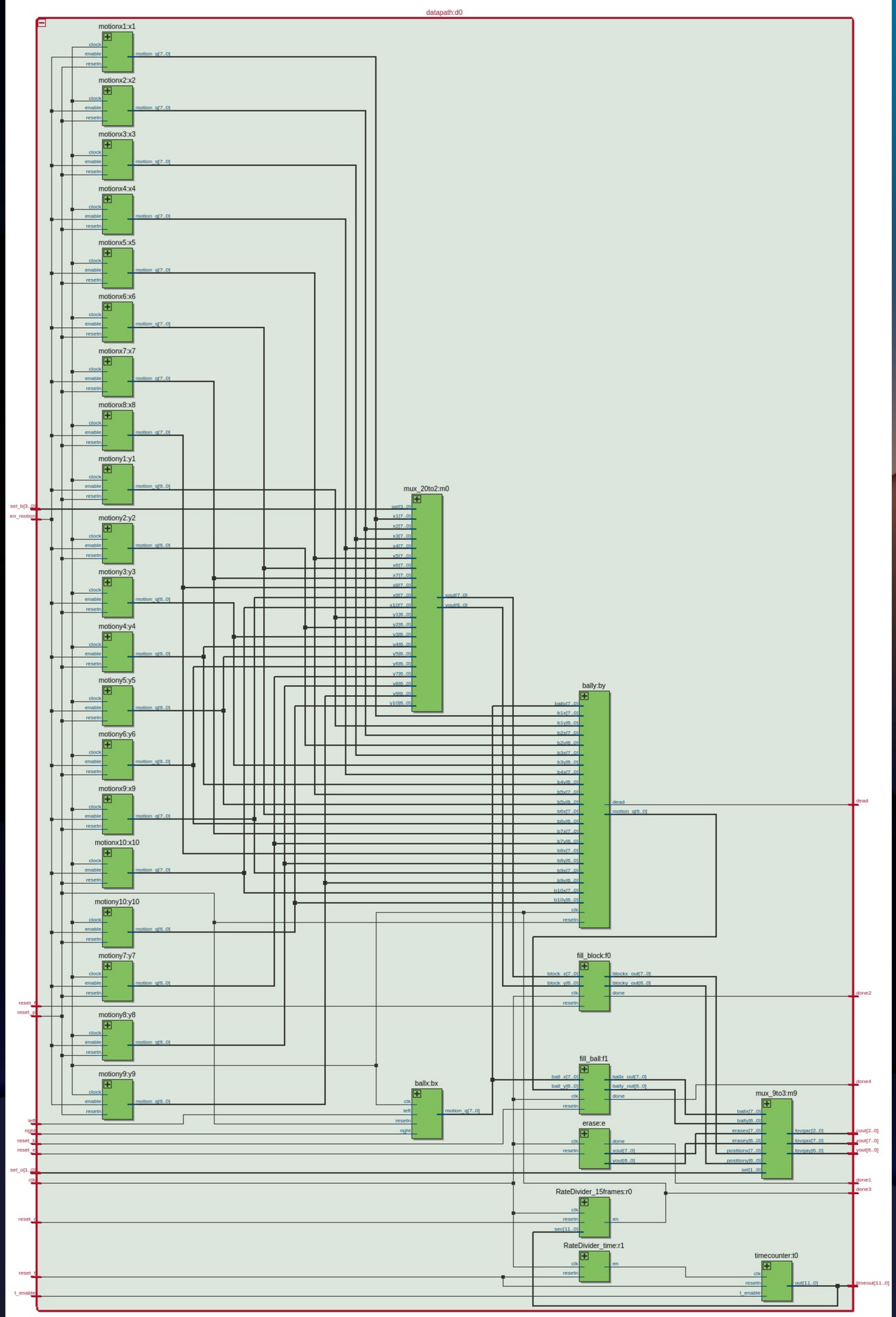
Our game is called Rolling In The Deep, which is a one player game where player controls a ball that move left or right.
There are blocks that randomly appeared start from bottom of the screen that move constantly up. If the ball falls on a block, it would move up along with the block it fell on. However, whenever blocks are generated there are always at least one gap that allow player to escape from moving out of screen. Whenever the ball falls on a hole, it will drop from the initial level until a block appeared below.

The goal of player is to survived as long as he could. The game will keep track of the time has passed as the score of that player. The player controls the ball through keys that provided from the De1-Soc board.

Block diagram




control:c0
current_state
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
clk
dead
done1
done2
done3
done4
go
resetn
DBALL
DELAY
DRED
ERASE
RBALL
RESET
RE
RF1
RF2
RF3
RF4
RF5
RF6
RF7
RF8
RF9
RF10
RRED
RR
WideOr19
sel_o[1..0]
sel_o
WideOr18
plot
WideOr22
sel_b[3..0]
WideOr20
WideOr21
sel_b
en_motion
reset_b
WideOr15
reset_b~not
reset_b
reset_e~not
reset_e
WideOr16
reset_f~not
reset_f
reset_r~not
reset_r
WideOr17
t_enable~not
t_enable
reset_p~not
reset_p
reset_t~not
reset_t


datapath:d0
motionx1:x1
motionx2:x2
motionx3:x3
motionx4:x4
motionx5:x5
motionx6:x6
motionx7:x7
motionx8:x8
motiony1:y1
motiony2:y2
motiony3:y3
motiony4:y4
motiony5:y5
motiony6:y6
motionx9:x9
motionx10:x10
motiony10:y10
motiony7:y7
motiony8:y8
motiony9:y9
mux_20to2:m0
bally:by
fill_block:f0
fill_ball:f1
ballx:bx
erase:e
mux_9to3:m9
RateDivider_15frames:r0
RateDivider_time:r1
timecounter:t0
clock
enable
resetn
motion_q[7..0]
motion_q[6..0]
clk
left
right
en
dead
done
done1
done2
done3
done4
t_enable

Msgs	
/erase/clk	St0
/erase/resetn	St1
/erase/xout	01001111
/erase/yout	1000100
/erase/done	0
/erase/q1	01001111
/erase/q2	01000100

Msgs	
/fill_block/clk	St1
/fill_block/resetn	St1
/fill_block/block_x	00001010
/fill_block/block_y	0010100
/fill_block/block...	00011000
/fill_block/block...	0010101
/fill_block/done	0
/fill_block/q1	001110
/fill_block/q2	000001

Msgs	
/motionx1/resetn	St0
/motionx1/clock	St0
/motionx1/enable	St0
/motionx1/moti...	00000000

Msgs	
/mux_20to2/x1	00000001
/mux_20to2/x2	00000011
/mux_20to2/x3	00000101
/mux_20to2/x4	00000111
/mux_20to2/x5	00001001
/mux_20to2/x6	00001011
/mux_20to2/x7	00001101
/mux_20to2/x8	00001111
/mux_20to2/x9	00010001
/mux_20to2/x10	00010011
/mux_20to2/y1	0000010
/mux_20to2/y2	0000100
/mux_20to2/y3	0000110
/mux_20to2/y4	0001000
/mux_20to2/y5	0001010
/mux_20to2/y6	0001100
/mux_20to2/y7	0001110
/mux_20to2/y8	0010000
/mux_20to2/y9	0010010
/mux_20to2/y10	0010100
/mux_20to2/sel	1001
/mux_20to2/xout	00010011
/mux_20to2/yout	0010100

Msgs	
/motiony1/resetn	St1
/motiony1/clock	St0
/motiony1/enable	St1
/motiony1/moti...	0000000

Msgs	
/RateDivider_15...	St1
/RateDivider_15...	St1
/RateDivider_15...	1
/RateDivider_15...	0000000000...

Now 10010 ns

Cursor 1 4059 ns

4056 ns 4058 ns 4059 ns 4060 ns 4062 ns

Difficulties

The most difficulties we have encountered are data flows and controlling of a large number of devices in data path.

At first, we have no idea how to erase the screen completely. After we had conversation with Professor Mytro, we know we have to reset the filling counter and go to the next state to have the counter to output every (x,y) coordinates on the screen. After many times farilure, we finally got the correct animation.

In addition,we also had a hard time dealing with collision detection. At the beginning, we thought the detection process should be in FSM. We spent more than 3 hours to realize that we should implement it in the position counter instead. Since position counter is the only device that record the position of objects. Also, when we did the simulation on De1-Soc board, we found that the block did not prevent the ball from dropping down. We then spent another 2 hours to figured out that it is caused by the sequence of if and else if in ball module.If we put one condition before another, the program will never go to the another condition since when the first condition is true,the program won't check the next condition.After we corrected the order, collision detection works as we expected.

Difficulties

Job Done by

Guanxiong Liu:

Ten pairs X and y position counters,Counter for delay the animation (frame counter), mux 20 to 2 for choosing which block to draw,FSM for showing ten blocks on the screen, FSM for controlling the motion of the blocks, counter for recording how much time it elapsed. Ball’s collision detection, accelerating game speed as time progresses, debugging and stimulation.

Bin Liu:

Counter for drawing block, counter for erasing the whole screen, mux 9 to 3 for choosing the object to draw(erase, block or ball), counter for drawingball, FSM for erasing the whole screen and make two block moving in different direction, position counters for ball’s coordinates, Ball’s collision detection, debugging and stimulation.

What we learned

We are now have more experience on how to implement a circuit that we expect. The very first thing is to draw the high level schematic. This will make our mind much clear when do Verilog coding as well as debugging.

Be careful when choosing name for wire and device. Otherwise, the time spend on debugging will more than the time spend on implementing it if we connect the wire in the wrong place. This project improves our carefulness when dealing with large number of devices, also enhance our experience with Verilog as well as modelsim.

Other than that, we kind learned how game actually draw pixel on the screen and interact with users. It is very exciting to play a game that wrote by ourself.