Sheet (2)

Q1-Write the HDL gate-level hierarchical description of a four-bit-adder-subtractor for unsigned binary numbers.

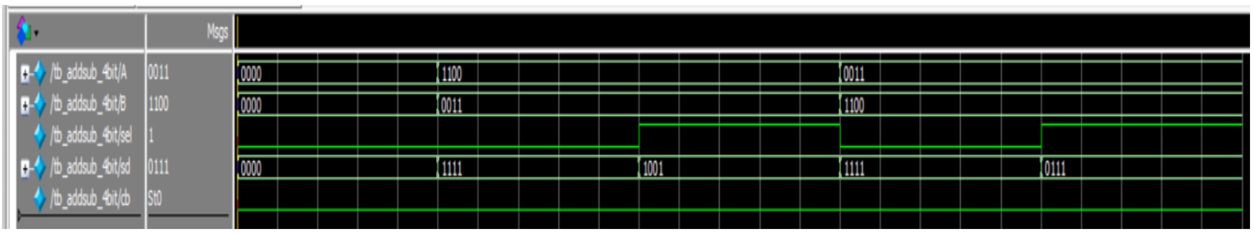
ANS

(a) Code:-

```
module H_A (output sum,carry,input A,B);
xor (sum,A,B);
and (carry,A,B);
endmodule

module F_A (output sum,carry,input A,B,C);
wire s1,c1,c2;
H_A U0 (s1,c1,A,B);
H_A U1 (sum,c2,s1,C);
or (carry,c1,c2);
endmodule

module adder_4bit (output [3:0] sum,output carry,input [3:0] A,B,input sel);
wire c0,c1,c2;
F_A inist1 (sum[0],c0,A[0],B[0],sel);
F_A inist2 (sum[1],c1,A[1],B[1],c0);
F_A inist3 (sum[2],c2,A[2],B[2],c1);
F_A inist4 (sum[3],carry,A[3],B[3],c2);
endmodule

module addsubt (output [3:0] sd,output cb,input[3:0] A,B,input sel);
wire [3:0]x;
wire det;
xor (x[0],sel,B[0]);
xor (x[1],sel,B[1]);
xor (x[2],sel,B[2]);
xor (x[3],sel,B[3]);
adder_4bit V0 (sd,det,A,x,sel);
and (cb,~sel,det);
endmodule

`timescale 1ns/1ps

module tb_addsub_4bit ();
reg [3:0] A,B;
reg sel;
wire [3:0] sd;
wire cb;
addsubt M (sd,cb,A,B,sel);
initial
begin
A=0;B=0;sel=0;
#10 A=4'b1100;B=4'b0011;
#10 sel=1;
```

```
#10 sel=0;A=4'b0011;B=4'b1100;
#10 sel=1;
end
endmodule
```

(b) Simulation:-



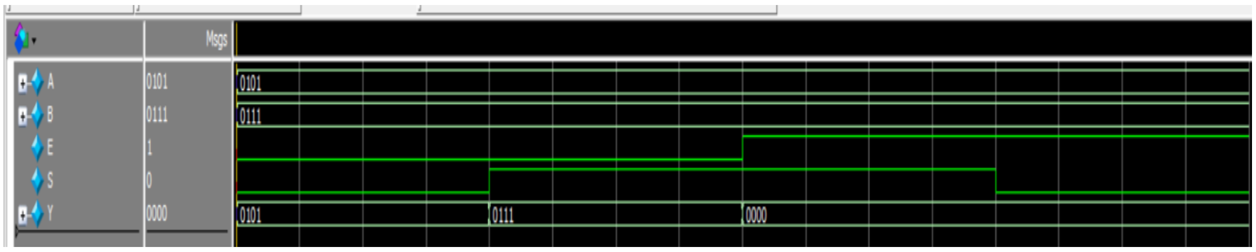
Q2-Write the HDL dataflow description of a quadruple 2-to-1 line multiplexer with enable.

ANS

(a) Code:-

```
module mux_2_1_quadruble (output reg [3:0] Y,input [3:0] A,B,input E,S);
always @ (E,S)
if (E==1)
Y=0;
else if (E==0 & S==0)
Y=A;
else
Y=B;
endmodule

`timescale 1ns/1ps

module tb_mux_2_1_quadruble ();
reg [3:0] A,B;
reg E,S;
wire [3:0] Y;
mux_2_1_quadruble M (Y,A,B,E,S);
initial
begin
A=0101;B=1111;E=0;S=0;
#20 S=1;
#20 E=1;
#20 S=0;
end
endmodule
```

Simulation:-



Q3-Using the conditional operator (?:),write an HDL dataflow description of a four-bit-adder-subtractor of unsigned numbers.

ANS

(a) Code:-

```
module con_oper_4_bit (output [3:0] sd,output cb,input [3:0] A,B,input sel);
assign {cb,sd}= (sel==0) ? A+B:A-B;
Endmodule
```

```
`timescale 1ns/1ps

module tb_addsub_4bit ();
reg [3:0] A,B;
reg sel;
wire [3:0] sd;
wire cb;
addsubt M (sd,cb,A,B,sel);
initial
begin
A=0;B=0;sel=0;
#10 A=4'b1100;B=4'b0011;
#10 sel=1;
#10 sel=0;A=4'b0011;B=4'b1100;
#10 sel=1;
end
endmodule
```

Q4-Repeating Q3 using always statement

ANS

(a) Code:-

```
module con_oper_4_bit (output reg [3:0] sd,output reg cb,input [3:0] A,B,input sel);
always @ (A,B,sel)
{cb,sd}= (sel==0) ? A+B:A-B;
Endmodule
```

```
`timescale 1ns/1ps

module tb_addsub_4bit ();
reg [3:0] A,B;
reg sel;
wire [3:0] sd;
wire cb;
addsubt M (sd,cb,A,B,sel);
initial
begin
A=0;B=0;sel=0;
#10 A=4'b1100;B=4'b0011;
#10 sel=1;
#10 sel=0;A=4'b0011;B=4'b1100;
#10 sel=1;
end
endmodule
```