SOURCE CODE FOR DESIGN OF COUNTER USING REVERSIBLE LOGIC

1)TOFFOLI GATE FOR 4 BIT COUNTER

```
module feynman_gate(a,b,p,q,reset);
input a, b, reset;
output reg p,q;
  always@(a)
      begin
    if (reset)
      begin
       p = 0;
       q = 0;
    end
    else begin
       p = a;
       q=a^b;
    end
    end
endmodule

module toffoli_gate(a, b, c, p, q, r, reset);
input a, b, c, reset;
output reg p, q, r;
  always@( b) begin
        if(reset) begin
            p = 0;
            q = 0;
            r = 0;
        end
```

```
        else begin
            p = a;
            q = b;
            r =(a&b) ^ c;
        end
    end
endmodule
module one_bit(t_i, clk, q0, reset);
input t_i, clk, reset;
output q0;
wire g1, g2, o1, en;
  toffoli_gate t1 (.a(t_i), .b(clk), .c(en) , .p(g1), .q(g2), .r(o1), .reset (reset));
  feynman_gate f1 (.a(o1), .b(1'b1), .p(q0), .q(en), .reset (reset));
endmodule
module four_bit(t_i, clk, q0, reset);
  input t_i, clk, reset;
  output [3:0]q0;
//wire g1, g2, o1, en;
  one_bit count1 (.t_i(1), .clk(clk), .q0(q0[0]), .reset(reset));
  one_bit count2 (.t_i(1), .clk(q0[0]), .q0(q0[1]), .reset(reset));
  one_bit count3 (.t_i(1), .clk(q0[1]), .q0(q0[2]), .reset(reset));
  one_bit count4 (.t_i(1), .clk(q0[2]), .q0(q0[3]), .reset(reset));
  endmodule
```

TESTBENCH

```
module tb();
wire [3:0]q0;
reg clk, t_i;
reg reset;
four_bit DUT (.t_i (t_i), .clk(clk), .q0(q0), .reset (reset) );
    initial begin
        reset = 1;
```

```
      t_i = 1;
      clk = 1;
     repeat (40) begin
        #10;
        reset = 0;
        clk = ~clk;
      end
      $finish();
   end
   initial
     $monitor ("VALUES OF clk=%0d t_i=%0d reset=%0d q0=%0d",clk,t_i,reset,q0);
Endmodule
```

2)PERES GATE

```
module feynman_gate(a,b,p,q,reset);
input a, b, reset;
output reg p,q;
  always@(a)
     begin
   if (reset)
     begin
      p = 0;
      q = 0;
   end
   else begin
      p = a;
      q=a^b;
   end
   end
endmodule

module peres_gate(a, b, c, p, q, r, reset);
```

```
input a, b, c, reset;
output reg p, q, r;
  always@( b) begin
        if(reset) begin
            p = 0;
            q = 0;
            r = 0;
        end
        else begin
            p = a;
          q = (a^b);
            r =(a&b) ^ c;
       end
    end
endmodule

module one_bit(t_i, clk, q0, reset);
input t_i, clk, reset;
output q0;
wire gl, g2, o1, en;
  peres_gate t1 (.a(clk), .b(t), .c(en) , .p(g1), .q(g2), .r(o1), .reset (reset));
  feynman_gate f1 (.a(o1), .b(1'b0), .p(q0), .q(en), .reset (reset));
endmodule

module four_bit(t_i, clk, q0, reset);
  input t_i, clk, reset;
  output [3:0]q0;
//wire gl, g2, o1, en;
  one_bit count1 (.t_i(clk), .clk(clk), .q0(q0[0]), .reset(reset));
  one_bit count2 (.t_i(q0[0]), .clk(1), .q0(q0[1]), .reset(reset));
  one_bit count3 (.t_i(q0[1]), .clk(1), .q0(q0[2]), .reset(reset));
  one_bit count4 (.t_i(q0[2]), .clk(1), .q0(q0[3]), .reset(reset));
```

```
endmodule


module tb();
wire [3:0]q0;
reg clk, t_i;
reg reset;
four_bit DUT (.t_i (t_i), .clk(clk), .q0(q0), .reset (reset) );
    initial begin
       reset = 1;
       t_i = 1;
       clk = 1;
      repeat (40) begin
          #10;
          reset = 0;
          clk = ~clk;
       end
       $finish();
    end
    initial
      $monitor ("VALUES OF clk=%0d t_i=%0d reset=%0d q0=%0d",clk,t_i,reset,q0);
Endmodule
```

3)FREDKIN GATE

```
module feynman(a,b,p,q,reset);
  input a,b,reset;
  output reg p,q;


  always@(*) begin
    if(reset) begin
      p = 0;
      q = 0;
    end
```

```
    else begin
      p = a;
      q = a ^ b;
    end
  end
endmodule

module fredkin(a,b,c,x,y,z,reset);
  input a,b,c,reset;
  output reg x,y,z;

  always@(*) begin
    if(reset) begin
      x = 0;
      y = 0;
      z = 0;
    end
    else begin
      x = a;
      y = (~a & b) + (a & c);
      z = (~a & c) + (a & b);
    end
  end
endmodule

module one_bit(clk,q,reset);
  input clk,reset;
  output q;
  wire fd11,fd12,fd13;
  wire fn11,fn12;
  wire fd21,fd22,fd23;
  wire fn21,fn22;
```

```
  wire fn32;


  fredkin fd1 (.a(clk) , .b(fn12), .c(fn32)    , .x(fd11), .y(fd12), .z(fd13), .reset(reset));
  feynman fn1 (.a(fd12), .b(0)   , .p(fn11), .q(fn12), .reset(reset));
  fredkin fd2 (.a(fd11), .b(fn11), .c(fn22)    , .x(fd21), .y(fd22), .z(fd23), .reset(reset));
  feynman fn2 (.a(fd22), .b(0)   , .p(fn21), .q(fn22), .reset(reset));
  feynman fn3 (.a(fn21), .b(1)   , .p(q)   , .q(fn32), .reset(reset));
endmodule

module four_bit(clk,q,reset);
  input clk,reset;
  output [3:0]q;
  one_bit o1 (.clk(clk), .reset(reset), .q(q[0]));
  one_bit o2 (.clk(q[0]), .reset(reset), .q(q[1]));
  one_bit o3 (.clk(q[1]), .reset(reset), .q(q[2]));
  one_bit o4 (.clk(q[2]), .reset(reset), .q(q[3]));
endmodule

module tb;
  reg clk,reset;
  wire [3:0]q;

  /*one_bit dut(.clk(clk), .q(q), .reset(reset));
  initial
    begin
      reset = 1;
      repeat(20) begin
        #10 clk = ~ clk;
        reset = 0;
      end
    end
```

```
  initial
    $monitor("CLK=%0d reset=%0d q=%0d",clk,reset,q);*/

  four_bit dut(.clk(clk), .q(q), .reset(reset));
  initial
    begin
      reset = 1;
      clk=0;
      #10;
      repeat(40) begin
        #10 clk = ~ clk;
            reset = 0;
      end
    end

  initial
    $monitor("CLK=%0d reset=%0d q=%0d",clk,reset,q);
Endmodule
```

4)SVS GATE

SOURCE CODE

```
module svs(a,b,c,x,y,z,reset);
  input a,b,c,reset;
  output reg x,y,z;

  always@(*) begin
    if(reset) begin
      x = 0;
      y = 0;
      z = 0;
    end
```

```
    else begin
      x = a;
      y = (~a & b) + (a & c);
      z = (~a & c) + (a & b);
    end
  end
endmodule
module feynman(a,b,p,q,reset);
  input a,b,reset;
  output reg p,q;

  always@(*) begin
    if(reset) begin
      p = 0;
      q = 0;
    end
    else begin
      p = a;
      q = a ^ b;
    end
  end
endmodule

module one_bit(clk,q,reset);
  input clk,reset;
  output q;
  wire fd11,fd12,fd13;
  wire fn11,fn12;
  wire fd21,fd22,fd23;
  wire fn21,fn22;
  wire fn32;
```

```
  svs fd1 (.a(clk) , .b(fn12), .c(fn32)    , .x(fd11), .y(fd12), .z(fd13), .reset(reset));
  feynman fn1 (.a(fd12), .b(0)   , .p(fn11), .q(fn12), .reset(reset));
  svs fd2 (.a(fd11), .b(fn11), .c(fn22)    , .x(fd21), .y(fd22), .z(fd23), .reset(reset));
  feynman fn2 (.a(fd22), .b(0)   , .p(fn21), .q(fn22), .reset(reset));
  feynman fn3 (.a(fn21), .b(1)   , .p(q)   , .q(fn32), .reset(reset));
endmodule

module four_bit(clk,q,reset);
  input clk,reset;
  output [3:0]q;
  one_bit o1 (.clk(clk), .reset(reset), .q(q[0]));
  one_bit o2 (.clk(q[0]), .reset(reset), .q(q[1]));
  one_bit o3 (.clk(q[1]), .reset(reset), .q(q[2]));
  one_bit o4 (.clk(q[2]), .reset(reset), .q(q[3]));
endmodule
```

TESTBENCH

```
module tb;
  reg clk,reset;
  wire [3:0]q;

  /*one_bit dut(.clk(clk), .q(q), .reset(reset));
  initial
    begin
      reset = 1;
      repeat(20) begin
        #10 clk = ~ clk;
        reset = 0;
```

```
    end
  end

 initial
  $monitor("CLK=%0d reset=%0d q=%0d",clk,reset,q);*/

 four_bit dut(.clk(clk), .q(q), .reset(reset));
 initial
  begin
    reset = 1;
    clk=0;
    #10;
    repeat(40) begin
      #10 clk = ~ clk;
          reset = 0;
    end
  end

 initial
  $monitor("CLK=%0d reset=%0d q=%0d",clk,reset,q);

Endmodule
```

5)SAYEM GATE

```
module sayem_gate(a,b,c,d,p,q,r,s,reset);
 input a,b,c,d,reset;
 output reg p,q,r,s;

 always@(*) begin
  if(reset) begin
    p = 0;
    q = 0;
```

```
      r = 0;
      s = 0;
    end
    else begin
      p = a;
      q = ((~a&b) ^ (a&c));
      r = ((~a&b) ^ (a&c) ^ (d));
      s = ((a&b) ^ (~a&c) ^ (d));
    end
  end
 endmodule

module feynman_gate(a,b,p,q,reset);
  input a,b,reset;
  output reg p,q;

  always@(*) begin
    if(reset) begin
      p = 0;
      q = 0;
    end
    else begin
      p = a;
      q = a ^ b;
    end
  end
endmodule

module one_bit(clk,q,reset);
  input clk,reset;
  output q;
  wire s1op,s1oq,s1or,s1os;
```

```
  wire s2op,s2oq,s2or,s2os;
  wire fnoq;

  sayem_gate s1 (.a(clk) , .b(s1or), .c(fnoq), .d(0), .p(s1op), .q(s1oq), .r(s1or), .s(s1os), .reset(reset));
  sayem_gate s2 (.a(s1op), .b(s1oq), .c(s2or), .d(0), .p(s2op), .q(s2oq), .r(s2or), .s(s2os), .reset(reset));
  feynman_gate f1 (.a(s2oq), .b(1), .p(q), .q(fnoq), .reset(reset));
endmodule

module four_bit(clk,q,reset);
  input reset,clk;
  output [3:0]q;

  one_bit o1 (.clk(clk), .reset(reset), .q(q[0]));
  one_bit o1 (.clk(q[0]), .reset(reset), .q(q[1]));
  one_bit o1 (.clk(q[1]), .reset(reset), .q(q[2]));
  one_bit o1 (.clk(q[2]), .reset(reset), .q(q[3]));

endmodule

module tb;
  bit clk,reset;
  wire [3:0]q;

  /*one_bit dut (.clk(clk), .q(q), .reset(reset));

  initial begin
    reset = 1;
    #10;
    repeat(10) begin
      reset = 0;
      #10 clk = ~clk;
    end
```

```
  end
  initial
    $monitor("VALUES OF Reset=%0d clk=%0d q=%0d",reset,clk,q);*/


    four_bit dut (.clk(clk), .q(q), .reset(reset));


  initial begin
    reset = 1;
    #10;
    repeat(40) begin
      reset = 0;
      #10 clk = ~clk;
    end
  end
  initial
    $monitor("VALUES OF Reset=%0d clk=%0d q=%0d",reset,clk,q);
endmodule
```