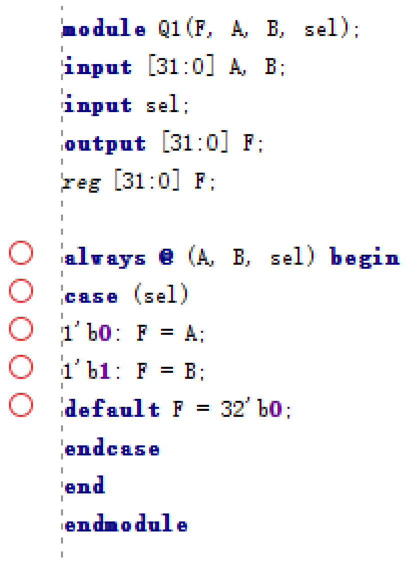
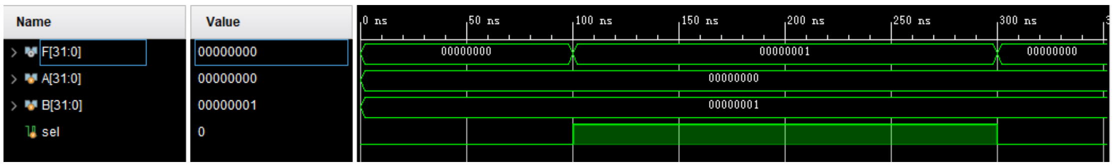
Q1

Verilog code:



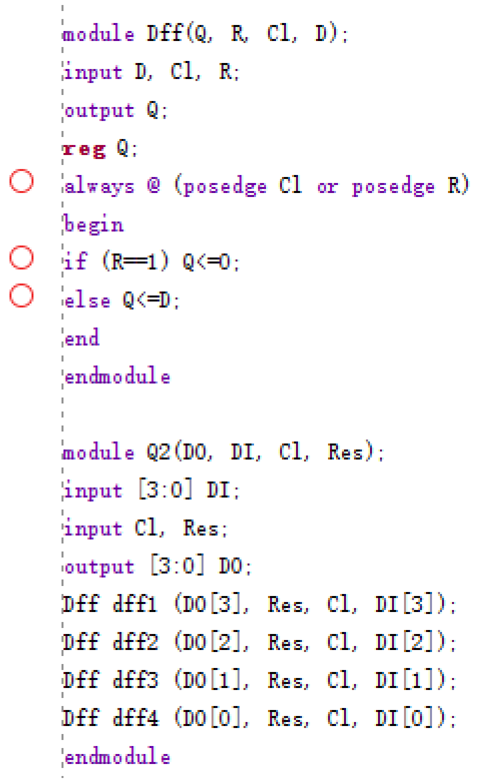
```
module Q1(F, A, B, sel);
input [31:0] A, B;
input sel;
output [31:0] F;
reg [31:0] F;

always @ (A, B, sel) begin
case (sel)
1'b0: F = A;
1'b1: F = B;
default F = 32'b0;
endcase
end
endmodule
```

Simulation result:



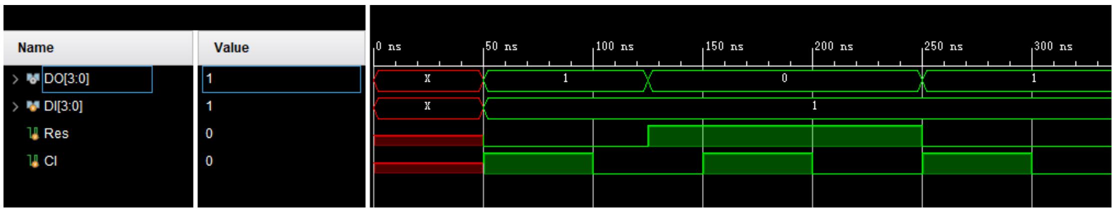
```
Name        Value
F[31:0]     00000000
A[31:0]     00000000
B[31:0]     00000001
sel         0
```

Q2
Verilog code:



```
module Dff(Q, R, Cl, D);
input D, Cl, R;
output Q;
reg Q;
always @ (posedge Cl or posedge R)
begin
if (R==1) Q<=0;
else Q<=D;
end
endmodule

module Q2(DO, DI, Cl, Res);
input [3:0] DI;
input Cl, Res;
output [3:0] DO;
Dff dff1 (DO[3], Res, Cl, DI[3]);
Dff dff2 (DO[2], Res, Cl, DI[2]);
Dff dff3 (DO[1], Res, Cl, DI[1]);
Dff dff4 (DO[0], Res, Cl, DI[0]);
endmodule
```

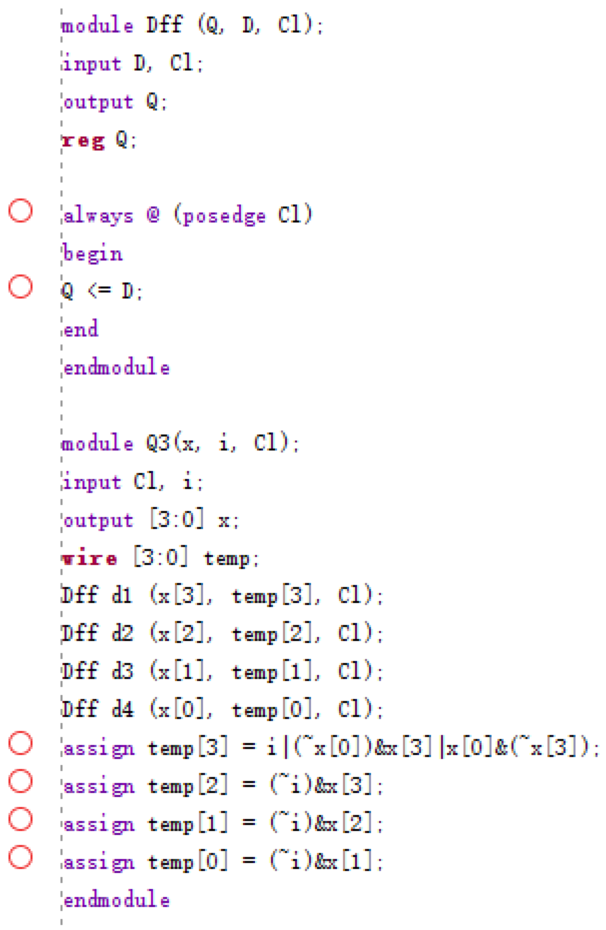
Simulation result:



Name	Value
DO[3:0]	1
DI[3:0]	1
Res	0
Cl	0

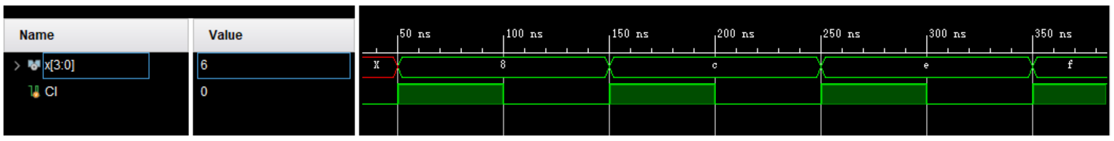
Q3

Verilog code:



```
module Dff (Q, D, Cl);
input D, Cl;
output Q;
reg Q;

always @ (posedge Cl)
begin
Q <= D;
end
endmodule

module Q3(x, i, Cl);
input Cl, i;
output [3:0] x;
wire [3:0] temp;
Dff d1 (x[3], temp[3], Cl);
Dff d2 (x[2], temp[2], Cl);
Dff d3 (x[1], temp[1], Cl);
Dff d4 (x[0], temp[0], Cl);
assign temp[3] = i|(~x[0])&x[3]|x[0]&(~x[3]);
assign temp[2] = (~i)&x[3];
assign temp[1] = (~i)&x[2];
assign temp[0] = (~i)&x[1];
endmodule
```

Simulation result:



```
Name      Value
x[3:0]    6
Cl        0
```

Q4

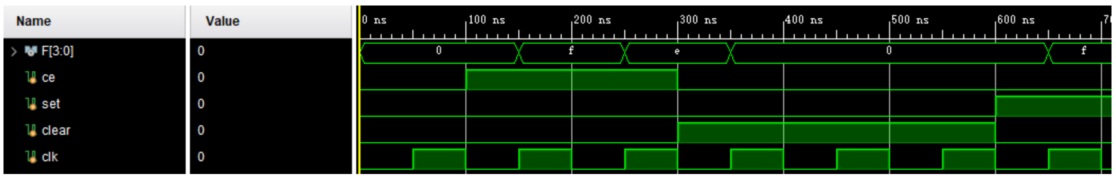
Verilog code:

```verilog
module MUX(out, i0, i1, sel);
input i0, i1;
input sel;
output out;
reg out;
always @ (i0, i1, sel) begin
case (sel)
1'b0: out=i0;
1'b1: out=i1;
default out=1'b0;
endcase
end
endmodule

module Dff (Q, D, Cl);
input D, Cl;
output Q;
reg Q;
always @ (posedge Cl) begin
Q <= D;
end
endmodule
```

```verilog
module Q4(F, ce, set, clear, clk);
input ce, set, clear, clk;
output [3:0] F;
reg [3:0] F;
wire [3:0]add;
wire a1, a2, a3, a4, a, b1, b2, b3, b4, b, c1, c2, c3, c, d1, d2, d3, d4, d;
initial begin
F = 4'b0;
end
assign add[3] = ((~F[3])&(~F[2])&(~F[1])&(~F[0]))|(F[3]&F[2])|(F[3]&F[1])|(F[3]&F[0]);
assign add[2] = ((~F[2])&(~F[1])&(~F[0]))|(F[2]&F[0])|(F[2]&F[1]);
assign add[1] = ((~F[1])&(~F[0]))|(F[1]&F[0]);
assign add[0] = (~F[0]);
MUX A1 (a1, F[3], add[3], ce);
MUX A2 (a2, a1, 1'b1, set);
MUX A3 (a3, a2, 1'b0, clear);
Dff A (a, a3, clk);
MUX B1 (b1, F[2], add[2], ce);
MUX B2 (b2, b1, 1'b1, set);
MUX B3 (b3, b2, 1'b0, clear);
Dff B (b, b3, clk);
```

```
MUX C1 (c1, F[1], add[1], ce);
MUX C2 (c2, c1, 1'b1, set);
MUX C3 (c3, c2, 1'b0, clear);
Dff C (c, c3, clk);
MUX D1 (d1, F[0], add[0], ce);
MUX D2 (d2, d1, 1'b1, set);
MUX D3 (d3, d2, 1'b0, clear);
Dff D (d, d3, clk);
always @ * begin
F[3] <= a;
F[2] <= b;
F[1] <= c;
F[0] <= d;
end
endmodule
```

Simulation result:



Q5

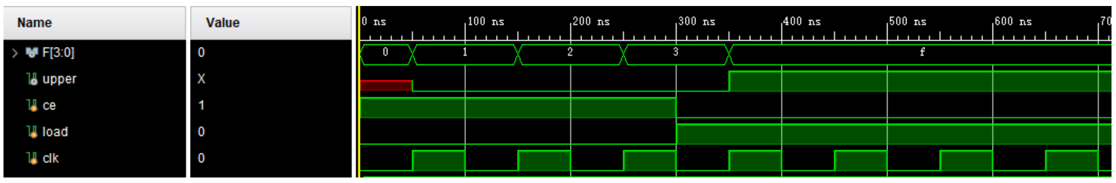
Verilog code:

```
module MUX(out, i0, i1, sel);
input i0, i1;
input sel;
output out;
reg out;
always @ (i0, i1, sel) begin
case (sel)
1'b0: out=i0;
1'b1: out=i1;
default out=1'b0;
endcase
end
endmodule

module Dff (Q, D, Cl);
input D, Cl;
output Q;
reg Q;
always @ (posedge Cl) begin
Q <= D;
end
endmodule
```

```
module Q5(F, upper, ce, load, clk);
input ce, load, clk;
output [3:0] F;
output upper;
reg upper;
reg [3:0] F;
wire a1, a2, a3, a4, a, b1, b2, b3, b4, b, c1, c2, c3, c, d1, d2, d3, d4, d;
wire [3:0] add;
initial begin
F=4'b0;
end
assign add[3]=((~F[3])&F[2]&F[1]&F[0])|(F[3]&(~F[2]))|(F[3]&F[1])|(F[3]&F[0]);
assign add[2]=(F[3]&(~F[2])&F[1]&F[0])|(F[2]&F[1])|(F[2]&F[0]);
assign add[1]=((~F[1])&F[0])|(F[1]&(~F[0]));
assign add[0]=~F[0];
MUX A1 (a1, F[3], add[3], ce);
MUX A2 (a2, a1, 1'b1, load);
Dff A (a, a2, clk);
MUX B1 (b1, F[2], add[2], ce);
MUX B2 (b2, b1, 1'b1, load);
Dff B (b, b2, clk);
```

```
MUX C1 (c1, F[1], add[1], ce);
MUX C2 (c2, c1, 1'b1, load);
Dff C (c, c2, clk);
MUX D1 (d1, F[0], add[0], ce);
MUX D2 (d2, d1, 1'b1, load);
Dff D (d, d2, clk);
always @ * begin
F[3] <= a;
F[2] <= b;
F[1] <= c;
F[0] <= d;
upper <= a;
end
endmodule
```

Simulation result:



Name
Value
F[3:0]
upper
ce
load
clk
0
X
1
0
0
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
0
1
2
3
f