|  |  |
| --- | --- |
| 32bit Full\_Adder:  module fadder32bit(sum,cout,a,b,cin,clk,rst);  input [31:0]a,b;  inputcin,clk,rst;  outputreg[31:0]sum;  outputregcout;  reg [31:0]ai,bi;  reg ci;  always @(posedgeclk)begin  ai<=a; bi<=b;  ci<=cin;  if(~rst)begin  sum<=0;  cout<=0;  end  else  {cout,sum}<=ai+bi+ci;  end  endmodule | Test Bench:  module fadder32bit\_tb;  reg [31:0]a,b;  regcin,clk,rst;  wire [31:0]sum;  wirecout;  fadder32bit m2(sum,cout,a,b,cin,clk,rst);  initial begin  a=0;b=0;cin=0;clk=0;rst=0;  #10 rst=1;  #21000 $finish;  end  always #10 clk=~clk;  always #20 a=a+1;  always #40 b=b+1;  always #80 cin=~cin;  endmodule |

|  |  |
| --- | --- |
| Multiplier: mult32bit  module mult32bit(clk,rst,a,b,cout);  input [31:0]a,b;  inputclk,rst;  outputreg [63:0]cout;  reg [31:0]ai,bi;  always @(posedgeclk)begin  ai<=a;  bi<=b;  if (~rst)  cout<= 0;  else  cout<=ai\*bi;  end  endmodule | Test Bench:  Test\_bench:  module mult32bit\_tb;  reg [31:0]a,b;  regclk,rst;  wire [63:0]cout;  mult32bit stage1(clk,rst,a,b,cout);  initial begin  clk = 0; rst = 0; a=0;b=0;  #20 rst =1 ;  #3000 $finish;  end  always #5 clk=~clk;  always #10 a=a+1;  always #20 b=b+1;  endmodule |

|  |  |
| --- | --- |
| **Subtractor:** 32bit  module sub32bit(clk,rst,a,b,cout,sub);  input [31:0]a,b;  inputclk,rst;  outputreg [31:0]sub;  outputregcout;  reg [31:0]ai,bi;  always @(posedgeclk) begin  ai<=a;  bi<=b;  if (~rst)  {sub,cout}<= 0;  else  {sub,cout}<=ai-bi;  end  endmodule | **Test\_Bench:**  module sub32bit\_tb;  reg [31:0]a,b;  regclk,rst;  wire [31:0]sub;  wirecout;  sub32bit stage1(clk,rst,a,b,cout,sub);  initial begin  clk=0; rst=0; a=0;b=0;  #10 rst =1;  #3000 $finish;  end  always #10 clk=~clk;  always #20 a=a+1;  always #40 b=b+1;  endmodule |

|  |  |
| --- | --- |
| Univer\_Shift\_Reg:  module usr32bit(cout,clk,rst,load,shift\_lt\_rt,a);  outputreg [31:0]cout;  input load;  input [1:0]shift\_lt\_rt;  input [31:0]a; inputclk,rst;  reg [31:0]ai;  always @(posedgeclk or posedgerst)begin  if (rst)  cout = 0;  else  case(load)  1'b1:  begin  ai = a;  end  1'b0:  case (shift\_lt\_rt)  2'b00: cout = ai<<1;  2'b01: cout = ai>>1;  endcase  endcase  end  endmodule | Test Bench:  module usr32bit\_tb;  reg [31:0]a;  reg [1:0]shift\_lt\_rt;  regload,rst,clk;  wire [31:0] cout;  usr32bit inst1(cout,clk,rst,load,shift\_lt\_rt,a);  initial begin  a=0; shift\_lt\_rt=0; load=0; clk=0; rst=0;  #20 rst =1;  #6000 $finish;  end  always #50 clk=~clk;  always #100 load=~load;  always #150 a=~a;  always #200 shift\_lt\_rt=shift\_lt\_rt+1;  endmodule |

|  |  |
| --- | --- |
| ALU 32bit:  module alu32bit(cout,a,b,sel,clk,rst);  input [31:0]a,b;  input [1:0]sel;  inputclk,rst;  outputreg[31:0]cout;  always @(posedgeclk) begin  if(~rst)  cout<=0;  else begin  case (sel)  0: cout<= a & b;  1: cout<= a | b;  2: cout<= a>>1;  3: cout<= a<<1;  endcase  end  end  endmodule | Test\_Bench:  module alu32bit\_tb;  reg [31:0]a,b;  reg [1:0]sel;  regrst,clk;  wire [31:0]cout;  alu32bit inst1(cout,a,b,sel,clk,rst);  initial begin  a=0; b=0; sel=0; clk=0; rst=0;  #10 rst=1; a=7; b=5;  #20 a=7; b=5;  #30 sel=1; a=7; b=5;  #40 sel=2; a=7; b=5;  #50 sel=3; a=7; b=5;  #2000 $finish;  end  always #5 clk=~clk;  endmodule |

|  |  |
| --- | --- |
| **8bit Counter( Verilog Code)**  module cntr8bit(cnt,clk,rst);  parameter n=8;  outputreg[n-1:0]cnt;  inputclk;  inputrst;  initial  cnt=0;  always @(posedgeclk or negedgerst)  if (!rst)  cnt=0;  else  cnt=cnt+1;  endmodule | **Test\_Bench:**  module cntr8bit\_tb;  reg clk; reg rst;  wire [7:0] cnt;  cntr8bit g(  .cnt(cnt),  .clk(clk),  .rst(rst) );  initial begin  clk = 0; rst= 1;  #100 rst = 0;  #200 rst = 1;  #500 $finish;  end  always  #10 clk= !clk;  Endmodule |

|  |  |
| --- | --- |
| **Halfadder:**  module halfadder(s,c,a,b);  output s,c;  input a,b;  xor g1(s,a,b);  and g2(c,a,b);  endmodule | Test\_Bench:  module halfadder\_tb;  wire s,c;  rega,b;  halfadderinst(s,c,a,b);  initial begin  a=0;b=0;  #10 a=0;b=1;  #10 a=1;b=0;  #10 a=1;b=1;  #10 $finish;  end  endmodule |

|  |  |
| --- | --- |
| Fulladder:  module fulladder(s,c0,a,b,ci);  output s,c0;  input a,b,ci;  wire c1,c2,s1;  halfadder g1(s1,c1,a,b);  halfadder g2(s,c2,s1,ci);  or g3(c0,c1,c2);  endmodule | Test\_Bench:  module fulladder\_tb;  wire s,c0;  rega,b,ci;  fulladder inst1(s,c0,a,b,ci);  initial begin  a=0; b=0;ci=0;  #10 a=0;b=0;ci=1;  #10 a=0;b=1;ci=0;  #10 a=0;b=1;ci=1;  #10 a=1;b=0;ci=0;  #10 a=1;b=0;ci=1;  #10 a=1;b=1;ci=0;  #10 a=1;b=1;ci=1;  end  endmodule |

|  |  |
| --- | --- |
| **4bit\_Adder:**  module fa4bit(s,cout,cin,x,y);  parameter n=4;  input cin;  input[n-1:0]x,y;  output cout;  output[n-1:0]s;  wire[n:0]c;  genvar k;  assign c[0]=cin;  assign cout=c[n];  generate  for(k=0;k<=n-1;k=k+1)  begin:addbit  fulladder stage(s[k],c[k+1],c[k],x[k],y[k]);  end  endgenerate  endmodule | Tesr\_Bench:  module fa4bit\_tb;  regcin;  reg[3:0]x;  reg[3:0]y;  wire[3:0]s;  wire cout;  fa4bit inst(  .cin(cin),  .x(x),  .y(y),  .cout(cout),  .s(s) );  initial begin  cin = 0;  x = 0;  y = 0;  #2560 $finish;  end  always #5 cin=~cin;  initial begin  repeat(16)begin  repeat(16)  #10 y=y+1;  x=x+1;  end  end  endmodule |

|  |  |
| --- | --- |
| **2 to 1 mux (Verilog code) [Structure model]**  module mux21(f,s,i0,i1);  input s,i0,i1;  output f;  wire w1,w2,s\_bar;  and g2(w1,i1,s);  and g3(w2,i0,s\_bar);  not g1(s\_bar,s);  or g4(f,w1,w2);  endmodule | Test bench  module mux21\_tb;  reg s,i0,i1;  wire f;  mux21 inst1(f,s,i0,i1);  initial  begin s=1'b0; i0=1'b0; i1=1'b0;  #10 i0=1'b1; i1=1'b0;  #10 i0=1'b0; i1=1'b1;  #10 i0=1'b1; i1=1'b1;  #10 s=1'b1; i0=1'b0; i1=1'b0;  #10 i0=1'b1; i1=1'b0;  #10 i0=1'b0; i1=1'b1;  #10 i0=1'b1; i1=1'b1;  end  initial #200 $finish;  endmodule |

|  |  |  |  |
| --- | --- | --- | --- |
| Counter 8bit | Counter 8bit test bench |  |  |
| module cntr8bit(count,rst,clk);  input rst,clk;  output reg [7:0]count;  always @(posedge clk) begin  if(~rst)  count<=0;  else  count<=count+1;  end  endmodule | module cntr8bit\_tb;  reg rst,clk;  wire [7:0]count;  cntr8bit stage1(count,rst,clk);  initial begin  clk=0;rst=0;  #15 rst = 1;  #3000 rst=0;  #10 $finish;  end  always #5 clk=~clk;  //always #10 rst=~rst;  endmodule |  |  |

|  |  |
| --- | --- |
| module adder\_8bit\_rtl(clk,rst,a,b,cin,s,cout);  input [7:0]a,b;  inputcin,clk,rst;  outputregcout;  outputreg [7:0]s;  reg [7:0]A,B;  reg CI;  always @(posedgeclk) begin  A<=a;  B<=b;  CI<=cin;  if (~rst) begin  s <= 0;  cout<= 0;  end  else  {cout,s} <= A+B+CI;  end  endmodule | module adder\_8bit\_rtl\_tb;  reg [7:0]a,b;  regcin,clk,rst;  wirecout;  wire [7:0]s;  adder\_8bit\_rtl stage1(clk,rst,a,b,cin,s,cout);  initial begin  clk = 0; rst = 0; cin=0; a=0;b=0;  #10 rst =1 ;  #10 cin=1;  #3000 $finish;  end  always #5 clk=~clk;  initial begin  repeat( 2 ) begin  repeat( 256) begin  repeat( 256 )  #10 b = b + 1;  a= a + 1;  end  cin = cin + 1;  end  end  endmodule |

|  |  |
| --- | --- |
| module cntr8bit(cnt,clk,rst);  parameter n=8;  output reg [n-1:0]cnt;  input clk;  input rst;  initial  cnt=0;  always @(posedge clk or negedge rst)  if (!rst)  cnt=0;  else  cnt=cnt+1;  endmodule | module cntr8bit\_tb;  reg clk;  reg rst;  wire [7:0]cnt;  cntr8bit g(  .cnt(cnt),  .clk(clk),  .rst(rst));  initial begin  clk=0; rst=1;  #100 rst=0;  #200 rst =1;  #500 $finish;  end  always  #10 clk=!clk;  endmodule |

|  |  |
| --- | --- |
| module mux21(y, sel, i1, i0);  output reg y;  input sel, i1, i0;    always @ (sel, i1, i0)  if(sel)  y <= i1;  else  y <= i0;  endmodule  module mux41(y, s0, s1, i0, i1, i2, i3);  output y;  input s0, s1, i0, i1, i2, i3;    mux21 m1 (w3, s0, i1, i0);  mux21 m2 (w4, s0, i3, i2);  mux21 m3 (y, s1, w4, w3);    endmodule  module mux81(y, s0, s1, s2, i0, i1, i2, i3, i4, i5, i6, i7);  output y;  input i0, i1, i2, i3, i4, i5, i6, i7;  input s0, s1, s2;    mux41 m1 (w3, s0, s1, i0, i1, i2, i3);  mux41 m2 (w4, s0, s1, i4, i5, i6, i7);  mux21 m3 (y, s2, w4, w3);  endmodule    module com ( a,b,p,q,r,clk,rst);  input [31:0] a,b;  reg [31:0]a1,b1;  input clk,rst;  output reg p, q, r;  always @ (posedge clk)  begin  a1<=a;  b1<=b;  if(rst!=0)  begin  p<=0;  q<=0;  r<=0;  end    else  begin    p <= ( a1 > b1 )? 1'b1 : 1'b0;  q <= ( a1 < b )? 1'b1 : 1'b0;  r <= ( a1 == b1)? 1'b1 : 1'b0;  end  end    endmodule  odule decode32 (out,a,clk,rst);  input[4:0]a;  input clk,rst;  output reg[31:0]out;  reg[4:0]ai;  integer i;  always@(posedge clk)  begin  ai<=a;  if(~rst)  begin  out<=0;  end  else    for(i=0;i<32;i=i+1)  begin  out[i]=(ai==i)?1'b1:1'b0;  end  end  endmodule  module adder\_sub(s, cout, a, b, cin, cnt, clk, rst);  parameter n = 32;  input cin, clk, rst;  input cnt;  input [n-1:0] a, b;  output reg cout;  output reg [n-1:0] s;  reg [31:0] ai,bi;  reg ci;  always @ (posedge clk)  begin  ai <= a;  bi <= b;  ci <= cin;  if(~rst)  s <= 0;  cout <= 0;  if(cnt == 0)  {cout, s} <= ai + bi + ci;  else if(cnt == 1)  {cout, s} <= ai - bi - ci;  end  endmodule | module mux21\_tb;  reg sel, i1, i0;  wire y;    mux21 inst (  .sel(sel),  .y(y),  .i1(i1),  .i0(i0)  );    initial begin  sel=0; i1=0; i0=0;  #80 $finish;  end    always #10 i0 = ~i0;  always #20 i1 = ~i1;  always #40 sel = ~sel;    endmodule  module mux41\_tb;  reg s0, s1, i0, i1, i2, i3;  wire y;      mux41 inst (  .s0(s0),  .s1(s1),  .y(y),  .i0(i0),  .i1(i1),  .i2(i2),  .i3(i3)  );    initial begin  $dumpfile("test\_mux41.vcd");  $dumpvars(0, mux41\_tb);  s0 = 0; s1 = 0; i0 = 0; i1 = 0; i2 = 0; i3 = 0;  #320 $finish;  end    always #5 i0 = ~i0;  always #10 i1 = ~i1;  always #20 i2 = ~i2;  always #40 i3 = ~i3;  always #80 s0 = ~s0;  always #160 s1 = ~s1;    endmodule  module mux81\_tb;  reg s0, s1, s2, i0, i1, i2, i3, i4, i5, i6, i7;  wire y;      mux81 inst (  .s0(s0),  .s1(s1),  .s2(s2),  .y(y),  .i0(i0),  .i1(i1),  .i2(i2),  .i3(i3),  .i4(i4),  .i5(i5),  .i6(i6),  .i7(i7)  );    initial begin  $dumpfile("test\_mux81.vcd");  $dumpvars(0, mux81\_tb);  s0 = 0; s1 = 0; s2 = 0; i0 = 0; i1 = 0; i2 = 0; i3 = 0; i4 = 0; i5 = 0; i6 = 0; i7 = 0;  #2048 $finish;  end    always #1 i0 = ~i0;  always #2 i1 = ~i1;  always #4 i2 = ~i2;  always #8 i3 = ~i3;  always #16 i4= ~i4;  always #32 i5 = ~i5;  always #64 i6 = ~i6;  always #128 i7 = ~i7;  always #256 s0 = ~s0;  always #512 s1 = ~s1;  always #1024 s2 = ~s2;    endmodule  module com\_tb;  reg clk,rst;  reg [31:0]a,b;  wire p, q, r;  com ut (  .clk(clk),  .p(p),  .q(q),  .r(r),  .a(a),  .b(b),  .rst(rst)  );      initial  begin  clk=0;  a = 0;  b = 0;  rst=0;  #2 rst=1;  #4 rst=0;        #4048 $finish;  end    always #2 clk=~clk;  always #4 a=~a;  always #8 b=~b;  endmodule  module decode32\_tb;  reg[4:0]a;  reg clk,rst;  wire[31:0]out;  integer i;  decode32 ut (  .clk(clk),  .a(a),  .rst(rst),  .out(out)  );  initial begin  clk=0; a=0; rst=0;  #10 rst=1;  #2500 $finish;  end  initial  begin  for(a=0;a<32;a=a+1)  #20;  end  always #10 clk=~clk;  //always #20 a=a[i];  endmodule  module adder\_sub\_tb;  parameter n = 32;  reg cin, cnt, clk, rst;  reg [n-1:0] a, b;  wire cout;  wire [n-1:0] s;    adder\_sub inst(  .s(s),  .cout(cout),  .a(a),  .b(b),  .cin(cin),  .cnt(cnt)  );  initial begin  a = 0; b = 0; cin = 0; cnt = 0;  #5 rst=1;  end    always #5 a = ~a;  always #10 a = ~a;  always #20 b = ~b;  always #40 cin = ~cin;  always #80 cnt = ~cnt;  initial begin  #160 $finish;  end  endmodule |