**P**

**Comparator**

Comparing if dice value is seven

**DICE ROLL**

By using three D Flip-flop we make upward counter by attaching a clock

**Comparator**

Comparing if dice value is zero

MUX SYSTEM

A system of three MUXes returning same value from dice if it is not 0 or 7.

If so giving 1 in case of zero and 6 in case of 7

Y

-> Two registers with a clock system are attached to store player's position on board

-> Since we know register stores value only on positive tig so a clock system,

switch and two registers are used to tackle this problem.

->when switch is one means blue player’s turn so both of these registers will not

operate

-> But when switch is zero means red player’s turn one of these two registers will operate

-> If clock tig is negative with switch zero register 1 will store value otherwise if clock tig is positive with switch zero then register two will operate.

SYSTEM OF REGISTERS WITH CLOCK SETUP

PLAYER’S NEW POSITION GOING BACK TO SAVE IN REGISTERS

Player’s position

**Comparator**

Five bit comparator to check if current position of played is equal to first snake mouth or not

**Position of first snakes mouth = 15**

**WORKING FOR PLAYER1**

**Adder**

Adder to add initial position of player 1\and dice value

and dice value

Giving players position to subtractor

Output from above comparator in form of zero or 1

**MUX**

>here are five multiplexers. Each multiplexer gives one bit of player one's new position

>All multiplexers have same selection lines that are outputs of fours comparators (means what S0 of first Mux is same for other four mux)

**>Working**

->So from four comparators only one will give one at a time if coming positions matches its conditions while other comparators will give 0.

->Since all Mux have same selection inputs so all will give output coming from same mux slots (means if four selection bits for a mux

are 0001 then they will same for all other mux and ultimately all will give Output from I1 input slot.

-> now for above example if 0001 selection inputs represents 1st comparator so player position according to its subtractor should be consider.

So to tackle this since all Muxes will give I1 output in this case so so each mux contain a bit of that position at I1 position

**Subtractor**

>A switch controlled subtractor if turned on by above comparator, subtracts and

Brings player back to first snake's tail

> Else give same player position

**Difference between second snake's mouth and tail = 10 (19)**

**Comparator**

Five bit comparator to check if current position of played is equal to second snake mouth or not

**Position of second snake's mouth = 29**

**Subtractor**

>A switch controlled subtractor if turned on by above comparator, subtracts and

Brings player back to first snake's tail

> Else give same player position

**Difference between first snake mouth and tail = 8(7)**

Player’s position

Giving players position to subtractor

Output from above comparator in form of zero or 1

Common selection lines of each of five MUX from four comparitors

Player’s position

**Comparator**

Five bit comparator to check if current position of player is equal to first lader's foot, if same gives zero else one

Position of first ladder's bottom = 12

Giving players position to adder

Output from above comparator in form of zero or 1

**Adder**

>A switch controlled adder if turned on by above comparator, adds and

Brings player to top of first ladder

> Else give same player position

Difference between first ladder's top and bottom = 9 (21)

**Comparator**

Five bit comparator to check if current position of player is equal to second lader's foot, if same gives zero else one

Position of second ladder’s bottom = 18

Player’s position

Giving players position to adder

Output from above comparator in form of zero or 1

**Adder**

>A switch controlled adder if turned on by above comparator, adds and

Brings player to top of second ladder

> Else give same player position

Difference between second ladder top and bottom = 12(30)

**GAME BOARD**

**BOARD**

**WORKING FOR PLAYER 2**

>31 cells for each player and

A clock or a dice that give the

Players position

PLAYER’S NEW POSITION GOING BACK TO SAVE IN REGISTERS

**Subtractor**

>A switch controlled subtractor if turned on by above comparator, subtracts and

Brings player back to first snake's tail

> Else give same player position

**Difference between first snake mouth and tail = 8(7)**

Output from above comparator in form of zero or 1

Player’s position

**Comparator**

Five bit comparator to check if current position of played is equal to first snake mouth or not

**Position of first snakes mouth = 15**

**Adder**

Adder to add initial position of player 2 and dice value

Player’s position

**Comparator**

Five bit comparator to check if current position of played is equal to second snake mouth or not

**Position of second snake's mouth = 29**

**MUX**

>here are five multiplexers. Each multiplexer gives one bit of player one's new position

>All multiplexers have same selection lines that are outputs of fours comparators (means what S0 of first Mux is same for other four mux)

**>Working**

->So from four comparators only one will give one at a time if coming positions matches its conditions while other comparators will give 0.

->Since all Mux have same selection inputs so all will give output coming from same mux slots (means if four selection bits for a mux

are 0001 then they will same for all other mux and ultimately all will give Output from I1 input slot.

-> now for above example if 0001 selection inputs represents 1st comparator so player position according to its subtractor should be consider.

So to tackle this since all Muxes will give I1 output in this case so so each mux contain a bit of that position at I1 position

Giving players position to subtractor

Output from above comparator in form of zero or 1

**Subtractor**

>A switch controlled subtractor if turned on by above comparator, subtracts and

Brings player back to first snake's tail

> Else give same player position

**Difference between second snake's mouth and tail = 10 (19)**

Player’s position

**Comprator**

Five bit comparitor to check if current position of player is equal to first lader's foot, if same gives zero else one

Position of first ladder's bottom = 12

Giving players position to adder

Output from above comparator in form of zero or 1

**Adder**

>A switch controlled adder if turned on by above comparator, adds and

Brings player to top of first ladder

> Else give same player position

Difference between first ladder's top and bottom = 9 (21)

Common selection lines of each of five MUX from four comparitors

Player’s position

**Adder**

>A switch controlled adder if turned on by above comparator, adds and

Brings player to top of second ladder

> Else give same player position

Difference between second ladder top and bottom = 12(30)

**Comprator**

Five bit comparitor to check if current position of player is equal to second lader's foot, if same gives zero else one

Position of second ladder’s bottom = 18

Output from above comparator in form of zero or 1

Giving players position to adder