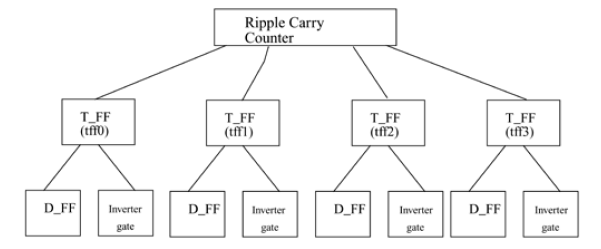
**Ripple carry counter (structural modelling)**



// counter. It instantiates 4 T-flipflops.

// 4-bit Ripple Carry Counter.

module ripple\_carry\_counter(q, clk, reset);

output [3:0] q; //I/O signals and vector declarations

input clk, reset; //I/O signals will be explained later.

//Four instances of the module T\_FF are created. Each has a unique

//Each instance is passed a set of signals. t

//each instance is a copy of the module T\_FF.

T\_FF tff0(q[0],clk, reset);

T\_FF tff1(q[1],q[0], reset);

T\_FF tff2(q[2],q[1], reset);

T\_FF tff3(q[3],q[2], reset);

endmodule

module T\_FF(q, clk, reset);

output q;

input clk, reset;

wire d;

D\_FF dff0(q, d, clk, reset); // Instantiate D\_FF. Call it dff0.

not n1(d, q); // not gate is a Verilog primitive

endmodule

// module D\_FF with synchronous reset

module D\_FF(q, d, clk, reset);

output q;

input d, clk, reset;

reg q;

// Concentrate on how the design block.

always @(posedge reset or negedge clk)

if (reset)

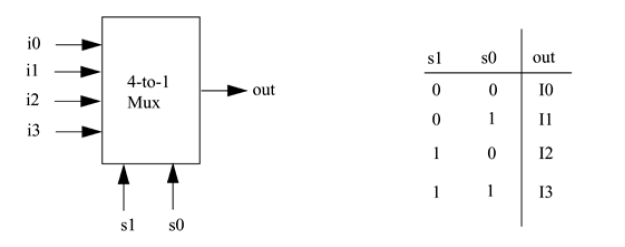
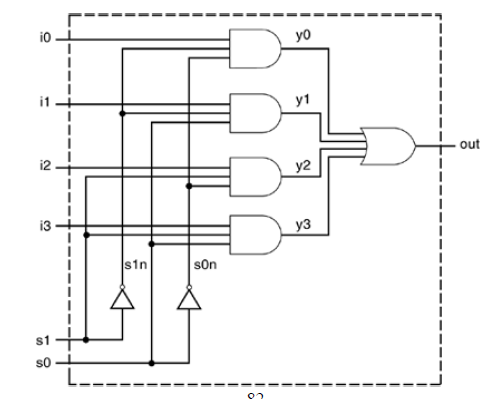
q <= 1'b0;

else

q <= d;

endmodule

**4:1 mux(gatelevel modelling**

// Module 4-to-1 multiplexer. Port list is taken exactly from

// the I/O diagram.

module mux4\_to\_1 (out, i0, i1, i2, i3, s1, s0);

// Port declarations from the I/O diagram

output out;

input i0, i1, i2, i3;

input s1, s0;

// Internal wire declarations

wire s1n, s0n;

wire y0, y1, y2, y3;

// Gate instantiations

// Create s1n and s0n signals.

not (s1n, s1);

not (s0n, s0);

// 3-input and gates instantiated

and (y0, i0, s1n, s0n);

and (y1, i1, s1n, s0);

and (y2, i2, s1, s0n);

and (y3, i3, s1, s0);

// 4-input or gate instantiated

or (out, y0, y1, y2, y3);

endmodule