**PDSD PRACTICAL VERILOG PROGRAMS**

**1.LOGIC GATES USING VERILOG**

**AND**

**module and\_123 (a,b,y);**

**input a,b;**

**output y;**

**assign y=a&b;**

**endmodule**

**OR GATE**

**module or\_123 (a,b,y);**

**input a,b;**

**output y;**

**assign y=a|b;**

**endmodule**

**NOR GATE**

**module nor\_123 (a,b,y);**

**input a,b;**

**output y;**

**assign y=~(a|b);**

**endmodule**

**XOR GATE**

**module xor\_123 (a,b,y);**

**input a,b;**

**output y;**

**assign y=a^b;**

**endmodule**

**NAND**

**HALF ADDER**

**module half\_adder123(a,b,s,c);**

**input a,b;**

**output s,c;**

**assign s=a^b;**

**assign c=a&b;**

**endmodule**

**FULL ADDER**

**module fulladd\_123(a,b,c,sum,carry);**

**input a,b,c;**

**output sum,carry;**

**assign sum=a^b^c;**

**assign carry=(a&b)|(b&c)|(c&a);**

**endmodule**

**HALF SUB**

**module halfsub\_123 (a,b,d,br);**

**input a,b;**

**output d,br;**

**wire d,br;**

**assign d=a^b;**

**assign br=~a&b;**

**endmodule**

**FULL SUB**

**module fullsub\_123(a,b,bin,diff,bout);**

**input a,b,bin;**

**output diff,bout;**

**wire diff,bout;**

**assign diff=bin^a^b;**

**assign bout=(~a&bin)|(~a&b)|(b&bin);**

**endmodule**

**[1.4][4:1] DECODER and ENCODER**

**module decoder\_123(a,b,c,d0,d1,d2,d3,d4,d5,d6,d7);**

**input a,b,c;**

**output d0,d1,d2,d3,d4,d5,d6,d7;**

**assign d0=(~a&~b&~c),d1=(~a&~b&c),d2=(~a&b&~c),d3=(~a&b&c),d4=(a&~b&~c),d5=(a&~b&c),d6=(a&b&~c),d7=(a&b&c);**

**endmodule**

**module encoder\_123 (a,b,c,d0,d1,d2,d3,d4,d5,d6,d7);**

**input d0,d1,d2,d3,d4,d5,d6,d7;**

**output a,b,c;**

**or(a,d4,d5,d6,d7);**

**or(b,d2,d3,d6,d7);**

**or(c,d1,d3,d5,d7);**

**endmodule**

**MULTIPLEXER**

**Multiplexer:**

**module mul\_1( a, b, c, d, s0, s1, out);**

**input wire a, b, c, d;**

**input wire s0, s1;**

**output reg out;**

**always @ (a or b or c or d or s0, s1)**

**begin**

**case (s0 | s1)**

**2'b00 : out <= a;**

**2'b01 : out <= b;**

**2'b10 : out <= c;**

**2'b11 : out <= d;**

**endcase**

**end**

**endmodule**

**DEMULTIPLEXER:**

**module demultiplexer1to4case(d0,d1,d2,d3,s0,s1,y);**

**input s0,s1,y;**

**output d0,d1,d2,d3;**

**reg d0,d1,d2,d3;**

**always@(s0,s1,y);**

**begin**

**case({s1,s0})**

**2'b00:begin d0<=y;d1<=0;d2<=0;d3<=0;end**

**2'b01:begin d0<=0;d1<=y;d2<=0;d3<=0;end**

**2'b10:begin d0<=0;d1<=0;d2<=y;d3<=y;end**

**2'b11:begin d0<=0;d1<=0;d2<=0;d3<=y;end**

**endcase**

**endmodule**

**SIPO:**

**module sipo\_123(C, SI, PO);**

**input C,SI;**

**output [7:0] PO;**

**reg [7:0] tmp;**

**always @(posedge C)**

**begin**

**tmp = {tmp[6:0], SI};**

**end**

**assign PO = tmp;**

**endmodule**

**SISO:**

**module siso\_123(input shift\_in,input clock,output shift\_out);**

**reg bit0;**

**reg bit1;**

**reg bit2;**

**reg bit3;**

**assign shift\_out=bit3;**

**always@(posedge clock)begin**

**bit3<=bit2;**

**bit2<=bit1;**

**bit1<=bit0;**

**bit0<=shift\_in;**

**end**

**endmodule**

**PISO:**

**module piso\_123(din,clk,load,dout);**

**output reg dout;**

**input [15:0]din;**

**input clk;**

**input load;**

**reg [15:0]temp;**

**always@(clk or load)begin**

**if(load)**

**temp<=din;**

**else begin**

**dout<=temp[0];**

**temp<={1'b0,temp[15:1]};**

**end**

**end**

**endmodule**

**RIPLLE CARRY**

**8 Bit ripple carry adder:**

**module ripple\_8\_bit(a, b, cin, sum, cout);**

**input [07:0] a;**

**input [07:0] b;**

**input cin;**

**output [7:0]sum;**

**output cout;**

**wire[6:0] c;**

**fulladd a1(a[0],b[0],cin,sum[0],c[0]);**

**fulladd a2(a[1],b[1],c[0],sum[1],c[1]);**

**fulladd a3(a[2],b[2],c[1],sum[2],c[2]);**

**fulladd a4(a[3],b[3],c[2],sum[3],c[3]);**

**fulladd a5(a[4],b[4],c[3],sum[4],c[4]);**

**fulladd a6(a[5],b[5],c[4],sum[5],c[5]);**

**fulladd a7(a[6],b[6],c[5],sum[6],c[6]);**

**fulladd a8(a[7],b[7],c[6],sum[7],cout);**

**endmodule**

**module fulladd(a, b, cin, sum, cout);**

**input a,b,cin;**

**output sum,cout;**

**assign sum=(a^b^cin);**

**assign cout=((a&b)|(b&cin)|(a&cin));**

**endmodule**