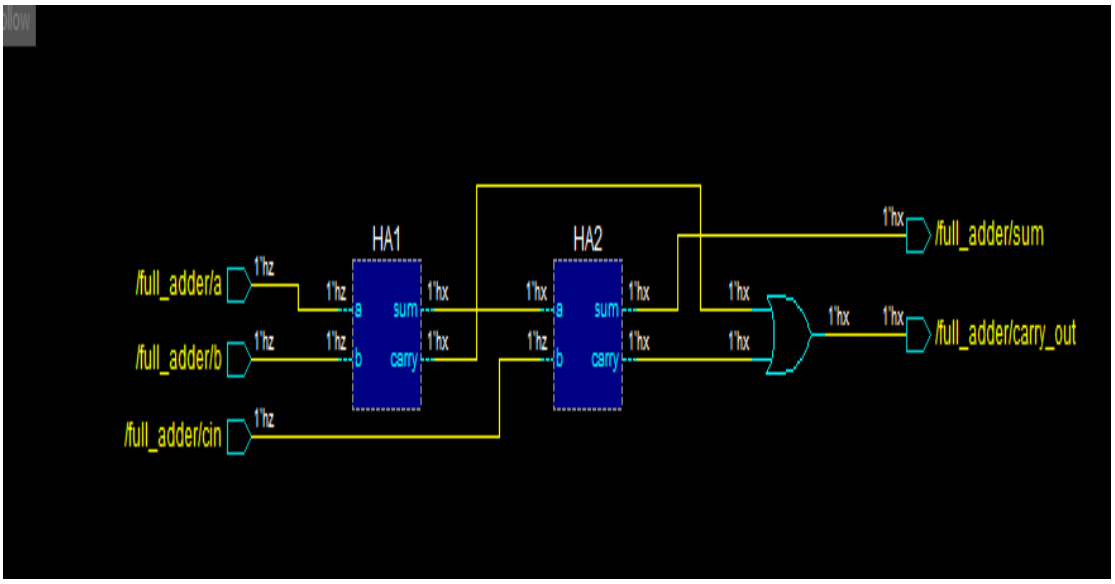
report

half adder and fall adder

```
1   module half_adder(
2   input a,b,
3   output sum,carry
4   );
5   assign sum=a^b;
6   assign carry=a&b;
7
8   endmodule
9
```

C:/Users/marrw/Desktop/adder/full_adder.v - Default *

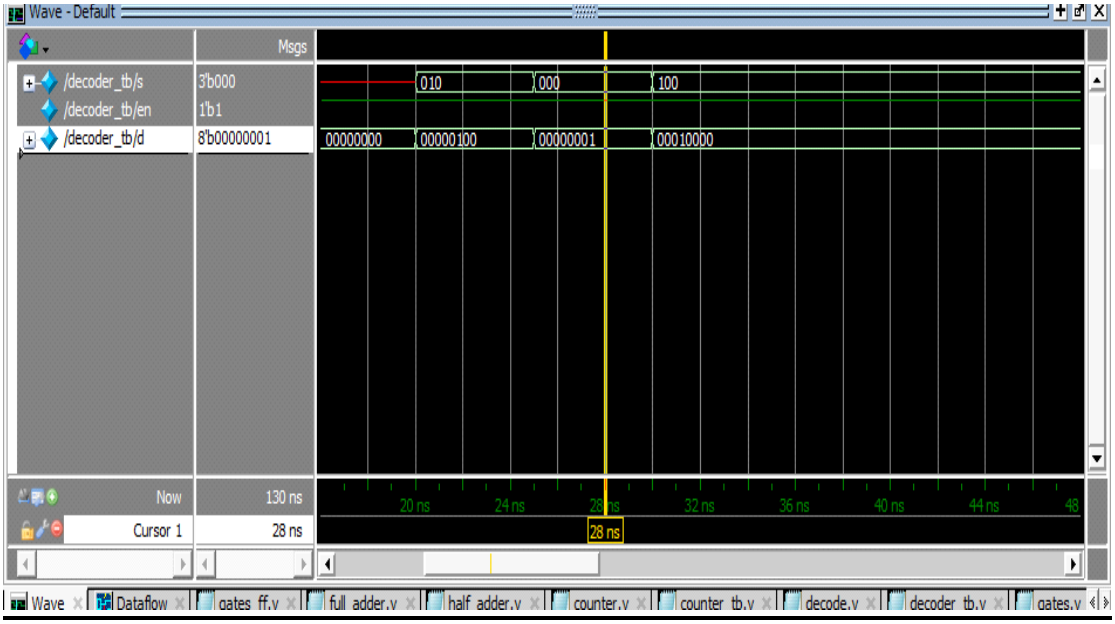
```
Ln#
1   module full_adder(
2   input a,b,cin,
3   output sum,carry_out
4   );
5   wire s_a,carry_t1,carry_t2;
6   half_adder HA1(.a(a),.b(b),.sum(s_a),.carry(carry_t1));
7   half_adder HA2(.a(s_a),.b(cin),.sum(sum),.carry(carry_t2));
8   assign carry_out=carry_t1|carry_t2;
9
10
11  endmodule
12
```

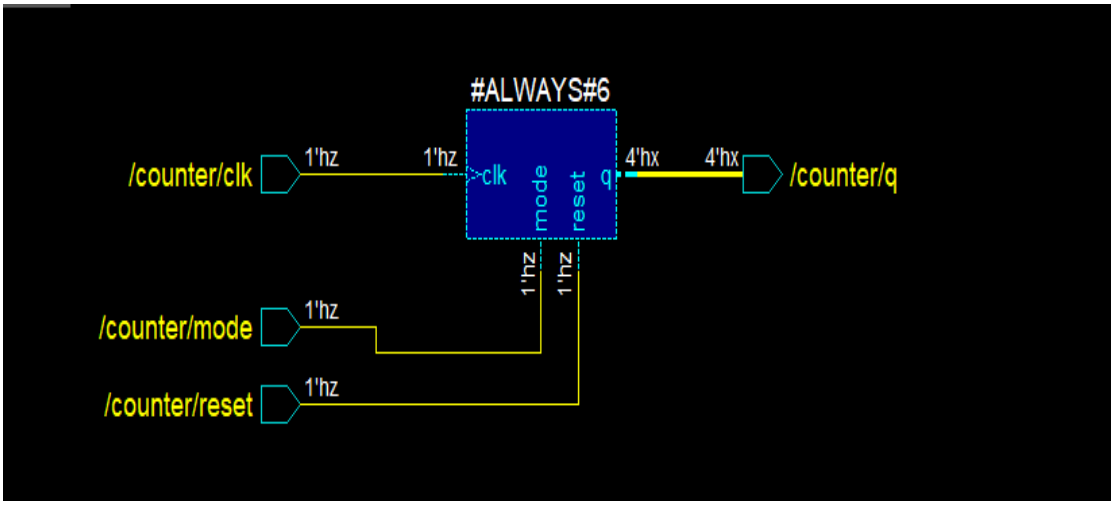


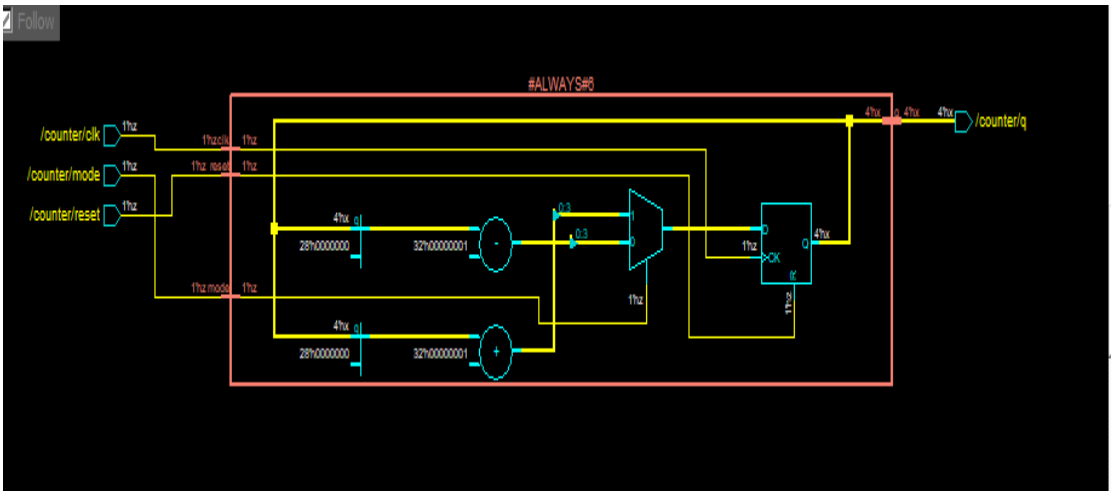
counter

```
module counter#(parameter n=4)(
input mode,clk,reset,
output reg [n-1:0]q
);

always@(posedge clk,posedge reset)
begin
if(reset)
q<=0;
//for 1 is up and 0 is down
else if(mode)
q<=q+1;
else
q<=q-1;
end

endmodule
```

```
1  module counter_tb();
2  parameter n=8;
3  reg clk=0,reset,mode;
4  wire[n-1:0]q;
5  counter #(.n(n))CA (.clk(clk),.reset(reset),.mode(mode),.q(q));
6  always #10 clk=~clk;
7  initial begin
8  mode=1;//up
9  reset=1'b1;
10 #10;
11 reset=0;
12 #100;
13 mode=0;
14 #100;
15 mode=01;
16 #300;
17 $finish;
18 end
19
20 endmodule
21
```







decoder

C:/Users/marrw/Desktop/decoderr/decoder_tb.v (/decoder_tb) - Default *

Ln#

18 ns

```
1  module decoder_tb();
2  reg [2:0]s;
3  reg en;
4  wire [7:0]d;
5  decode decoder(.en(en),.s(s),.d(d));
6  initial begin
7  en=0;
8
9  #10;
10 en=1;
11 #10;
12 s=3'b010;
13 #5;
14 s=3'b000;
15 #5;
16 s=3'b100;
17 $finish;
18 end
19 endmodule
20
```

C:/Users/marrw/Desktop/decoderr/decode.v (/decoder_tb/decoder) - Default

```
Ln#
 1  module decode(
 2  input [2:0]s,
 3  input en,
 4  output reg [7:0]d
 5  );
 6  always@*
 7  begin
 8  case({en,s})
 9  4'b0xxx:d=8'b00000000;
10  4'b1000:d=8'b00000001;
11  4'b1001:d=8'b00000010;
12  4'b1010:d=8'b00000100;
13  4'b1011:d=8'b00001000;
14  4'b1100:d=8'b00010000;
15  4'b1101:d=8'b00100000;
16  4'b1110:d=8'b01000000;
17  4'b1111:d=8'b10000000;
18  default:d=8'b00000000;
19  endcase
20  end
21  endmodule
```

Wave | Dataflow | alu_tb.v | decode.v | decoder_tb.v | alu.v

memory

C:/Users/marrw/Desktop/memory/memory.v (/memory_tb/memory1) - Default

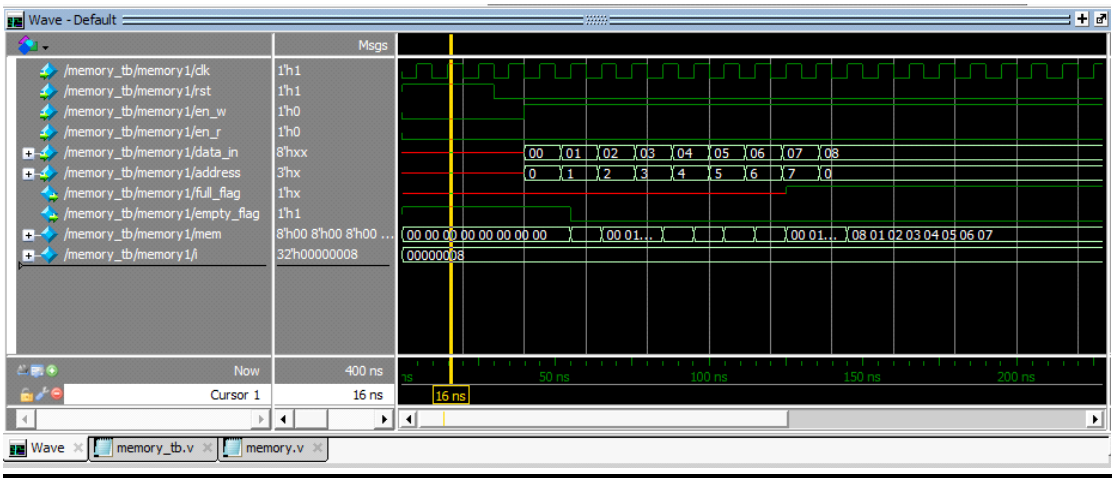
```
module memory(
input clk,rst,en_w,en_r,
input [7:0]data_in,
input [2:0]address,
output reg full_flag,empty_flag,
output reg [7:0]data_out
);

reg [7:0] mem[0:7];
integer i,j;


integer counter=0;
always@(posedge clk or posedge rst)
begin
if(rst)
begin

for(i=0;i<8;i=i+1)
  begin
```

```
for(i=0;i<8;i=i+1)
  begin
        mem[i]<='b0;
  end
        empty_flag=1;
end

else if(en_w)
begin
mem[address]<=data_in;
counter<=counter+1;
end

else if(en_r)
begin
data_out<=mem[address];
counter<=counter-1;
end
else
$display("at time %0t there is no read or write",$time);
```

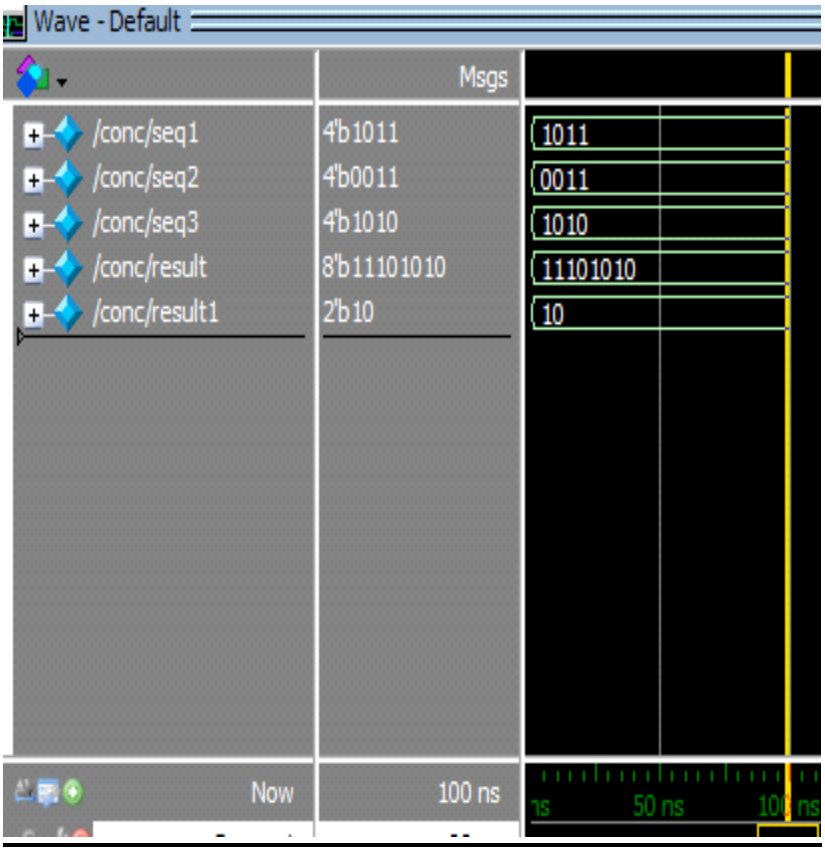
```
else
$display("at time %0t there is no read or write",$time);

if(counter==8)
full_flag=1;
else if(counter==0)
empty_flag=1;
else if (counter>0)
empty_flag=0;
else
$display("not empty not full");
end


endmodule
```

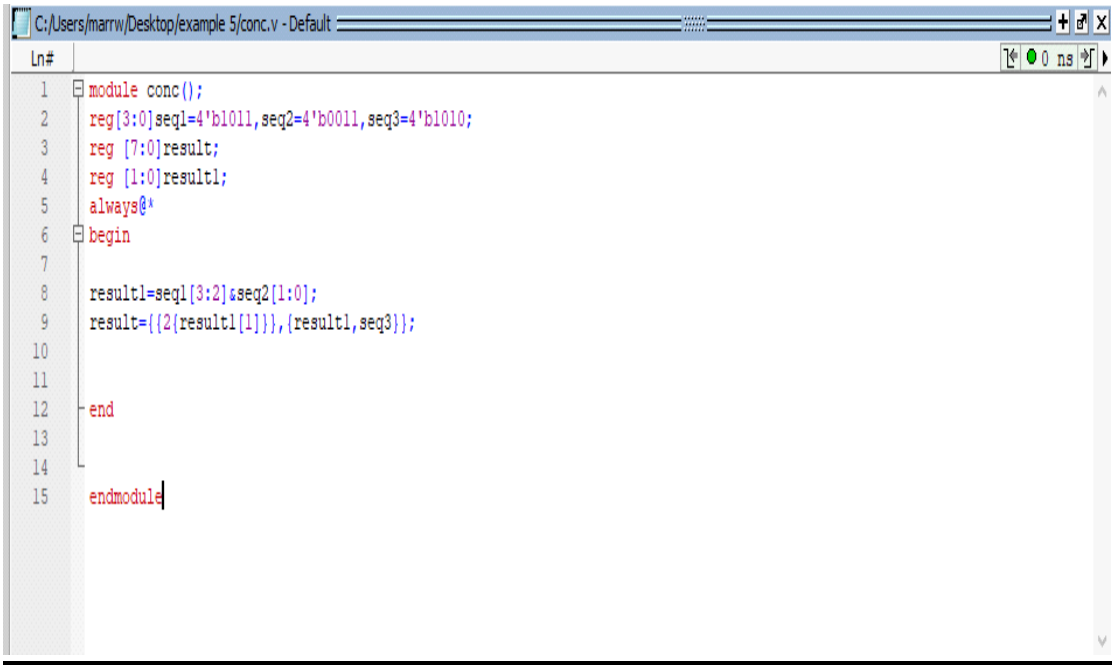
```
module memory_tb();
reg clk=0,rst,en_w,en_r;
reg [7:0]data_in;
reg [2:0]address;
wire full_flag,empty_flag;
wire [7:0]data_out;
memory memory1(.clk(clk),.rst(rst),.en_w(en_w),.en_r(en_r),.data_in(data_in),
.address(address),.full_flag(full_flag),.empty_flag(empty_flag),.data_out(data_out));
integer i=0;
always #5 clk=~clk;
initial begin
rst=1;
en_w=0;
en_r=0;
#30;
rst=0;
#10;
en_w=1;
for(i=0;i<8;i=i+1)
begin
```

```
20  begin
21  address=i;
22  data_in=i;
23  #12;
24  end
25
26  for(i=0;i<8;i=i+1)
27  begin
28  address=i;
29  data_out=mem[address];
30  #12;
31  end
32
33  end
34
35  endmodule
```



concatination


Wave - Default
Msgs
/conc/seq1
/conc/seq2
/conc/seq3
/conc/result
/conc/result1
4'b1011
4'b0011
4'b1010
8'b11101010
2'b10
1011
0011
1010
11101010
10
Now
100 ns
50 ns


C:/Users/marrw/Desktop/example 5/conc.v - Default
Ln#
module conc();
reg[3:0]seq1=4'b1011,seq2=4'b0011,seq3=4'b1010;
reg [7:0]result;
reg [1:0]result1;
always@*
begin
result1=seq1[3:2]&seq2[1:0];
result={{2{result1[1]}},{result1,seq3}};
end
endmodule

gates

C:/Users/marrw/Desktop/gates/gates.v - Default

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

wire s1,s2,s3;
assign s1=b^c;
assign s2=!b & c;
assign d=a^s1;
assign s3=a & !s1;
assign f=s2&s3;


endmodule
```

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


reg s1,s2;
wire s3;

assign s3=!s1 &a;
always@(posedge clk)
begin
s1<=b^c;
s2<= !b & c;
d<=a^s1;
f<=s2|s3;
end
```

```
always@(posedge clk)
begin
s1<=b^c;
s2<= !b & c;
d<=a^s1;
f<=s2|s3;
end

endmodule
```

alu

C:/Users/marrw/Desktop/alu/alu.v - Default *

```
Ln#
 1  module alu(
 2  input [2:0]a,b,
 3  input clk,
 4  input[3:0]opcode,
 5  output reg [5:0]result
 6  );
 7  always@(posedge clk)
 8  begin
 9  case(opcode)
10  4'b0000:result<=a+b;
11  4'b0001:result<=a*b;
12  4'b0011:result<=a&b;
13  4'b0100:result<=a|b;
14  4'b0101:result<=a^b;
15  4'b0110:result<=~(a&b);
16  4'b0111:result<=~(a|b);
17  4'b1000:result<=~(a^b);
18  4'b1001:result<=a<<1;
```

```
4'b0000:result<=a+b;
4'b0001:result<=a*b;
4'b0011:result<=a&b;
4'b0100:result<=a|b;
4'b0101:result<=a^b;
4'b0110:result<=~(a&b);
4'b0111:result<=~(a|b);
4'b1000:result<=~(a^b);
4'b1001:result<=a<<1;
4'b1010:result<=a>>1;
4'b1011:result<={a,b};
default:$display("at time=%0t opcode is =%0d thats not valid code",$time,opcode);
endcase
end
endmodule
```

```
Ln#
1  module alu_tb();
2  reg [2:0]a,b;
3  reg clk=0;
4  reg[3:0]opcode;
5  wire [6:0]result;
6  alu al(.a(a),.b(b),.clk(clk),.opcode(opcode),.result(result));
7  always #10 clk=~clk;
8  initial begin
9  a=3'b101;
10 b=3'b011;
11 #10;
12 opcode=3;
13 #15;
14 opcode=2;
15 #15;
16 opcode=15;
17 #15;
18 $finish;
19 end
20 endmodule
21
```