Active-HDL PDF Export
MohamedTaha192000280 workspace

Contents

1 Table of Contents
2 processor 1
2.1 FA.vhd 1
2.2 NBitAdder.vhd 1
2.3 PartA.vhd 2
2.4 PartB.vhd 3
2.5 PartC.vhd 3
2.6 PartD.vhd 4
2.7 ALU.vhd 4
2.8 MainMemory.vhd 6
2.9 Register.vhd 7
2.10 PC.vhd 7
2.11 ControlUnit .vhd 8
2.12 ALU_Selc.vhd 8
2.13 Processor.vhd 9
2.14 untitled.awc 12

2 processor

2.1 FA.vhd

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

entity FA is
     port(
          A : in STD_LOGIC;
          B : in STD_LOGIC;
          CIN : in STD_LOGIC;
          S : out STD_LOGIC;
          COUT : out STD_LOGIC
          );
end FA;

--}} End of automatically maintained section

architecture FA of FA is
begin

    -- enter your statements here --
    S <= A Xor B Xor CIN;
    COUT <= ((A xor B) and CIN) or (A and B);

end FA;
```

2.2 NBitAdder.vhd

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

entity \add_sub\ is
    generic(
          n : Integer :=16
    );
     port(
          CIN : in STD_LOGIC;
          A : in STD_LOGIC_VECTOR(n-1 downto 0);
          B : in STD_LOGIC_VECTOR(n-1 downto 0);
          S : out STD_LOGIC_VECTOR(n-1 downto 0);
          COUT : out std_logic
          );
end \add_sub\;

architecture \Add_sub_model\ of \add_sub\ is
component FA is
     port(
          A : in STD_LOGIC;
          B : in STD_LOGIC;
          CIN : in STD_LOGIC;
          S : out STD_LOGIC;
          COUT : out STD_LOGIC
          );
end component;
signal temp: std_logic_vector(n downto 0);
begin
temp(0) <= CIN;
Loop1 : for i in 0 to n-1 generate
    adder: FA port map (A(i),B(i),temp(i),S(i),temp(i+1));
```

```
end generate;
COUT <= temp(n);

end \Add_sub_model\;
```

2.3 PartA.vhd

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

entity PartA is
    generic(
     n:Integer :=16
    );
     port(
         CIN : in STD_LOGIC;
         A : in STD_LOGIC_VECTOR(n-1 downto 0);
         B : in STD_LOGIC_VECTOR(n-1 downto 0);
         S : in STD_LOGIC_VECTOR(1 downto 0);
         F : out STD_LOGIC_VECTOR(n-1 downto 0);
          COUT : out std_logic
         );
end PartA;

--}} End of automatically maintained section

architecture \PartA_model\ of PartA is
component \add_sub\ is
    generic(
          n : Integer :=16
    );
     port(
         CIN : in STD_LOGIC;
         A : in STD_LOGIC_VECTOR(n-1 downto 0);
         B : in STD_LOGIC_VECTOR(n-1 downto 0);
         S : out STD_LOGIC_VECTOR(n-1 downto 0);
         COUT : out STD_LOGIC
         );
end component;
signal t,c,e,o,h,j:std_logic_vector(n-1 downto 0);
signal p,d,g,i,k:STD_LOGIC;
begin
      e <= not B;
    add1 : \add_sub\ port map(CIN,A,x"0000",t,p);
    add2 : \add_sub\ port map(CIN,A,B,c,d);
    add3 : \add_sub\ port map('1',A,e,o,g);
    add4 : \add_sub\ port map(CIN,o,x"1111",h,i);
    add5 : \add_sub\ port map('0',A,x"1111",j,k);
F <= t when S = "00" else c when S = "01" else h when S ="10" and CIN = '0' e
lse o when S ="10" and CIN = '1' else
j when S = "11" and CIN = '0' else x"0000" when S = "11" and CIN = '1' ;

COUT <= p when S = "00" else d when S = "01" else i when S ="10" and CIN = '0
' else g when S ="10" and CIN = '1' else
k when S = "11" and CIN = '0' else '0' when S = "11" and CIN = '1' ;

end \PartA_model\ ;
```

2.4 PartB.vhd

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

entity PartB is
    generic(
     n:Integer :=16
    );
     port(

          A : in STD_LOGIC_VECTOR(n-1 downto 0);
          B : in STD_LOGIC_VECTOR(n-1 downto 0);
          S : in STD_LOGIC_VECTOR(1 downto 0);
          F : out STD_LOGIC_VECTOR(n-1 downto 0)
          );
end PartB;

--}} End of automatically maintained section

architecture PartB_model of PartB is
begin

    -- enter your statements here --
F <= (A and B) when S = "00" else
           (A or B) when S = "01" else
           (A xor B) when S = "10" else
           (not A) when S = "11" ;


end PartB_model;
```

2.5 PartC.vhd

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

entity PartC is
    generic(
     n:Integer :=16
    );
     port(
          A : in STD_LOGIC_VECTOR(n-1 downto 0);
          CIN :in STd_logic;
          S : in STD_LOGIC_VECTOR(1 downto 0);
          F : out STD_LOGIC_VECTOR(n-1 downto 0)
          );
end PartC;

--}} End of automatically maintained section

architecture PartC of PartC is
begin

F <= '0' & A(n-1 downto 1) when S ="00" else
    A(0) & A(n-1 downto 1) when S ="01" else
     CIN & A(n-1 downto 1) when S ="10" else
   A(n-1) & A(n-1 downto 1) when S="11";

end PartC;
```

2.6 PartD.vhd

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

entity PartD is
    generic(
     n:Integer :=16
    );
     port(
         CIN : in STD_LOGIC;
         A : in STD_LOGIC_VECTOR(n-1 downto 0);
         S : in STD_LOGIC_VECTOR(1 downto 0);
         F : out STD_LOGIC_VECTOR(n-1 downto 0)
         );
end PartD;

--}} End of automatically maintained section

architecture PartD of PartD is
begin

     F <= A(14 downto 0) & '0' when S ="00" else
          A(14 downto 0) & A(n-1) when S ="01" else
          A(14 downto 0) & CIN when S ="10" else
          x"0000" when S ="11" ;


end PartD;
```

2.7 ALU.vhd

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

entity ALU is
    generic(
     n:Integer :=16
    );
     port(
         CIN : in STD_LOGIC;
         COUT : out STD_LOGIC;
         A : in STD_LOGIC_VECTOR(n-1 downto 0);
         B : in STD_LOGIC_VECTOR(n-1 downto 0);
         S : in STD_LOGIC_VECTOR(3 downto 0);
         F : out STD_LOGIC_VECTOR(n-1 downto 0);
         Negativeflag : out std_logic;
         Zeroflag : out std_logic;
         Carryflag : out std_logic
         );
end ALU;

--}} End of automatically maintained section

architecture ALU of ALU is
component PartA is
    generic(
     n:Integer :=16
    );
     port(
         CIN : in STD_LOGIC;
```

```
         A : in STD_LOGIC_VECTOR(n-1 downto 0);
         B : in STD_LOGIC_VECTOR(n-1 downto 0);
         S : in STD_LOGIC_VECTOR(1 downto 0);
         F : out STD_LOGIC_VECTOR(n-1 downto 0);
          COUT : out std_logic
         );
end component;
component PartB is
    generic(
     n:Integer :=16
    );
     port(

         A : in STD_LOGIC_VECTOR(n-1 downto 0);
         B : in STD_LOGIC_VECTOR(n-1 downto 0);
         S : in STD_LOGIC_VECTOR(1 downto 0);
         F : out STD_LOGIC_VECTOR(n-1 downto 0)
         );
end component;
component PartC is
    generic(
     n:Integer :=16
    );
     port(
         A : in STD_LOGIC_VECTOR(n-1 downto 0);
         CIN :in STd_logic;
         S : in STD_LOGIC_VECTOR(1 downto 0);
         F : out STD_LOGIC_VECTOR(n-1 downto 0)
         );
end component;
component PartD is
    generic(
     n:Integer :=16
    );
     port(
         CIN : in STD_LOGIC;
         A : in STD_LOGIC_VECTOR(n-1 downto 0);
         S : in STD_LOGIC_VECTOR(1 downto 0);
         F : out STD_LOGIC_VECTOR(n-1 downto 0)
         );
end component;
signal s1 ,s2 : std_logic_vector(1 downto 0);
signal o,l,c,d,e : std_logic_vector(n-1 downto 0);
signal g,h : std_logic;
begin
s1 <= S(1 downto 0);
s2 <= S(3 downto 2);

ParA : PartA port map (CIN,A,B,s1,o,g);
ParB : PartB port map (A,B,s1,l);
ParC : PartC port map (A,CIN,s1,c);
ParD : PartD port map (CIN,A,s1,d);
e <= o when s2 ="00" else l when s2 ="01" else c when s2 ="10" else d when s2
 ="11";
h <= g when s2 ="00" else '0';
Negativeflag <= '1' when e(n-1) ='1' else '0';
Zeroflag <= '1' when e = x"0000" else '0';
Carryflag <= '1' when h ='1' else '0';
F <= e ;
COUT <= h;
end ALU;
```

2.8 MainMemory.vhd

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

entity \Main Memory\ is
    generic(
          n : Integer :=16
    );
     port(
          clk : in STD_LOGIC;
          rst : in STD_LOGIC;
          MR : in STD_LOGIC;
          MW : in STD_LOGIC;
          address : in STD_LOGIC_VECTOR(n-1 downto 0);
          WD : in STD_LOGIC_VECTOR(n-1 downto 0);

          RD : out STD_LOGIC_VECTOR(n-1 downto 0)
          );
end \Main Memory\;

--}} End of automatically maintained section

architecture \Main_Memory_model\ of \Main Memory\ is
type RAM_TYPE is array(0to((2**(n-1)-1)))of std_logic_vector(n-1 downto 0);
type ROM_TYPE is array(0 to 15)of std_logic_vector(n-1 downto 0);
signal ram : RAM_TYPE;
signal rom : ROM_TYPE;
signal EN:std_logic;
begin
    EN <='1' when MR = '1' or MW = '1'else '0';
process (clk,EN,rst)
begin
if (rst='1')then
     -- memory locations (contents) --
ram(3630)<=x"0002";  ---E2E---
ram(3631)<=x"0001";  ---E2F---
ram(3632)<=x"0008";  ---E30---
ram(3633)<=x"0007";  ---E31---
ram(3634)<=x"0005";  ---E32---
ram(3635)<=x"0003";  ---E33---
ram(3636)<=x"0F0F";  ---E34---
ram(3637)<=x"000A";  ---E35---
ram(3638)<=x"0000";  ---E36---
     -- Instructions --                                  ---test case: ((2+10-
8)+1)-1 = 4
rom(0)<=    x"0E2E" ;  ---load---                        ---load 2
rom(1)<=    x"2E35" ;  ---add---                         ---add 10
rom(2)<=    x"1E30" ;  ---sub---                         ---sub 8
rom(3)<=    x"0E32" ;  ---store---                       ---store 4 in E32
rom(4)<=    x"DE32" ;  ---inc---                         ---Inc
rom(5)<=    x"1E2F" ;  ---store---                       ---store 5 in E2F
rom(6)<=    x"0E2F" ;  ---dec---                         ---Dec
rom(7)<=    x"EE33" ;  ---store---                       ---store 4 in E33
-------------------------------------
elsif (rising_edge(clk)) and MR = '0' and MW = '0' Then
    RD <= rom(to_integer(unsigned(address)));
elsif(rising_edge(clk))and MW ='1' then
    ram(to_integer(unsigned(address))) <= WD;
elsif(rising_edge(clk)) and MR = '1' Then
    RD <= ram(to_integer(unsigned(address(11 downto 0))));
elsif (rising_edge(Clk)) and EN='1' and address /= "XXXXXXXXXXXXXXX" and add
```

```
ress /= "ZZZZZZZZZZZZZZZZ"then
        RD <= rom(to_integer(unsigned(Address)));

end if;
end process;
end \Main_Memory_model\ ;
```

2.9 Register.vhd

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

entity \Register\ is
    generic(n:integer :=16);
     port(
         CLK : in STD_LOGIC;
         Rst : in STD_LOGIC;
         Datain : in STD_LOGIC_VECTOR(n-1 downto 0);
         Dataout : out STD_LOGIC_VECTOR(n-1 downto 0)
         );
end \Register\;

--}} End of automatically maintained section

architecture \Register\ of \Register\ is
begin

process(CLK,Rst)
begin
if Rst='1' then Dataout<=(others=>'Z');
elsif Rst='0' and rising_edge(CLK) then Dataout <= Datain;
end if;
end process;
end \Register\;
```

2.10 PC.vhd

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

entity PC is
    generic(n:integer:=16);
     port(
         Rst : in STD_LOGIC;
         CLK : in STD_LOGIC;
         PCin : in STD_LOGIC_VECTOR(n-1 downto 0);
         PCout : out STD_LOGIC_VECTOR(n-1 downto 0)
         );
end PC;

--}} End of automatically maintained section

architecture PC of PC is
signal s1 : std_logic_vector(n-1 downto 0);
begin
process(Rst,CLK)
begin
if Rst='1' then s1 <= x"FFFF";
```

```
elsif Rst='0' and rising_edge(CLK) then
    s1<=PCin;
    PCout<=s1+1;
end if;
end process;
end PC;
```

2.11 ControlUnit .vhd

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

entity Control_U is
    generic(n:integer:=16);
     port(
         MR,MW,ALU_EN : out STD_LOGIC;
         OPcode : in STD_LOGIC_VECTOR(3 downto 0)
         );
end Control_U;

--}} End of automatically maintained section

architecture Control_U of Control_U is
signal w , r :std_logic;
begin

r <='1' when OPcode="0000" else '0';       ---load---
w <='1' when OPcode="0001" else '0';       ---store---
ALU_EN <='1' when (r ='0' and w ='0') else '0'; ---ALU operations---
MR <= r;
MW <= w;
end Control_U;
```

2.12 ALU_Selc.vhd

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

entity ALU_Selc is
    generic(n:integer:=16);
    port(
         EN : in STd_logic;
         OPcode : in STD_LOGIC_VECTOR(3 downto 0);
         S : out STD_LOGIC_VECTOR(3 downto 0);
         Carry : out STD_LOGIC
         );
end ALU_Selc;

--}} End of automatically maintained section

architecture ALU_Selc of ALU_Selc is
begin

S <= "0001" when OPcode="0010" and EN='1'       -----CIN=0-----             -
----ADD-----
else "0010" when OPcode="0011" and EN='1'       -----CIN=1-----             -
----SUB-----
else "0100" when OPcode="0100" and EN='1'                                   -
```

```
----AND-----
else "0101" when OPcode="0101" and EN='1'                                    -
----OR------
else "0110" when OPcode="0110" and EN='1'                                    -
----XOR-----
else "0111" when OPcode="0111" and EN='1'                                    -
----NOT-----
else "1100" when OPcode="1000" and EN='1'                                    -
----SHL-----
else "1000" when OPcode="1001" and EN='1'                                    -
----SHR-----
else "1101" when OPcode="1010" and EN='1'                                    -
----ROL-----
else "1001" when OPcode="1011" and EN='1'                                    -
----ROR-----
else "0000" when OPcode="1101" and EN='1'       -----CIN=1-----             -
----Inc-----
else "0011" when OPcode="1110" and EN='1'       -----CIN=0-----             -
----Dec-----
else "1111" when OPcode="1111" and EN='1';                                   -
----Rst-----


Carry<='1' when OPcode="0011" and OPcode="1101" and EN='1' else '0';

end ALU_Selc;
```

2.13 Processor.vhd

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

entity Processor is
    generic( n:integer:=16);
     port(
          CLK : in STD_LOGIC;
          Rst : in STD_LOGIC
          );
end Processor;

--}} End of automatically maintained section

architecture Processor of Processor is
component ALU is
    generic(
     n:Integer :=16
    );
     port(
          CIN : in STD_LOGIC;
          COUT : out STD_LOGIC;
          A : in STD_LOGIC_VECTOR(n-1 downto 0);
          B : in STD_LOGIC_VECTOR(n-1 downto 0);
          S : in STD_LOGIC_VECTOR(3 downto 0);
          F : out STD_LOGIC_VECTOR(n-1 downto 0);
          Negativeflag : out std_logic;
          Zeroflag : out std_logic;
          Carryflag : out std_logic
          );
end component;
component \Main Memory\ is
    generic(
```

```
          n : Integer :=16
    );
     port(
          clk : in STD_LOGIC;
          rst : in STD_LOGIC;
          MR : in STD_LOGIC;
          MW : in STD_LOGIC;
          address : in STD_LOGIC_VECTOR(n-1 downto 0);
          WD : in STD_LOGIC_VECTOR(n-1 downto 0);

          RD : out STD_LOGIC_VECTOR(n-1 downto 0)
          );
end component;
component \Register\ is
    generic(n:integer :=16);
     port(
          CLK : in STD_LOGIC;
          Rst : in STD_LOGIC;
          Datain : in STD_LOGIC_VECTOR(n-1 downto 0);
          Dataout : out STD_LOGIC_VECTOR(n-1 downto 0)
          );
end component;
component PC is
    generic(n:integer:=16);
     port(
          Rst : in STD_LOGIC;
          CLK : in STD_LOGIC;
          PCin : in STD_LOGIC_VECTOR(n-1 downto 0);
          PCout : out STD_LOGIC_VECTOR(n-1 downto 0)
          );
end component;
component Control_U is
    generic(n:integer:=16);
     port(
          MR,MW,ALU_EN : out STD_LOGIC;
          OPcode : in STD_LOGIC_VECTOR(3 downto 0)
          );
end component;
component ALU_Selc is
    generic(n:integer:=16);
    port(
          EN : in STd_logic;
          OPcode : in STD_LOGIC_VECTOR(3 downto 0);
          S : out STD_LOGIC_VECTOR(3 downto 0);
          Carry : out STD_LOGIC
          );
end component;
signal mr,mw,aluen,Cin,Cout,Negativeflag,Zeroflag,Carryflag : std_logic;
signal pcout,pcin,addr,wd,rd,F,AC,B: std_logic_vector(n-1 downto 0);
signal S,opcode :std_logic_vector(3 downto 0);
signal data_in,data_out: std_logic_vector(n-1 downto 0);
signal s_addr:std_logic_vector(11 downto 0);
begin
    p_c : PC port map(Rst,CLK,pcin,pcout);
    buf : \Register\ port map(CLK,Rst,data_in,data_out);
    addr<=("0000"&s_addr) when mr='1' or mw='1' else pcout;
    m_m : \Main Memory\ port map(CLK,Rst,mr,mw,addr,wd,rd);
    C_U : Control_U port map(mr,mw,aluen,rd(15 downto 12));
    sel : ALU_Selc port map(aluen,rd(15 downto 12),S,Cin);
    a_l_u: ALU port map(Cin,Cout,AC,rd,S,F,Negativeflag,Zeroflag,Carryflag);
    pcin<=pcout;
    S_addr<=RD(11 downto 0);
    AC<= F when aluen = '1' else rd;
```

```
end Processor;
```

2.14 untitled.awc