

AND_GATE

```
library IEEE;  
use IEEE.std_logic_1164.all;
```

```
entity andgate is  
  port(  
    A:in std_logic;  
    B:in std_logic;  
    Y:out std_logic);  
end andgate;
```

```
architecture andlogic of andgate is  
begin  
  Y<=A AND B;  
end andlogic;
```

NAND_GATE

```
library IEEE;  
use IEEE.std_logic_1164.all;
```

```
entity nandgate is  
  port(  
    A:in std_logic;  
    B:in std_logic;  
    C:in std_logic;  
    Y:out std_logic);  
end nandgate;
```

```
architecture nandlogic of nandgate is  
begin  
  Y<=not(A and B and C);  
end nandlogic;
```

NOR_GATE

```
library IEEE;  
use IEEE.std_logic_1164.all;
```

```
entity nandgate is  
  port(  
    A:in std_logic;  
    B:in std_logic;  
    C:in std_logic;  
    Y:out std_logic);  
end nandgate;
```

```
architecture nandlogic of nandgate is
begin
    Y<=not(A and B and C);
end nandlogic;
```

NOT_GATE

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity notgate is
    port(
        A:in std_logic;
        Y:out std_logic);
end notgate;
```

```
architecture notlogic of notgate is
begin
    Y<=NOT(A);
end notlogic;
```

OR_GATE

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity orgate is
    port(
        A:in std_logic;
        B:in std_logic;
        Y:out std_logic);
end orgate;
```

```
architecture orlogic of orgate is
begin
    Y<=A or B;
end orlogic;
```

XNOR_GATE

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity xnorgate is
    port(
        A:in std_logic;
        B:in std_logic;
        C:in std_logic;
        Y:out std_logic);
```

```
end xnorgate;
```

```
architecture xnorlogic of xnorgate is  
begin
```

```
    Y<=not(A xor B xor C);  
end xnorlogic;
```

XOR_GATE

```
library IEEE;  
use IEEE.std_logic_1164.all;
```

```
entity xorgate is
```

```
    port(  
        A:in std_logic;  
        B:in std_logic;  
        C:in std_logic;  
        Y:out std_logic);  
end xorgate;
```

```
architecture xorlogic of xorgate is  
begin
```

```
    Y<=A xor B xor C;  
end xorlogic;
```

FULL_ADDDER

```
library IEEE;  
use IEEE.std_logic_1164.all;
```

```
entity full_adder is
```

```
    port(  
        a, b, c : in std_logic;  
        sum, carry : out std_logic);  
end full_adder;
```

```
architecture Behavioral of full_adder is  
begin
```

```
    sum <= a xor b xor c;  
    carry <= (a and b) or (b and c) or (c and a);  
end Behavioral;
```

FULL_SUBTRACTOR

```
library IEEE;  
use IEEE.std_logic_1164.all;
```

```
entity full_subtractor is
  port(
    a, b, cin : in std_logic;
    diff, borrow : out std_logic);
end full_subtractor;
```

```
architecture Behavioral of full_subtractor is
begin
  diff <= a xor b xor cin;
  borrow <= ((not a) and b) or ((not a) and cin) or (b and cin);
end Behavioral;
```

HALF_ADDER

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity half_adder is
  port(
    a, b : in std_logic;
    sum, carry : out std_logic);
end half_adder;
```

```
architecture Behavioral of half_adder is
begin
  sum <= a xor b;
  carry <= a and b;
end Behavioral;
```

HALF_SUBTRACTOR

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity half_subtractor is
  port(
    a, b : in std_logic;
    diff, borrow : out std_logic);
end half_subtractor;
```

```
architecture Behavioral of half_subtractor is
begin
  diff <= a xor b;
  borrow <= (not a) and b;
end Behavioral;
```

PRIORITY_ENCODER

```
library ieee;
use ieee.std_logic_1164.all;

entity priority_encoder_4x2 is
port(
d: in std_logic_vector(3 downto 0);
y: out std_logic_vector(1 downto 0);
v: out std_logic
);
end priority_encoder_4x2;

architecture behave of priority_encoder_4x2 is
begin
process(d)
begin
if(d(3)='1')then
    y<="11";
    v<='1';
elsif(d(2)='1')then
    y<="10";
    v<='1';
elsif(d(1)='1')then
    y<="01";
    v<='1';
elsif(d(0)='1')then
    y<="00";
    v<='1';
else
    y<="ZZ";
    v<='0';
end if;
end process;
end behave;
```

ENTITY_DECODER

```
library ieee;
use ieee.std_logic_1164.all;

entity decoder_2x4 is
```

```

port(
a: in std_logic_vector(1 downto 0);
b: out std_logic_vector(3 downto 0)
);
end decoder_2x4;

```

architecture behave of decoder_2x4 is

```

begin
process(a)
begin
if(a="00")then
    b<="0001";
elsif(a="01")then
    b<="0010";
elsif(a="10")then
    b<="0100";
elsif(a="11")then
    b<="1000";
end if;
end process;
end behave;

```

ENTITY_MUX

```

library ieee;
use ieee.std_logic_1164.all;

```

```

entity mux_2x1 is
port(
a,b,s: in std_logic;
z: out std_logic
);
end mux_2x1;

```

architecture behave of mux_2x1 is

```

begin
process(a,b,s)
begin
if(s='0')then
    z<=a;
elsif(s='1')then
    z<=b;
end if;

```

```
end process;  
end behave;
```

ENTITY_COMPARATOR

```
library ieee;  
use ieee.std_logic_1164.all;  
  
entity comparator_2bit is  
port(  
  a,b: in std_logic_vector(1 downto 0);  
  greater,equal,lower: out std_logic  
);  
end comparator_2bit;  
  
architecture behave of comparator_2bit is  
begin  
  process(a,b)  
  begin  
    if(a=b)then  
      equal<='1';  
      greater<='0';  
      lower<='0';  
    elsif(a>b)then  
      equal<='0';  
      greater<='1';  
      lower<='0';  
  
    elsif(a<b)then  
      equal<='0';  
      greater<='0';  
      lower<='1';  
    end if;  
  end process;  
end behave;
```

D_FLIP_FLOP

```
library ieee;  
use ieee.std_logic_1164.all;  
entity d_clk is  
port(  
  D,CLOCK: in std_logic;  
  Q,QBAR: out std_logic
```

```

);
end d_clk;
architecture bhv of d_clk is
begin
process(D,CLOCK)
variable tmp: std_logic;
begin
if(CLOCK= '1' and CLOCK'EVENT)then
if(D='0')then
tmp:= '0';
else
tmp:= '1';
end if;
end if;
Q <= tmp;
QBAR <= NOT tmp;
end process;
end bhv;

```

SR_FLIP_FLOP

```

library ieee;
use ieee.std_logic_1164.all;
entity sr_clk is
port
(
S,R,CLOCK: in std_logic;
Q,QBAR: out std_logic
);
end sr_clk;
architecture bhv of sr_clk is
begin
process(S,R,CLOCK)
variable tmp: std_logic;
begin
if(CLOCK= '1' and CLOCK'EVENT)then
if(S= '0' and R= '0') then
tmp:= tmp;
elsif(S= '0' and R= '1') then
tmp:= '0';
elsif(S= '1' and R= '0') then
tmp:= '1';
else
tmp:= '1';

```



```
end if;
end if;
Q <= tmp;
QBAR <= NOT tmp;
end process;
end bhv;
```

JK_FLIP_FLOP

```
library ieee;
use ieee.std_logic_1164.all;
entity jk_clk is
port
(
J,K,CLOCK: in std_logic;
Q,QBAR: out std_logic
);
end jk_clk;
architecture bhv of jk_clk is
begin
process(J,K,CLOCK)
variable tmp: std_logic;
begin
if(CLOCK= '1' and CLOCK'EVENT)then
if(J= '0' and K= '0') then
tmp:= tmp;
elsif(J= '0' and K= '1') then
tmp:= '0';
elsif(J= '1' and K= '0') then
tmp:= '1';
else
tmp:= not tmp;
end if;
end if;
Q <= tmp;
QBAR <= NOT tmp;
end process;
end bhv;
```

T_FLIP_FLOP

```
library ieee;
use ieee.std_logic_1164.all;
entity t_clk is
port
```

```

(
T,CLOCK: in std_logic;
Q,QBAR: out std_logic
);
end t_clk;
architecture bhv of t_clk is
begin
process(T,CLOCK)
variable tmp: std_logic;
begin
if(CLOCK= '1' and CLOCK'EVENT)then
tmp:= '1';
if(T='0')then
tmp:= tmp;
else
tmp:= not tmp;
end if;
end if;
Q <= tmp;
QBAR <= NOT tmp;
end process;
end bhv;

```

SISO

```

library ieee;
use ieee.std_logic_1164.all;
entity siso is
port
(
a, clk, clear : in std_logic;
q : out std_logic;
d : inout std_logic_vector(3 downto 0)
);
end siso;
architecture bhv of siso is
begin
process(clk, a)
begin
if(clear = '1') then
d <= "0000";
q <= '0';
end if;
if(clk'event and clk = '1') then

```

```

d(3 downto 1) <= d(2 downto 0);
d(0) <= a;
end if;
q <= d(3);
end process;
end bhv;

```

SIPO

```

library ieee;
use ieee.std_logic_1164.all;
entity sipo is
port
(
clk, clear : in std_logic;
Input_Data : in std_logic;
Q : inout std_logic_vector(2 downto 0)
);
end sipo;
architecture arch of sipo is
begin
process(clk)
begin
if(clear = '1') then
Q <= "000";
elsif(CLK'event and clk = '1') then
Q(2 downto 1) <= Q(1 downto 0);
Q(0) <= Input_Data;
end if;
end process;
end arch;

```

PIPO

```

library ieee;
use ieee.std_logic_1164.all;
entity pipo is
port
(
clk : in std_logic;
D : in std_logic_vector(2 downto 0);
Q : out std_logic_vector(2 downto 0)
);
end pipo;
architecture arch of pipo is

```

```

begin
process(clk)
begin
if(CLK'event and clk = '1') then
Q <= D;
end if;
end process;
end arch;

```

PISO

```

library ieee;
use ieee.std_logic_1164.all;
entity piso is
port
(
clk,shift,ld : inout std_logic;
b : in std_logic_vector(3 downto 0);
d : inout std_logic_vector(3 downto 0);
q : inout std_logic_vector(3 downto 0);
o : out std_logic
);
end piso;
architecture bhv of piso is
begin
ld <= not shift;
process(clk,shift)
begin
d(0) <= b(0);
if(clk'event and clk = '1') then
q(3 downto 0) <= d(3 downto 0);
for l in 3 downto 1 loop
d(1) <= (q(l - 1) and shift) or (ld and b(1));
end loop;
o <= q(3);
end if;
end process;
end bhv;

```

UP_COUNTER

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity up_counter is

```

```

port(
C,CLR : in std_logic;
Q : out std_logic_vector(3 downto 0)
);
end up_counter;
architecture bhv of up_counter is
signal tmp : std_logic_vector(3 downto 0);
begin
process(C,CLR)
begin
if(CLR='1')then
tmp <= "0000";
elsif(C'event and C='1')then
tmp <= tmp + 1;
end if;
end process;
Q <= tmp;
end bhv;

```

DOWN_COUNTER

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity down_counter is
port(
C,CLR : in std_logic;
Q : out std_logic_vector(3 downto 0)
);
end down_counter;
architecture bhv of down_counter is
signal tmp : std_logic_vector(3 downto 0);
begin
process(C,CLR)
begin
if(CLR='1')then
tmp <= "1111";
elsif(C'event and C='1')then
tmp <= tmp - 1;
end if;
end process;
Q <= tmp;
end bhv;

```

UP_DOWN_COUNTER

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity up_down_counter is
port(
C,CLR,mode : in std_logic;
Q : out std_logic_vector(3 downto 0)
);
end up_down_counter;
architecture bhv of up_down_counter is
signal tmp : std_logic_vector(3 downto 0);
begin
process(C,CLR,mode)
begin
if(mode='1')then
if(CLR='1')then
tmp <= "0000";
elsif(C'event and C='1')then
tmp <= tmp + 1;
end if;
else
if(CLR='1')then
tmp <= "1111";
elsif(C'event and C='1')then
tmp <= tmp - 1;
end if;
end if;
end process;
Q <= tmp;
end bhv;
```

MODE_COUNTER

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity mod_3_counter is
port(
C,CLR : in std_logic;
Q : out std_logic_vector(1 downto 0)
```

```

);
end mod_3_counter;
architecture bhv of mod_3_counter is
begin
process(C,CLR)
variable tmp : integer range 0 to 3;
begin
if(CLR='1')then
tmp := 0;
else
if(C'event and C='1')then
tmp := tmp + 1;
end if;
if(tmp=3)then
tmp := 0;
end if;
end if;
Q <= conv_std_logic_vector(tmp,2);
end process;
end bhv;

```

BOOTH_MULT

```

library ieee;

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity booth is
port(
m : in std_logic_vector(4 downto 1);
mc : in std_logic_vector(4 downto 1);
o : out std_logic_vector(8 downto 1)
);
end booth;

architecture arc of booth is
begin
process(m,mc)
variable ea:std_logic_vector(4 downto 1);
variable q:std_logic_vector(4 downto 1);
variable b:std_logic_vector(4 downto 1);
variable qn:std_logic_vector(2 downto 1);
begin

```

```

ea:="0000";
b:=m;
q:=mc;
qn:=q(1) & '0';
for i in 4 downto 1 loop
if qn="01" then
ea:=ea(4 downto 1)+b(4 downto 1);
q:=ea(1)&q(4 downto 2);
ea:=ea(4)&ea(4 downto 2);
qn:=q(1)&qn(2);
elsif qn="10" then
ea:=ea(4 downto 1)+not(b(4 downto 1))+1;
q:=ea(1)&q(4 downto 2);
ea:=ea(4)&ea(4 downto 2);
qn:=q(1)&qn(2);
else
q:=ea(1)&q(4 downto 2);
ea:=ea(4)&ea(4 downto 2);
qn:=q(1)&qn(2);
end if;
end loop;
o<=ea(4 downto 1)&q(4 downto 1);
end process;
end arc;

```

NON_RESTORING_DIVISION

```

library ieee;

```

```

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

```

```

entity nrd is
port(
q : in std_logic_vector(4 downto 1);
m : in std_logic_vector(4 downto 1);
r : out std_logic_vector(4 downto 1);
qo : out std_logic_vector(4 downto 1)
);
end nrd;

```

```

architecture arc of nrd is

```



```

begin
process(q,m)
variable a:std_logic_vector(4 downto 1);
variable qv:std_logic_vector(4 downto 1);
variable mv:std_logic_vector(4 downto 1);

begin
qv:=q;
mv:=m;
a:="0000";

for i in 4 downto 1 loop
if(a(4)='0')then
a(4 downto 2):=a(3 downto 1);
a(1):=qv(4);
qv(4 downto 2):=qv(3 downto 1);
a(4 downto 1):=a(4 downto 1)+(not mv(4 downto 1))+1;
elsif(a(4)='1')then
a(4 downto 2):=a(3 downto 1);
a(1):=qv(4);
qv(4 downto 2):=qv(3 downto 1);
a(4 downto 1):=a(4 downto 1)+mv(4 downto 1);
end if;
if(a(4)='0')then
qv(1):='1';
elsif(a(4)='1')then
qv(1):='0';
end if;
end loop;

if(a(4)='1')then
a(4 downto 1):= a(4 downto 1)+mv(4 downto 1);
end if;

qo<=qv(4 downto 1);
r<=a(4 downto 1);
end process;
end arc;

```

RESTORING_DIVISION

```
library ieee;
```

```
use ieee.std_logic_1164.all;
```

```

use ieee.std_logic_unsigned.all;

entity rd is
port(
q : in std_logic_vector(4 downto 1);
m : in std_logic_vector(4 downto 1);
r : out std_logic_vector(4 downto 1);
qo : out std_logic_vector(4 downto 1)
);
end rd;

architecture arc of rd is
begin
process(q,m)
variable a:std_logic_vector(4 downto 1);
variable qv:std_logic_vector(4 downto 1);
variable mv:std_logic_vector(4 downto 1);

begin
qv:=q;
mv:=m;
a:="0000";

for i in 4 downto 1 loop
a(4 downto 2):=a(3 downto 1);
a(1):=qv(4);
qv(4 downto 2):=qv(3 downto 1);
a(4 downto 1):=a(4 downto 1)+(not mv(4 downto 1))+1;

if(a(4)='0')then
qv(1):='1';
elsif(a(4)='1')then
qv(1):='0';
a(4 downto 1):=a(4 downto 1)+mv(4 downto 1);
end if;
end loop;

qo<=qv;
r<=a;
end process;
end arc;

```

ALU

```
library ieee;
```

```
use ieee.std_logic_1164.all;  
use ieee.std_logic_unsigned.all;  
use ieee.std_logic_arith.all;
```

```
entity ALU is  
port(  
  a,b: in std_logic_vector(1 downto 0);  
  sel: in std_logic_vector(1 downto 0);  
  res: out std_logic_vector(1 downto 0)  
);  
end ALU;
```

```
architecture bhv of ALU is  
begin  
  process(a,b,sel)  
  begin  
    case sel is  
      when "00" =>  
        res<= a+b;  
      when "01" =>  
        res<= a+(not b)+1;  
      when "10" =>  
        res<= a and b;  
      when "11" =>  
        res<= a or b;  
      when others =>  
        res<= "XX";  
    end case;  
  end process;  
end bhv;
```

ARITHMETIC_UNIT

```
library ieee;
```

```
use ieee.std_logic_1164.all;  
use ieee.std_logic_unsigned.all;  
use ieee.std_logic_arith.all;
```

```
entity arithmetic_unit is  
port(  
  a,b: in std_logic_vector(3 downto 0);
```

```

op: in std_logic_vector(2 downto 0);
f: out std_logic_vector(3 downto 0)
);
end arithmetic_unit;

architecture bhv of arithmetic_unit is
begin
process(op,a,b)
variable temp: std_logic_vector(3 downto 0);
begin
case op is
when "000" =>
temp:= a+b;
when "001" =>
temp:= a+b+1;
when "010" =>
temp:= a+(not b)+1;
when "011" =>
temp:= a+(not b);
when "100" =>
temp:= a+1;
when "101" =>
temp:= a-1;
when "110" =>
temp:= a;
when "111" =>
temp:= b;
when others =>
NULL;
end case;
f<=temp;
end process;
end bhv;

```

LOGICAL_UNIT

```

library ieee;

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity logical_unit is
port(

```

```

a,b: in std_logic_vector(3 downto 0);
op: in std_logic_vector(2 downto 0);
zero: out std_logic;
f: out std_logic_vector(3 downto 0)
);
end logical_unit;

```

```

architecture bhv of logical_unit is
begin
process(op,a,b)
variable temp: std_logic_vector(3 downto 0);
begin
case op is
when "000" =>
temp:= a OR b;
when "001" =>
temp:= a AND b;
when "010" =>
temp:= NOT a;
when "011" =>
temp:= a NOR b;
when "100" =>
temp:= a NAND b;
when "101" =>
temp:= a XOR b;
when "110" =>
temp:= a XNOR b;
when "111" =>
if a<b then
temp:= "1111";
else
temp:= "0000";
end if;
when others =>
NULL;
end case;
if temp="0000" then
zero<='1';
else
zero<='0';
end if;
f<=temp;
end process;
end bhv;

```

ADDER_SUBTRACTOR

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Adder_Subtractor is
  Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
        B : in  STD_LOGIC_VECTOR (3 downto 0);
        OP : in  STD_LOGIC;
        S : out STD_LOGIC_VECTOR (3 downto 0);
        C_OUT : out STD_LOGIC);
end Adder_Subtractor;

architecture Behavioral of Adder_Subtractor is
begin
  process (A, B, OP)
    variable temp : STD_LOGIC_VECTOR (4 downto 0);
  begin
    if OP = '0' then -- Addition operation
      temp := ("0" & A) + ("0" & B);
    else -- Subtraction operation
      temp := ("0" & A) - ("0" & B);
    end if;

    S <= temp(3 downto 0); -- Assign result to S
    C_OUT <= temp(4); -- Assign carry-out to C_OUT
  end process;
end Behavioral;
```

CARRY_LOOK_AHEAD

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity carry_lookahead_adder is
  Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
        B : in  STD_LOGIC_VECTOR (3 downto 0);
        Cin : in  STD_LOGIC;
        Sum : out STD_LOGIC_VECTOR (3 downto 0);
        Cout : out STD_LOGIC);
```

```
end carry_lookahead_adder;
```

architecture Behavioral of carry_lookahead_adder is

```
signal G, P, C : STD_LOGIC_VECTOR (3 downto 0);
```

```
begin
```

```
    G(0) <= A(0) AND B(0);
```

```
    P(0) <= A(0) XOR B(0);
```

```
    C(0) <= Cin;
```

```
    Sum(0) <= P(0) XOR C(0);
```

```
    G(1) <= A(1) AND B(1);
```

```
    P(1) <= A(1) XOR B(1);
```

```
    C(1) <= G(0) OR (P(0) AND Cin);
```

```
    Sum(1) <= P(1) XOR C(1);
```

```
    G(2) <= A(2) AND B(2);
```

```
    P(2) <= A(2) XOR B(2);
```

```
    C(2) <= G(1) OR (P(1) AND (G(0) OR (P(0) AND Cin)));
```

```
    Sum(2) <= P(2) XOR C(2);
```

```
    G(3) <= A(3) AND B(3);
```

```
    P(3) <= A(3) XOR B(3);
```

```
    C(3) <= G(2) OR (P(2) AND (G(1) OR (P(1) AND (G(0) OR (P(0) AND Cin)))));
```

```
    Sum(3) <= P(3) XOR C(3);
```

```
    Cout <= C(3);
```

```
end Behavioral;
```

RIPPLE_CARRY_FACTOR

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity ripple_carry_adder_4bit is
```

```
    Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
```

```
          B : in  STD_LOGIC_VECTOR (3 downto 0);
```

```
          Cin : in  STD_LOGIC;
```

```
          Sum : out STD_LOGIC_VECTOR (3 downto 0);
```

```
          Cout : out STD_LOGIC);
```

```
end ripple_carry_adder_4bit;
```

architecture Behavioral of ripple_carry_adder_4bit is

```
    signal carry : std_logic_vector(3 downto 0);
begin
    Sum(0) <= A(0) xor B(0) xor Cin;
    carry(0) <= (A(0) and B(0)) or (A(0) and Cin) or (B(0) and Cin);
    Sum(1) <= A(1) xor B(1) xor carry(0);
    carry(1) <= (A(1) and B(1)) or (A(1) and carry(0)) or (B(1) and carry(0));
    Sum(2) <= A(2) xor B(2) xor carry(1);
    carry(2) <= (A(2) and B(2)) or (A(2) and carry(1)) or (B(2) and carry(1));
    Sum(3) <= A(3) xor B(3) xor carry(2);
    Cout <= (A(3) and B(3)) or (A(3) and carry(2)) or (B(3) and carry(2));
end Behavioral;
```

SINGLE_PORT_RAM

```
library ieee;
```

```
use ieee.std_logic_1164.all;
```

```
entity single_port_ram is
```

```
    port(
        data:in std_logic_vector(7 downto 0);
        addr:in natural range 0 to 63;
        we:in std_logic:='1';
        clk:in std_logic;
        q:out std_logic_vector(7 downto 0)
    );
end entity;
```

```
architecture rtl of single_port_ram is
```

```
    subtype word_t is std_logic_vector(7 downto 0);
    type memory_t is array(63 downto 0) of word_t;
    signal ram:memory_t;
    signal addr_reg:natural range 0 to 63;
begin
    process(clk)
    begin
        if(rising_edge(clk)) then
            if(we = '1') then
                ram(addr)<=data;
            end if;
            addr_reg<=addr;
        end if;
    end process;
```



```
end process;  
q<=ram(addr_reg);  
end rtl;
```