Elektronski fakultet u Nišu
Katedra za Računarstvo
Predmet : Ugrađeni računarski sistemi

Aritmetičko Logička Jedinica
ALU i Množač (8x6 Ripple-Carry)

Studenti: Milena Dukić 10204
Iva Ćirković 10180

U Nišu,
21.03.2005. godine

1. Aritmetičko Logička jedinica

Aritmetičko-logička jedinica (Arithmetic Logic Unit - ALU) predstavlja deo centralnog procesora koja služi za obavljanje aritmetičkih i logičkih operacija nad celobrojnim ulaznim podacima. Pored ulaza za podatke, ALU sadrži i upravljačke (selektorske) ulaze pomoću kojih se bira određena operacija, iz skupa operacija ALU-a, koja će se obaviti. Operacija koja se obavlja na ulazu ALU-a predstavlja se u kodiranom obliku, tj. signal na selektorskim linijama ustvari predstavlja kod operacije koja ce se izvršiti nad ulaznim podacima. Selektorske linije se unutar ALU-a dekodiraju, tako da se pomoću *n* selektorskih linija može predstaviti maksimalno 2^n različitih operacija.

1.1. ALU - konkretno rešenje

Aritmetičko-logička jedinica, opisana u daljem tekstu, u stanju je da obavlja 14 različitih operacija, koje su definisane u Tabeli 1. Operacije se obavljaju nad osmobitnim ulaznim podacima, *A* i *B*. *Sel* je petobitni selektorski ulaz (*S4:S0*), a *Cin* je ulazni prenos (*Carry*). *Y* je osmobitni izlazni signal, a *CarryOut* je izlazni prenos.

S4 S3 S2 S1 S0 Cin	Operacija	Carry Out	Funkcija	Implementacioni blok
0 0 0 0 0 0	Y <= A	0	Prenosi A	Aritmetička Jedinica
0 0 0 0 0 1	Y <= A + 1	{0},{1}	Inkrementuje A	Aritmetička Jedinica
0 0 0 0 1 0	Y <= A + B	{0},{1}	Sabira A i B	Aritmetička Jedinica
0 0 0 0 1 1	Y <= A + B + 1	{0},{1}	Sabira sa Carry-em	Aritmetička Jedinica
0 0 0 1 0 0	Y <= A + /B	{0},{1}	A + komplement B	Aritmetička Jedinica
0 0 0 1 0 1	Y <= A - B	{0},{1}	Oduzimanje	Aritmetička Jedinica
0 0 0 1 1 0	Y <= A - 1	{0},{1}	Dekrementuje A	Aritmetička Jedinica
0 0 0 1 1 1	Y <= A	0	Prenosi A	Aritmetička Jedinica
0 0 1 0 0 0	Y <= A and B	0	AND	Logička Jedinica
0 0 1 0 1 0	Y <= A or B	0	OR	Logička Jedinica
0 0 1 1 0 0	Y <= A xor B	0	XOR	Logička Jedinica
0 0 1 1 1 0	Y <= /A	0	Komplement A	Logička Jedinica
0 0 0 0 0 0	Y <= A	0	Prenosi A	Pomerača Jedinica
0 1 0 0 0 0	Y <= shl A	0	Pomera ulevo A	Pomerača Jedinica
1 0 0 0 0 0	Y <= shr A	0	Pomera udesno A	Pomerača Jedinica
1 1 0 0 0 0	Y <= 0	0	Prenosi 0	Pomerača Jedinica

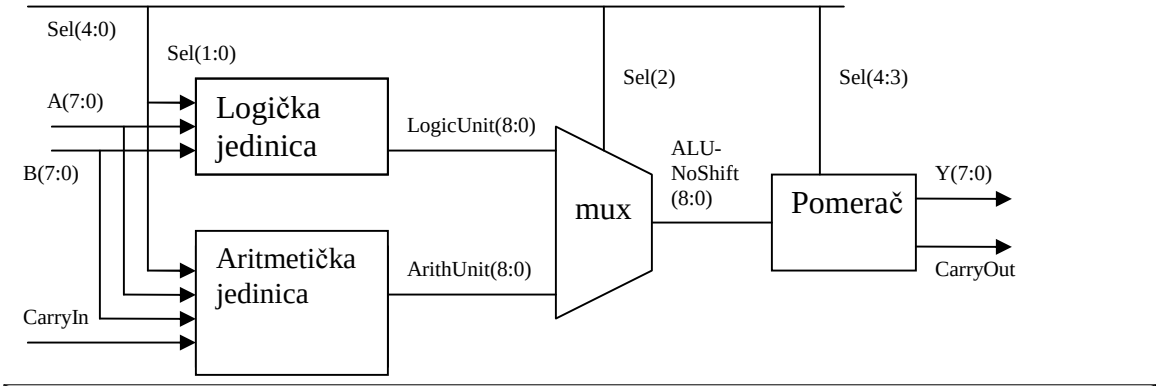
Tabela 1.

U konkretnom slučaju za modeliranje aritmetičko logičke jedinice korišćena je arhitektura specificirana na *behavioral* nivou, koja opisuje samo ponašanje komponente, zanemarujući njenu unutrašnju strukturu. Pored ovakvog načina modeliranja moguće je koristiti i strukturalnu ili *dataflow* konstrukciju (opis).

Kompletna funkcionalna tabela ALU-a mogla bi se modelovati korišćenjem samo jedne *case* naredbe, ali bi ovako dobijena struktura bila neefikasna. Umesto toga, ALU je modelovana korišćenjem odvojene logičke jedinice, aritmetičke jedinice i pomerača. Kod ovog rešenja je izvršena dekompozicija problema, tj aritmetičko-logički-pomerački proces je razbijen na tri procesa. Odvajanjem logičke i aritmetičke jedinice na ovaj način, i multipleksiranjem njihovih izlaza na ulazu pomerača, postižu se bolje performanse (kraće vreme propagacije signala kroz logiku ALU-a).

Razdvajanje operacija na aritmetičke, logičke i operacije pomeranja određeno je kodovima operacija.

Svaka jedinica modelovana je korišćenjem jedne *case* naredbe u kojoj se, na osnovu vrednosti selektorskih ulaza (i ulaznog prenosa u slučaju aritmetičke jedinice) određuje operacija koju će jedinica obaviti (videti Tabelu 1.). Izlazi iz aritmetičke i logičke jedinice vode se na ulaz multipleksera kojim se, na osnovu vrednosti odgovarajuće selektorske linije (u ovom slučaju S2), bira vrednost koja će se propustiti do ulaza pomerača. Aritmetička jedinica je modelirana korišćenjem jedne *case* naredbe. Ovakav način modeliranja realizuju izraze kao što je A+B+1 korišćenjem jednog sabirača sa ulaznim prenosom postavljenim na 1.



Slika 1. Strukturalni model Aritmetičko logičke jedinice

Izbor jedne od 4 mogućih logičkih operacija (koje obavlja logička jedinica) vrši se selektorskim ulazima *S0* i *S1*, a izbor aritmetičkih operacija ulazima *S0*, *S1* i ulazom *Cin*. Multipleksiranje izlaza iz aritmetičke i logičke jedinice na ulazu pomerača upravljano je selektorskim ulazom *S2* (*S2=0* označava aritmetičku, a *S2=1* logičku operaciju), a izbor operacije pomerača vrši se na osnovu ulaza *S3* i *S4*.

1.2. VHDL kôd ALU-a

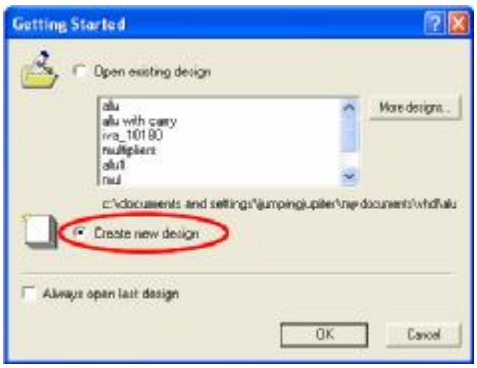
```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

entity ALU is
        port            (Sel:           in unsigned(4 downto 0);
                        CarryIn:        in std_logic;
                        A,B:            in unsigned(7 downto 0);
                        Y:              out unsigned(7 downto 0);
                        CarryOut:     out std_logic);
end ALU;



architecture alu of ALU is
begin
 ALU_AND_SHIFT:
 process (Sel,A,B,CarryIn)
   variable Sel0_1_CarryIn : unsigned (2 downto 0 );
        variable AA,BB : unsigned (8 downto 0);
        variable LogicUnit, ArithUnit,
                        ALU_NoShift : unsigned(8 downto 0);
        variable Pomoc : unsigned (8 downto 0);
        begin
                --------------------
                -- Logicka jedinica
                --------------------
                AA :='0' & A;
                BB :='0' & B;
                LOGIC_UNIT: case Sel(1 downto 0) is
                        when "00"  => LogicUnit := AA and BB;
                        when "01"  => LogicUnit := AA or BB;
                        when "10"  => LogicUnit := AA xor BB;
                        when "11"  => LogicUnit := not AA;
                        when others => LogicUnit:=(others => 'X');
                end case LOGIC_UNIT;
                ----------------------
                -- Aritmeticka jedinica
                ----------------------
                Sel0_1_CarryIn := Sel(1 downto 0) & CarryIn;
                ARITH_UNIT: case Sel0_1_CarryIn is
                        when "000"   => ArithUnit := AA;
                        when "001"   => ArithUnit := AA + 1;
                        when "010"   => ArithUnit := AA + BB;
```

```
            when "011"   => ArithUnit := AA + BB + 1;
            when "100"   => ArithUnit := AA + not BB;
            when "101"   => ArithUnit := AA - BB;
            when "110"   => ArithUnit := AA - 1;
            when "111"   => ArithUnit := AA;
            when others  => ArithUnit := (others => 'X');
        end case ARITH_UNIT;
        ---------------------------------------------------
        -- Multiplex izmedju Logicke & Aritmeticke jedinice
        ---------------------------------------------------
        LA_MUX : if (Sel(2) = '1') then
            ALU_NoShift := LogicUnit;
        else
            ALU_NoShift := ArithUnit;
        end if LA_MUX;


        ------------------------
        -- Shift operatori
        ------------------------
        CarryOut <= ALU_NoShift (8);
        SHIFT : case Sel(4 downto 3) is
            when "00"    => Pomoc := ALU_NoShift;
            when "01"    => Pomoc :=  Shift_left(ALU_NoShift, 1);
            when "10"    => Pomoc := Shift_right(ALU_NoShift, 1);
            when "11"    => Pomoc := (others => '0');
            when others  => Pomoc := (others => 'X');
        end case SHIFT;
        Y <= Pomoc (7 downto 0);
  end process ALU_AND_SHIFT;
end architecture alu;
```

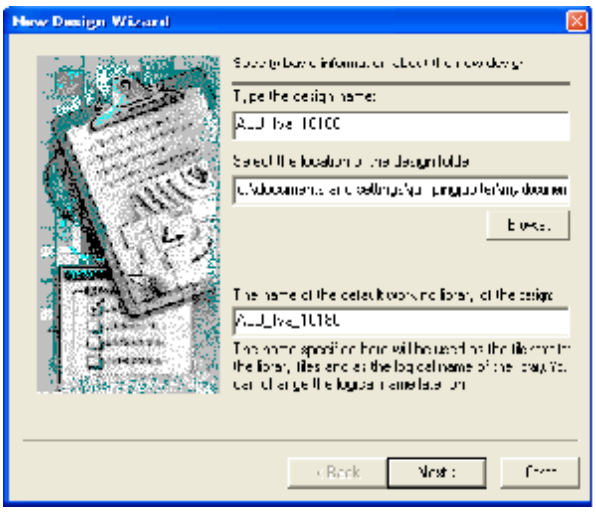
1.3. Vežba 1 - Postupak testiranja ALU-a

1. **Pokrenuti program *Active-HDL*.** 

2. **U prozoru *Getting Started* selektovati opciju *Create new design* i kliknuti na *OK*.**

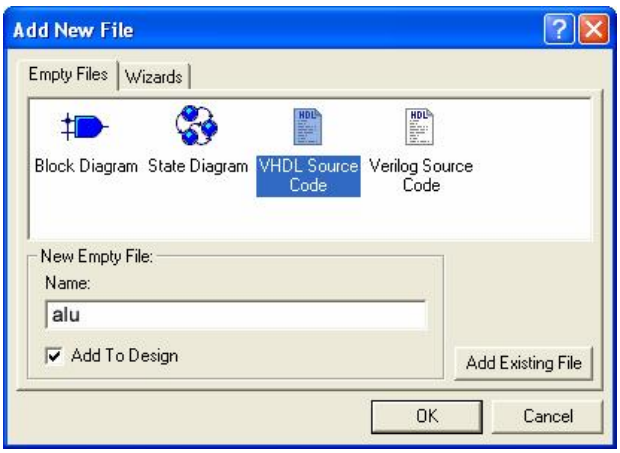


3. **U novootvorenom prozoru za naziv dizajna upisati *ALU_ime_broj_indexa* (na.pr. ALU_Iva_10180) i klinuti na *Next*.**



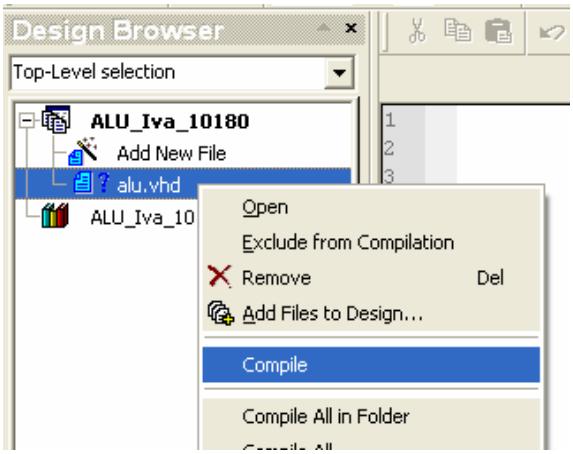
4. **Selektovati *Create an empty design* i kilknuti na *Next* a zatim *Finish*.**

5. **U prozoru *Design browser* dva puta kliknuti na *Add new file*. Za tip fajla izabrati *VHDL Source Code* i imenovati ga sa *ALU*. Selektovati opciju *Add To Design* i kliknuti na *OK*.**



6. **U prozoru editora prekucati dati kod, i obratiti pažnju da li su svi ulazni i izlazni portovi definisani.**

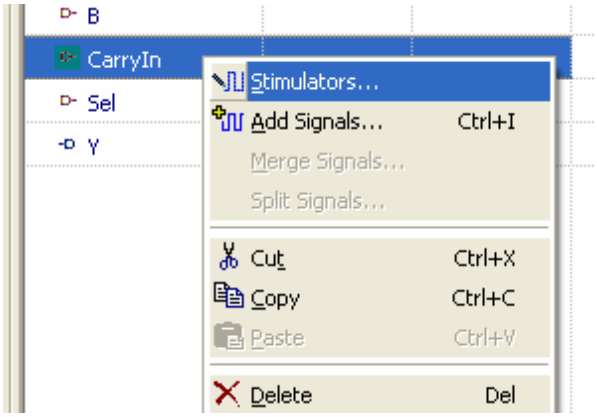
7. **Kompajlirati kod. U *Design browser* prozoru kliknuti desnim tasterom miša na snimljeni fajl i izabrati opciju *Compile*. Ukoliko je kompilacija izvršena korektno ispred imena fajla u *Design Browser* prozoru pojaviće se oznaka √. Ukoliko je došlo do greške, ispraviti grešku i ponovo kompajlirati kod.**



8. **Iz palete alata izabrati *Waveform Editor*. U dnu *Design browser* prozora izabrati karticu *Structure*.**

9. **U *Design browser* prozoru izabrati stavku *Root : ALU*. U donjem delu prozora pojaviće se ulazni (A,B,CarryIn i Sel) i izlazni (CarryOut i Y) signal. Signale „prevuci“ u prozor *Waveform Editor*.**

10. **U *Waveform Editor* prozoru, desnim tasterom miša izabrati određeni signal (Sel ili CarryIn) i izabrati opciju *Stimulators*. Iz *Type* prozora izabrati opciju *Hotkey* i dodeliti taster sa tastature koji će menjati vrednost odgovarajućem bitu signala.**



11. **Postupak ponoviti za svaki bit ulaznih signala Sel i CarryIn.**

12. **U *Waveform Editor* prozoru, desnim tasterom miša izabrati određeni signal i izabrati opciju *Stimulators*. Iz *Type* prozora izabrati opciju *Value* i dodeliti vrednost odgovarajućem bitu signala.**

13. **Postupak ponoviti za svaki bit ulaznih signala A i B.**

14. **Iz palete alata izabrati *Run.***

1.4. Zadaci za vežbu

Grupa 1.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4	S3	S2	S1	S0	Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0	0	0	0	0	0	_____	_____
Selektorski ulazi						izlazni signal	izlazni prenos

Grupa 2.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4	S3	S2	S1	S0	Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0	0	0	0	0	1	_____	_____
Selektorski ulazi						izlazni signal	izlazni prenos

Grupa 3.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4	S3	S2	S1	S0	Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0	0	0	0	1	0	_____	_____
Selektorski ulazi						izlazni signal	izlazni prenos

Grupa 4.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 0 0 1 1	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 5.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 0 1 0 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 6.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 0 1 0 1	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 7.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 0 1 1 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 8.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 0 1 1 1	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 9.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 1 0 0 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 10.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 1 0 1 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 11.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 1 1 0 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 12.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 0 1 1 1 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 13.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
0 1 0 0 0 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 14.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
1 0 0 0 0 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

Grupa 15.

Šta se dobija na izlazu (Y) za signale na ulazu A=10101010 i B=01010101 i odgovarajuće signale na selektorskim ulazima:

S4 S3 S2 S1 S0 Cin	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	CarryOut
1 1 0 0 0 0	_____	_____
Selektorski ulazi	izlazni signal	izlazni prenos

2. Množac (8x6 Ripple-Carry)

Binarni množač celobrojnih pozitivnih vrednosti, prihvata na ulazu dva binarna *n*-to bitna podatka, množenik *X* i množilac *Y*, a na izlazu generiše 2*n*-bitni rezultat *Z*. Rezultat predstavlja proizvod množenika i množioca (Z=X*Y). Kao i kod decimalnog množača, prvo se formira parcijalni proizvod bita najmanje težine množioca (Y0) sa svim bitovima množenika. Postupak se ponavlja za svaki sledeći bit množioca. U svakom koraku vrši se pomeranje parcijalnog proizvoda za jednu binarnu poziciju ulevo u odnosu na prethodni proizvod, kako bi proizvodi bili pozicionirani u skladu sa težinama bitova množioca. Da bi dobili konačan rezultat, svi parcijalni proizvodi se sabiraju, stim što dodajemo još jedan bit (bit najveće težine) u proizvodu, zbog mogućeg prenosa (carry).

2.1. Množač konkretno rešenje

U ovom primeru realizovan je množač koji obavlja množenje pozitivnih celih brojeva. Ulazni podaci, množenik i množilac su osmobitni i šestobitni, respektivno, dok je izlazni podatak četrnaestobitni.

Množenje se obavlja prema algoritmu množenja počev od cifre najmanje težine množioca, formiranjem parcijalnih proizvoda počev od proizvoda najmanje težine i njihovim sabiranjem.

Formalni opis postupka množenja je:
Ulazni podaci: $X=x_7x_6x_5x_4x_3x_2x_1x_0$ **i** $Y=y_5y_4y_3y_2y_1y_0$
Izlaz: $Z=X*Y$

$$z = x \cdot \left(\sum_{i=0}^{5} y_i \cdot 2\right)$$

Množenje sa 2^i je ekvivalentno pomeranju ulevo za *i* pozicija, što se ostvaruje pozicioniranjem operanada koji se sabiraju u skladu sa težinama.

Sabiranje parcijalnih proizvoda ostvaruje se korišćenjem 5 Ripple Carry sabirača. Ripple Carry sabirač sastoji se od 8 dvobitnih sabirača serijski povezanih u nizu. Implementacija *8x6* množača na ovaj način zahteva: *šest* nizova od po *osam* AND logičkih kola i *pet* Ripple Carry sabirača.

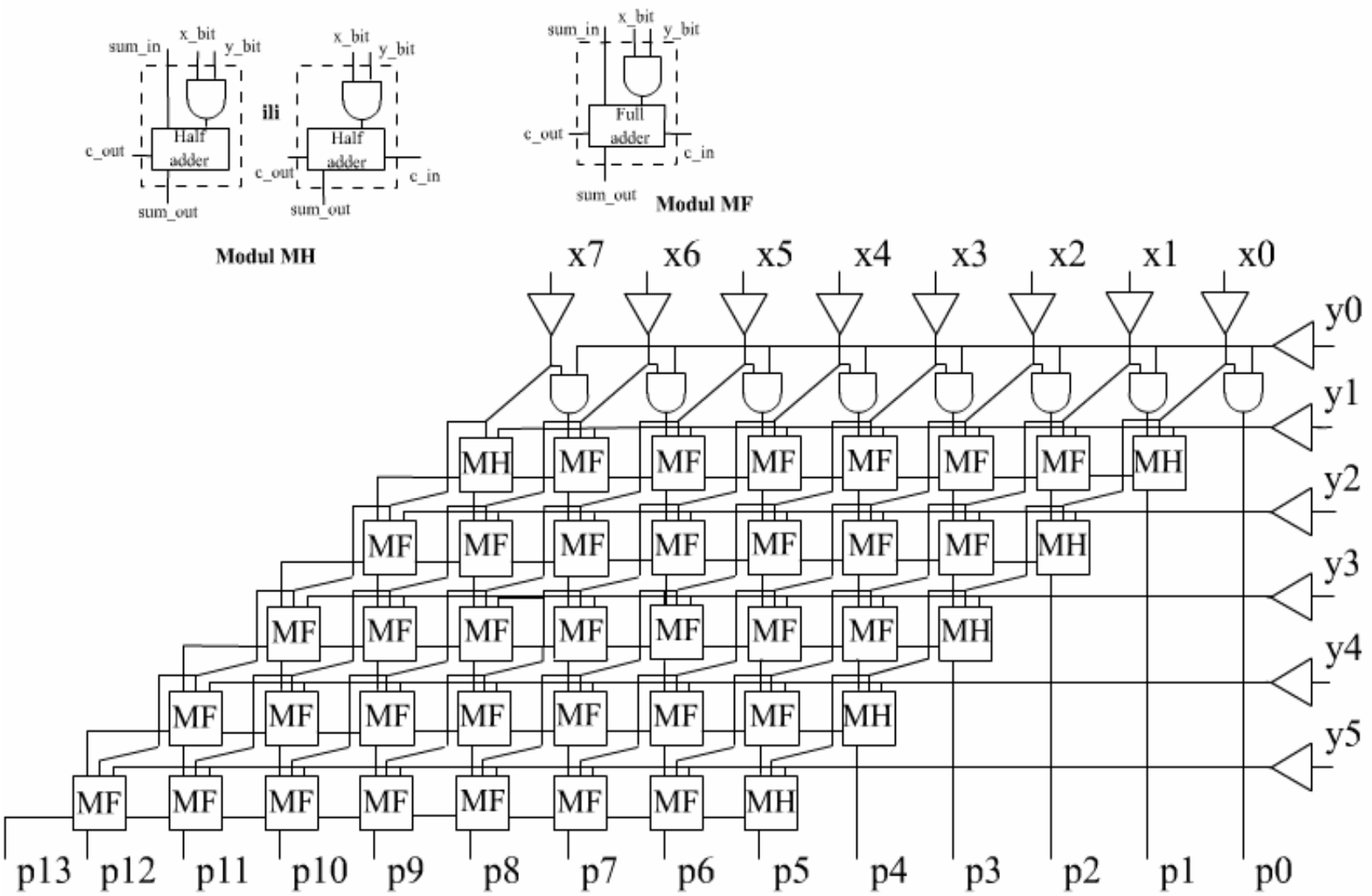
U konkretnom slučaju za modeliranje množača korišćena je arhitektura specificirana na *behavioral* nivou, koja opisuje ponašanje komponente, zanemarujući njenu unutrašnju strukturu. (Pored ovakvog načina modeliranja moguće je koristiti i strukturalnu ili *dataflow* konstrukciju.) Imajući u vidu da je množač kombinaciona logička mreža, za sintezu je

korišćena kombinaciona logika (bez vremenskih parametara, sve operacije koje se obavljaju trenutno reaguju na promenu ulaznih signala), nasuprot sekvencijalnoj (gde se promene dešavaju u vremenskim trenucima određenim taktnim signalom (clock)).

Prvi parcijalni proizvod se dobija množenjem *Y0* bita sa signalom *X(X7:X0)*. Množenje se obavlja AND kolom. Za sabiranje parcijalnih proizvoda koristi se paralelni sabirač sa serijskim prenosom (Ripple Carry sabirač) koji je realizovan pomoću *MH* i *MF* modula (struktura modula je data na Slici 2.). Ukupno ima 5 sabirača. *MH* modul je realizovan pomoću polusabirača (Half adder) i jednog AND kola. U zavisnosti od potrebe, koristimo *MH* modul sa *C_In* ulazom ili sa *Sum_In* ulazom. *MF* se sastoji od jednog potpunog sabirača (Full adder) i jednog AND kola. Za razliku od *MH* modula, *MF* modul sadrži i *C_in* i *Sum_in* ulaze.

Implementacija množača tipa *8x6* bita, korišćenjem paralelnih sabirača sa serijskim prenosom (RCA-Ripple-Carry Adder) prikazana je na slici 2.:

Slika 2.


sum_in
x_bit
y_bit
c_out
Half adder
sum_out
ili
c_in
Modul MH
Full adder
Modul MF
x7 x6 x5 x4 x3 x2 x1 x0
y0 y1 y2 y3 y4 y5
MH MF
p13 p12 p11 p10 p9 p8 p7 p6 p5 p4 p3 p2 p1 p0

2.2. VHDL kôd množača

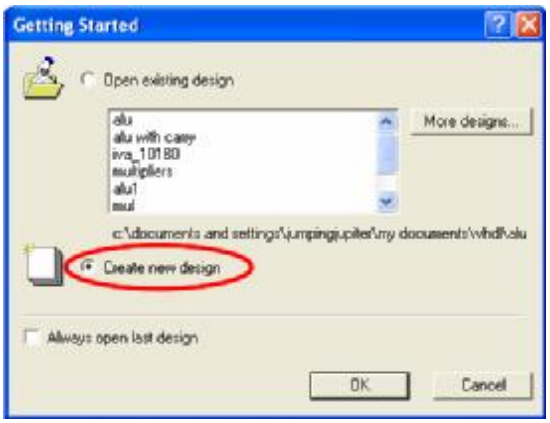
```
library IEEE;
use IEEE.std_logic_1164.all, IEEE.NUMERIC_STD.all;

entity multipliers is
        port (
                X: in unsigned (7 downto 0);
                Y: in unsigned (5 downto 0);
                Z: out unsigned (13 downto 0)
        );
end entity multipliers;

architecture mreza of multipliers is
begin
 MNOZAC :
 process(X,Y)
   variable izlaz : unsigned (8 downto 0);
   variable Carry  : std_ulogic;
        variable j,h  : integer;
        begin
                ----------------------------
                -- Prvi nivo
                ----------------------------
                Z(0) <= (X(0) and Y(0));
                for i in 0 to 6 loop
                        izlaz(i) := (X(i+1) and Y(0));
                end loop;
                izlaz(7) := X(7);
                ----------------------------
                -- drugi nivo
                ----------------------------
                Z(1) <= ((X(0) and Y(1)) xor izlaz(0));
                Carry := ((X(0) and Y(1)) and izlaz(0));
                j := 1;
                for i in 1 to 6 loop
                        izlaz(i-1) := ((X(i) and Y(j)) xor izlaz(i)) xor Carry;
                        Carry := ((X(i) and Y(j)) and izlaz(i)) or (((X(i) and Y(j)) xor izlaz(i)) and
Carry);
                end loop;
                        izlaz(6) := ((Y(1) and X(7)) xor Carry);
                        izlaz(7) := ((Y(1) and X(7)) and Carry);


                -------------------------
                -- treci i nivoi do kraja
```

```
        -------------------------
        j := 2;
        for h in 1 to 4 loop
                Z(h+1) <= ((X(0) and Y(h+1)) xor izlaz(0));
                Carry := ((X(0) and Y(h+1)) and izlaz(0));
                for i in 1 to 7 loop
                        izlaz(i-1) := ((X(i) and Y(j)) xor izlaz(i)) xor Carry;
                        Carry := ((X(i) and Y(j)) and izlaz(i)) or (((X(i) and Y(j)) xor
izlaz(i)) and Carry);
                end loop;
                izlaz(7) := Carry;
                j := j + 1;
        end loop;

        ----------------------------
        -- krajnje stanje
        ----------------------------
        Z(6) <= izlaz(0);
        Z(7) <= izlaz(1);
        Z(8) <= izlaz(2);
        Z(9) <= izlaz(3);
        Z(10) <= izlaz(4);
        Z(11) <= izlaz(5);
        Z(12) <= izlaz(6);
        Z(13) <= izlaz(7);


    end process MNOZAC;
end architecture mreza;
```

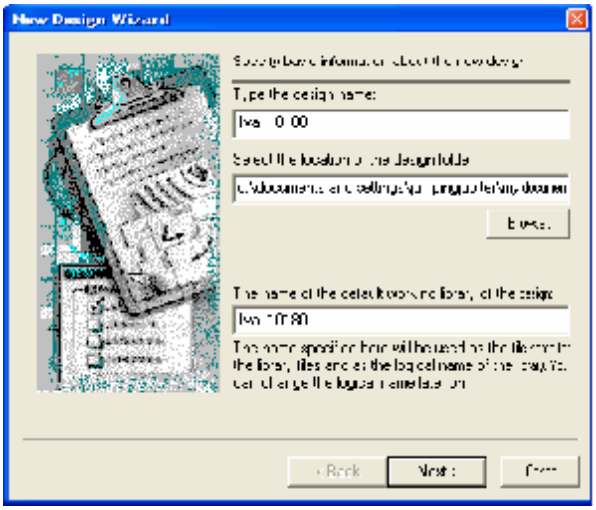
2.3. Vežba 2 – Testiranje množača

1. **Pokrenuti program *Active-HDL*.** 

2. **U prozoru *Getting Started* selektovati opciju *Create new design* i kliknuti na *OK*.**

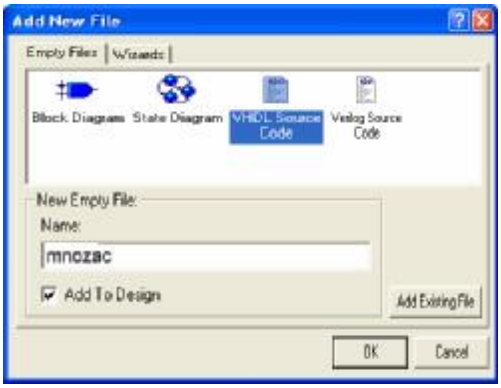


3. **U novootvorenom prozoru za naziv dizajna upisati *ime_broj_indexa* (na.pr. Iva_10180) i klinuti na *Next*.**



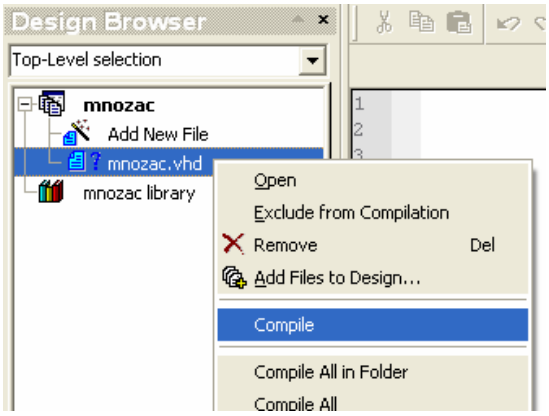
4. **Selektovati *Create an empty design* i kilknuti na *Next* a zatim *Finish*.**

5. **U prozoru *Design browser* dva puta kliknuti na *Add new file*. Za tip fajla izabrati *VHDL Source Code* i imenovati ga sa *Mnozac*. Selektovati opciju *Add To Design* i kliknuti na *OK*.**



6. **U prozoru editora prekucati dati kod, i obratiti pažnju da li su svi ulazni i izlazni portovi definisani.**

7. **Kompajlirati kod. U *Design browser* prozoru kliknuti desnim tasterom misa na snimljeni fajl i izabrati opciju Compile. Ukoliko je kompalacija izvršena korektno ispred imena fajla u *Design Browser* prozoru pojaviće se oznaka √. Ukoliko je došlo do greške, ispraviti grešku i ponovo kompajlirati kod.**

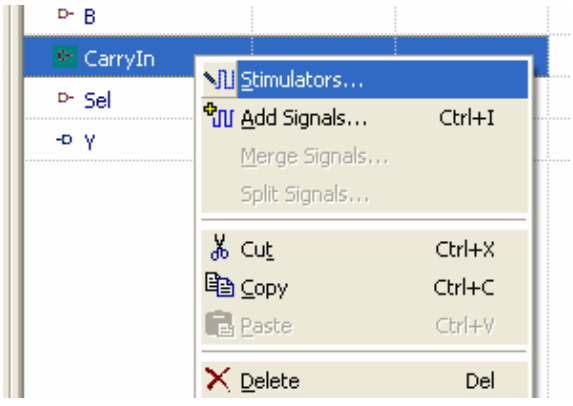


8. **Iz palete alata izabrati *Waveform Editor*. U dnu *Design browser* prozora izabrati karticu *Structure*.**

9. **U *Design browser* prozoru izabrati stavku *Root : Mnozac*. U donjem delu prozora pojavice se ulazni (X i Y) i izlazni (Z) signal. Signale „prevuci“ u prozor *Waveform Editor*.**

10. **Iz palete alata izabrati *Run.***

11. U ***Waveform Editor* prozoru, desnim tasterom misa izaabrati odredjeni signal i izabrati opciju *Stimulators*. Iz *Type* prozora izabrati opciju *Value* i dodeliti vrednost odgovarajucem bitu signala.**



12. **Postupak ponoviti za svaki bit ulaznih signala.**

13. **Iz palete alata izabrati *Run.***

2.4. Zadaci za vežbu

Grupa 1.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
0	0	0	0	0	0	0	1	1	0	0	0	0	1															
množenik								množilac						rezultat														izl.prenos

Grupa 2.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
0	0	0	0	0	1	0	1	1	0	1	0	0	1															
množenik								množilac						rezultat														izl.prenos

Grupa 3.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
0	1	0	1	0	1	0	1	1	0	1	0	0	1															
množenik								množilac						rezultat														izl.prenos

Grupa 4.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
1	0	0	1	1	0	0	1	1	0	1	1	0	1															
množenik								množilac						rezultat														izl.prenos

Grupa 5.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
1	1	1	1	1	1	1	1	1	1	1	1	1	1															

množenik množilac rezultat izl.prenos

Grupa 6.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
1	1	0	0	1	0	1	1	1	1	0	0	1	1															

množenik množilac rezultat izl.prenos

Grupa 7.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
0	0	0	0	0	1	1	1	0	0	0	0	1	1															

množenik množilac rezultat izl.prenos

Grupa 8.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
0	0	0	1	1	1	1	1	0	0	0	1	1	1															

množenik množilac rezultat izl.prenos

Grupa 9.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
1	1	1	1	0	0	0	0	1	0	0	0	0	0															

množenik množilac rezultat izl.prenos

Grupa 10.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7	X6	X5	X4	X3	X2	X1	X0	Y5	Y4	Y3	Y2	Y1	Y0	Z13	Z12	Z11	Z10	Z9	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	Cout
0	0	0	1	0	0	0	0	1	1	1	1	1	1															

množenik množilac rezultat izl.prenos

Grupa 11.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7 X6 X5 X4 X3 X2 X1 X0	Y5 Y4 Y3 Y2 Y1 Y0	Z13 Z12 Z11 Z10 Z9 Z8 Z7 Z6 Z5 Z4 Z3 Z2 Z1 Z0	Cout
1 0 1 0 1 0 1 0	0 0 1 0 0 1	_____	____
množenik	množilac	rezultat	izl.prenos

Grupa 12.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7 X6 X5 X4 X3 X2 X1 X0	Y5 Y4 Y3 Y2 Y1 Y0	Z13 Z12 Z11 Z10 Z9 Z8 Z7 Z6 Z5 Z4 Z3 Z2 Z1 Z0	Cout
0 0 0 1 1 0 0 1	0 0 1 0 0 1	_____	____
množenik	množilac	rezultat	izl.prenos

Grupa 13.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7 X6 X5 X4 X3 X2 X1 X0	Y5 Y4 Y3 Y2 Y1 Y0	Z13 Z12 Z11 Z10 Z9 Z8 Z7 Z6 Z5 Z4 Z3 Z2 Z1 Z0	Cout
0 0 1 1 0 1 0 1	0 1 1 0 1 0	_____	____
množenik	množilac	rezultat	izl.prenos

Grupa 14.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7 X6 X5 X4 X3 X2 X1 X0	Y5 Y4 Y3 Y2 Y1 Y0	Z13 Z12 Z11 Z10 Z9 Z8 Z7 Z6 Z5 Z4 Z3 Z2 Z1 Z0	Cout
1 0 1 0 1 0 0 1	1 0 1 0 1 1	_____	____
množenik	množilac	rezultat	izl.prenos

Grupa 15.

Za date signale na ulazu proveriti šta se dobija na izlazu:

X7 X6 X5 X4 X3 X2 X1 X0	Y5 Y4 Y3 Y2 Y1 Y0	Z13 Z12 Z11 Z10 Z9 Z8 Z7 Z6 Z5 Z4 Z3 Z2 Z1 Z0	Cout
0 0 1 1 1 0 0 1	1 0 0 1 1 1	_____	____
množenik	množilac	rezultat	izl.prenos