VISOKA TEHNIČKA ŠKOLA

STRUKOVNIH STUDIJA

SUBOTICA

ALU sa 4 bita

Bez namenskih kola

projekat

iz predmeta: Arhitektura računara

mentor: Dr. Tibor Sakal

profesor strukovnih studija

kandidat :David Katrinka

br.indeksa : 26123035

studijski program: Informatika

Sadržaj

Subotica, 2024.

UVOD

Ovim projektom prikazujemo aritmetičko logičku jedinicu od 4 bita, implementiranu bez korišćenja namenskih kola.

Naša aritmetičko logička jedinica se sastoji od logičkog dela (AND, OR, XOR, NOT, NAND i NOR) aritmetičkog dela (sabiranje, oduzimanje, negacija, inkrement i dekrement) i od komparatora (> ,<, =, >= ,<= ,!=). Pošto imamo ALU od 4 bita znači da su nam ulazi i izlazi četvorobitni brojevi. Osim brojeva sa kojima baratamo, imamo ulaze za izbor operacije i za kontrolu komparatora. Na izlazu pored rezultata od 4 bita, imamo izlaz za Carry/Borrow i Overflow bitove kao i za komparator. Limitirani smo time što ne možemo da koristimo namenska kola (kao što su 74HC283 ili 74HC85), ali možemo da ih zamenimo koristeći module napravljene sa osnovnim logičkim kolima.

Ovaj dokument služi kao dodatak projektu napravljenom u Proteus, programu za digitalnu simulaciju.

1. PROJEKTNI ZADATAK

Izabrana tema: 36-E6

Naziv teme: ALU sa 4 bita

Dodatni zahtev: Bez namenskih kola

1.1 Svrha projekta i ograničenja

Svrha projekta je prikaz rada aritmetičko logičke jedinice pomoću Proteus-ove simulacije bez namenskih kola. Svrha dodatnog zahteva je da pokažemo da je moguće napraviti svako namensko kolo pomoću osnovnih kola. Glavno ograničenje je 4 bita, imamo samo brojeve od 0 do 15 tj. Od -8 do 7 ako koristimo brojeve sa predznakom. Simulacija mora biti proverena za sve kombinacije od dva četvorobitna broja.

1.2 Način realizacije

Projekat je realizovan kroz implementaciju najosnovnijih modula (npr. pun sabirač) pomoću osnovnih logičkih kapija, pa implementacijom složenijih modula koristeći nove module(npr. četvorobitni sabirač od četiri puna sabirača). Nakon implementacije svaki modul je posebno testiran van ALU i kasnije testiran u njoj. Finalni modul će biti prikazan šemama i njegova funkcija će biti prikazana simulacijom i tablicama istinitosti. Funkcije naše ALU su: AND, OR, XOR, NOT, NAND, NOR, sabiranje, oduzimanje, inkrement, dekrement i upoređivanje dva četvorobitna broja.

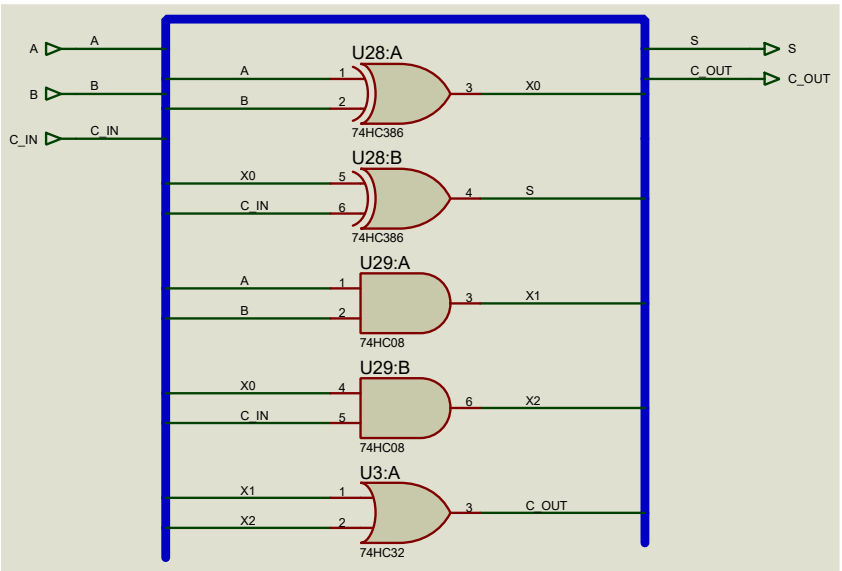
2. TEORIJSKE OSNOVE

2.1 Aritmetički deo

2.1.1 Modul za sabiranje

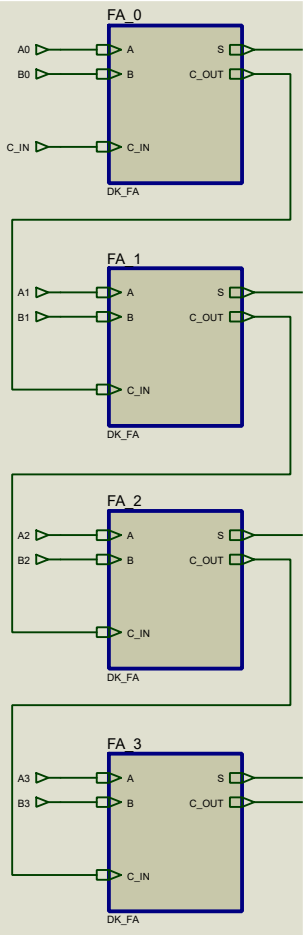
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabela 1. – tablica istinitosti za 1-bitni pun sabirač



Slika 1.- šema 1-bitnog punog sabirača

Uz pomoć dva XOR, dva AND i jednog OR kola sastavljamo pun sabirač sa Carry in i Carry out, koji sabira dva broja od jednog bita.



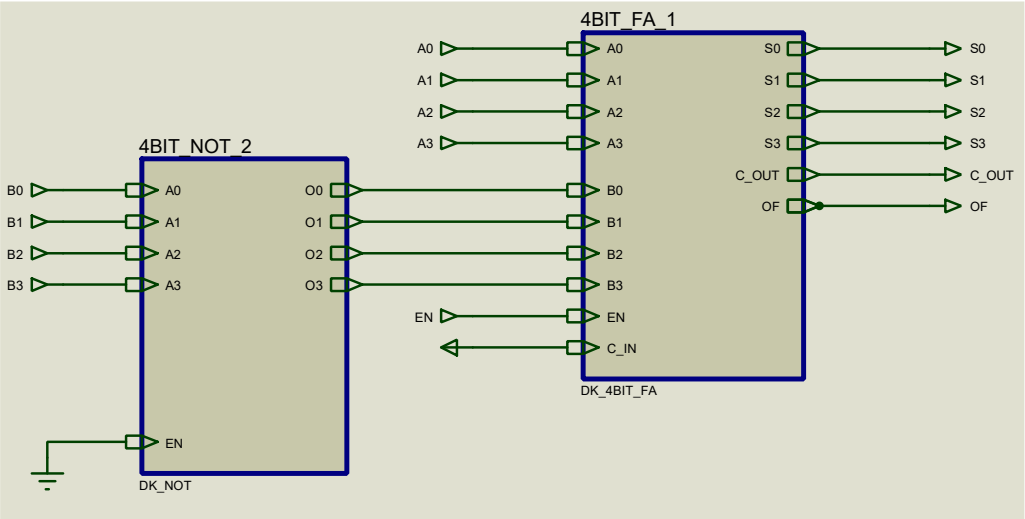
Slika 2.- šema 4-bitnog sabirača

Lančanjem 1-bitnih sabirača (iz slike 1.) tako što Carry out svakog prethodnog stavljamo u Carry in svakog sledećeg. Prvi Carry in je 0, poslednji Carry out je Carry out celog sabirača od 4 bita.

2.1.2 Modul za oduzimanje

A	B	S	Cout
0	0	0	1
0	1	0	0
1	0	1	1
1	1	0	1

Tabela 2. – tablica istinitosti za 1-bitni oduzimač



Slika 3. – Šema 4-bitnog oduzimača

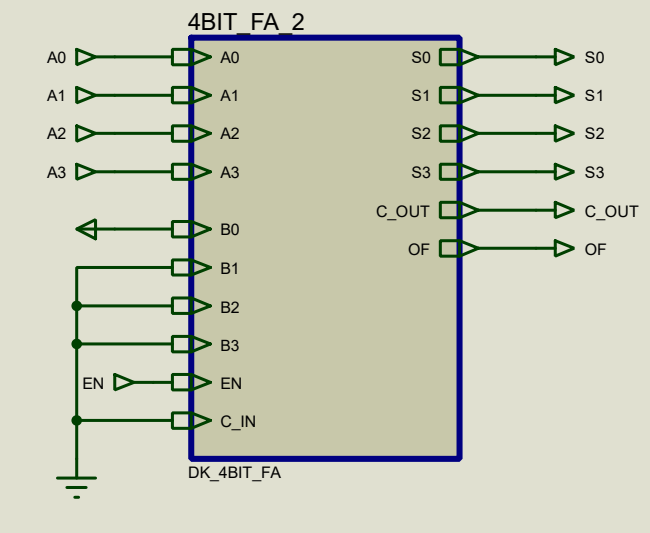
Oduzimač smo napravili koristeći 4-bitni sabirač (2.1.1) i 4-bitni NOT (2.2.1), tako da umesto A – B imamo A + (-B). Broj A ubacujemo u sabirač nepromenjenog, a broj B ubacujemo u NOT, pa dobijemo B u prvom komplementu. Pošto nam treba drugi komplement, umesto da inkrementiramo B, postavljamo Cin na Power tj. na 1 i sa tim dobijamo -B (broj B u drugom komplementu).

U tabeli 2 imamo prikaz 1 bitnog modula za oduzimanje jer bi tablica istinitosti za 4 bitni oduzimač bila prevelika za ovaj rad.

2.1.3 Modul za inkrement

A3	A2	A1	A0	R3	R2	R1	R0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

Tabela 3. – tablica istinitosti za modul za inkrement



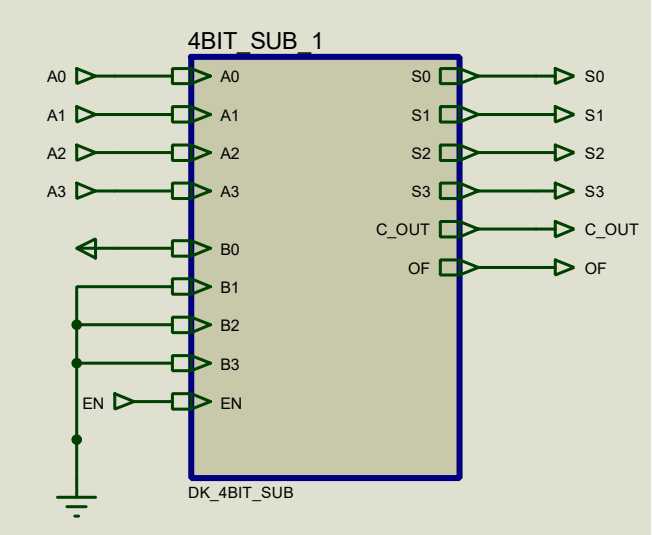
Slika 4. – šema 4.bitnog modula za inkrement

Modul za inkrement smo dobili koristeći jedan 4-bitni sabirač (2.1.1). Broj koji želimo da inkrementiramo smo stavili na ulaz A. Na ulazu B stavili smo LSB na Power (1), a ostale bitove na Ground (0). Tako da u sabiraču imamo A + 0001.

2.1.4 Modul za dekrement

A3	A2	A1	A0	R3	R2	R1	R0
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

Tabela 4. – tablica istinitosti za modul za dekrement



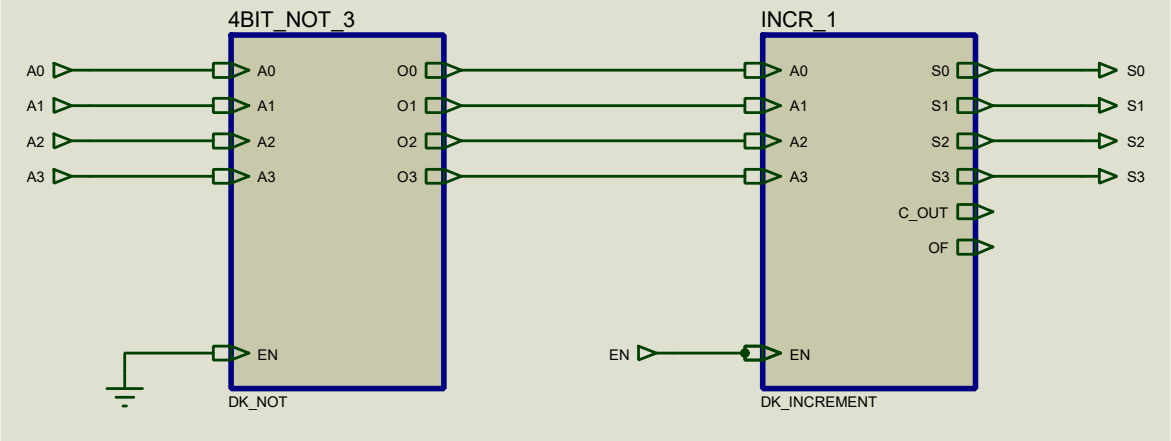
Slika 5. – šema 4.bitnog modula za dekrement

Modul za dekrement smo dobili koristeći jedan 4-bitni oduzimač (2.1.2). Broj koji želimo da dekrementiramo smo stavili na ulaz A. Na ulazu B stavili smo LSB na Power (1), a ostale bitove na Ground (0). Tako da u oduzimaču imamo A – 0001.

2.1.5 Modul za drugi komplement

A3	A2	A1	A0	R3	R2	R1	R0
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

Tabela 5. – tablica istinitosti za modul za drugi komplement



Slika 6. – šema 4.bitnog modula za drugi komplement

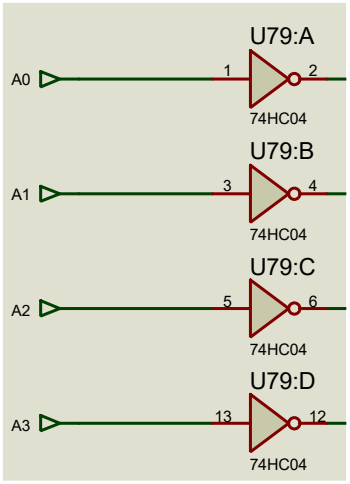
Modul za drugi komplement smo dobili, prvo ubacivanjem našeg ulaza (A) u NOT modul (2.2.1) posle čega izlaz iz NOT modula (prvi komplement) ulazi u modul za inkrement (2.1.3) i postaje drugi komplement tj. na izlazu je negiran ulazni broj (-A).

2.2 Logički deo

2.2.1 NOT modul

A	R
0	1
1	0

Tabela 6. – tablica istinitosti za NOT logičku funkciju



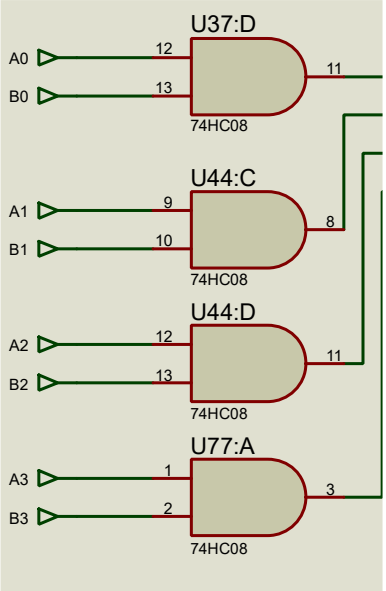
Slika 7. – šema 4-bitnog NOT modula

Sa 4 NOT logičke kapije svaki bit ulaza negiramo posebno.

2.2.2 AND modul

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 6. – tablica istinitosti za AND logičku funkciju



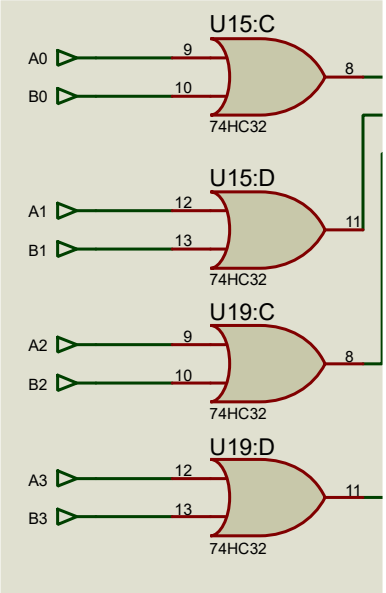
Slika 7. – šema 4-bitnog AND modula

Sa 4 AND logičke kapije, radimo AND svakog bita broja A sa svakim bitom broja B (A0 AND B0, A1 AND B1…).

2.2.3 OR modul

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 7. – tablica istinitosti za OR logičku funkciju



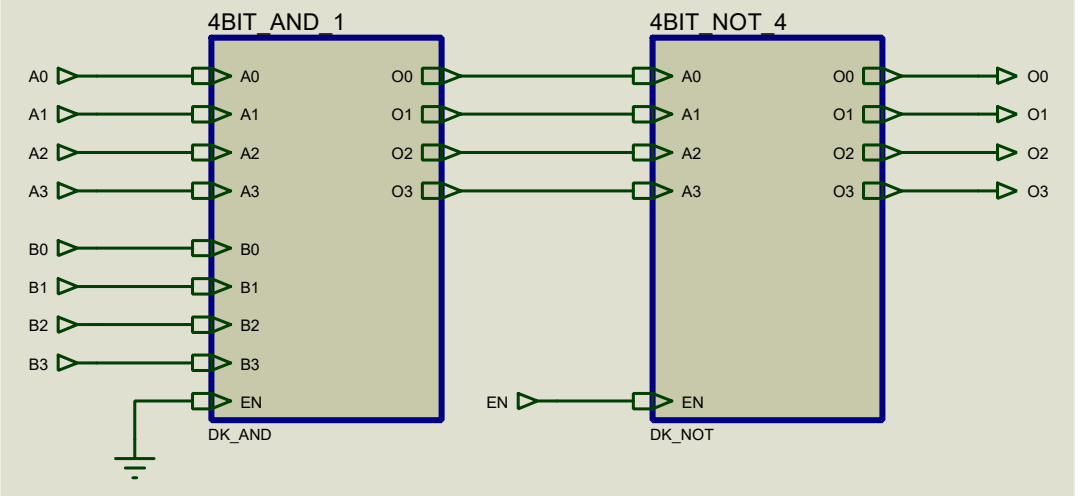
Slika 8. – šema 4-bitnog OR modula

Sa 4 OR logičke kapije, radimo OR svakog bita broja A sa svakim bitom broja B (A0 OR B0, A1 OR B1…).

2.2.4 NAND modul

A	B	R
0	0	1
0	1	1
1	0	1
1	1	0

Tabela 8. – tablica istinitosti za NAND logičku funkciju



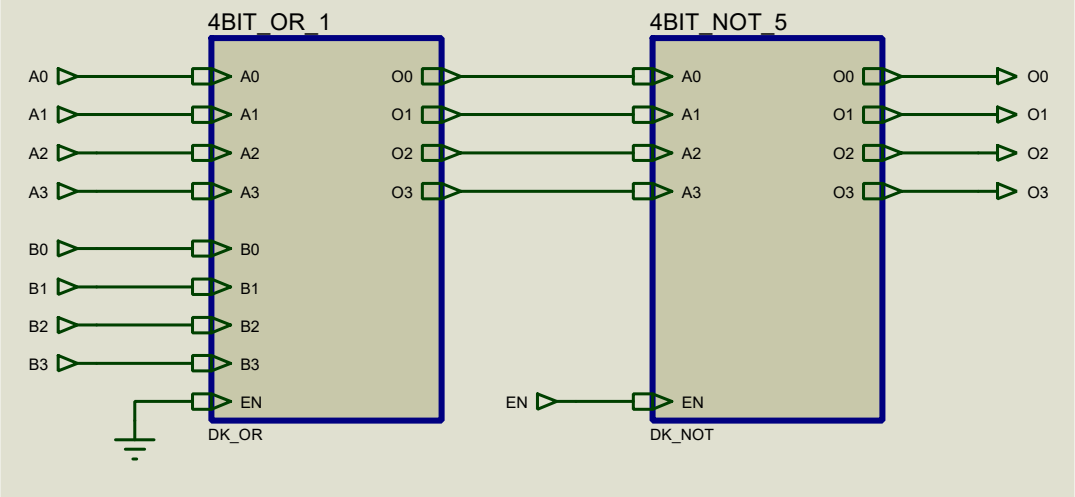
Slika 9. – šema 4-bitnog NAND modula

Koristili smo AND (2.2.2) i NOT (2.2.1) module kako bismo dobili 4-bitni NAND modul (A0 NAND B0, A1 NAND B1…).

2.2.5 NOR modul

A	B	R
0	0	1
0	1	0
1	0	0
1	1	0

Tabela 9. – tablica istinitosti za NOR logičku funkciju



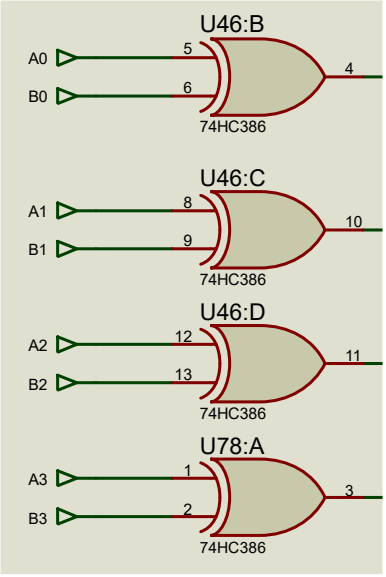
Slika 10. – šema 4-bitnog NOR modula

Koristili smo OR (2.2.3) i NOT (2.2.1) module kako bismo dobili 4-bitni NOR modul (A0 NOR B0, A1 NOR B1…).

2.2.6 XOR modul

A	B	R
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 10. – tablica istinitosti za XOR logičku funkciju



Slika 11. – šema 4-bitnog XOR modula

Sa 4 XOR logičke kapije, radimo XOR svakog bita broja A sa svakim bitom broja B (A0 XOR B0, A1 XOR B1…).

2.3 Komparator

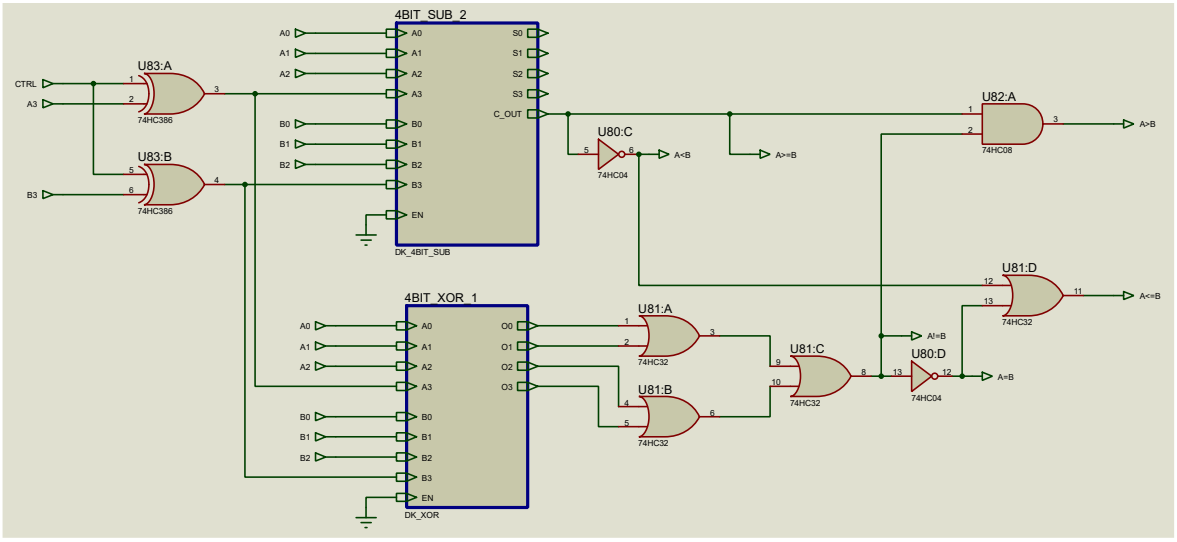
A	B	A>B	A<B	A=B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Tabela 11. – tablica istinitosti za 1-bitni komparator

A3B3	A2B2	A1B1	A0B0	A>B	A<B	A=B	A>=B	A<=B	A!=B
A3>B3	X	X	X	1	0	0	1	0	1
A3<B3	X	X	X	0	1	0	0	1	1
A3=B3	A2>B2	X	X	1	0	0	1	0	1
A3=B3	A2<B2	X	X	0	1	0	0	1	1
A3=B3	A2=B2	A1>B1	X	1	0	0	1	0	1
A3=B3	A2=B2	A1<B1	X	0	1	0	0	1	1
A3=B3	A2=B2	A1=B1	A0>B0	1	0	0	1	0	1
A3=B3	A2=B2	A1=B1	A0<B0	0	1	0	0	1	1
A3=B3	A2=B2	A1=B1	A0=B0	0	0	1	1	1	0

Tabela 12. – tablica istinitosti za 4-bitni komparator

Tabela 11. je za 1-bitni komparator i koristimo je da pojednostavimo tabelu za 4-bitni komparator (tabela 12.). Tabela za 4-bitni koristi rezultate komparacije svakog bita posebno.



Slika 12. – šema 4-bitnog komparatora

Komparator smo napravili koristeći modul za oduzimanje (2.1.2), XOR modul (2.2.6), XOR, OR, NOT i AND logička kola.

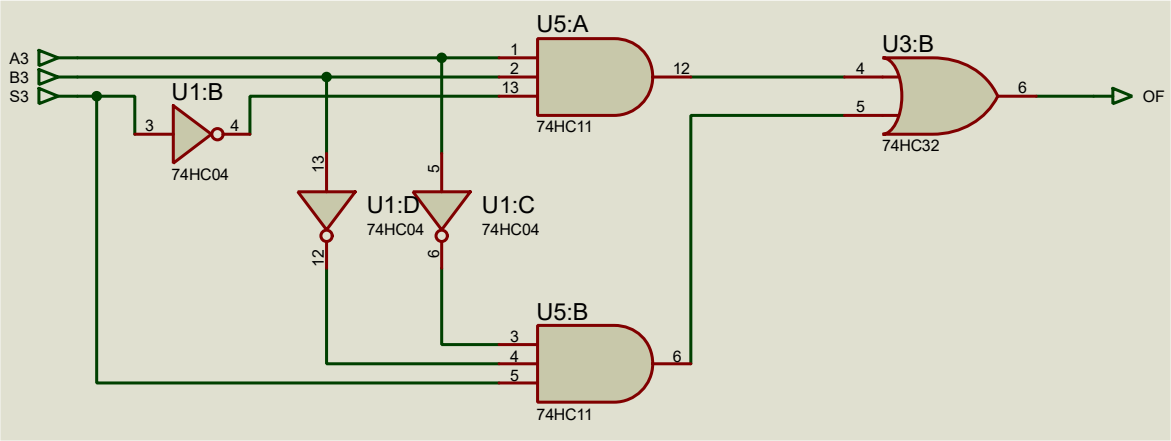
Koristili smo oduzimač jer znamo da, kada je Borrow out (c_out na slici 12.) 1, prvi broj mora biti veći ili jednak sa drugim. Kada je Borrow out 0 prvi broj mora biti manji od drugog. Znači Borrow out vežemo direktno sa izlazom A >= B i negaciju (NOT) Borrow out izlaza sa A < B. Sa XOR modulom i OR kolima proveravamo da li je A = B. Ako je bar jedan izlaz XOR modula 1 znamo da A != B i negacijom toga (NOT) dobijamo izlaz A = B. Za A > B ubacujemo A >= B i A != B u AND kolo. Za A <= B ubacujemo A < B i A = B u OR kolo. Na kraju za komparaciju Z brojeva koristimo dodatni input-CTRL i XOR kola. Kada je CTRL 0 kompariramo N brojeve i ulazi ostaju nepromenjeni, a kada je 1 kompariramo Y brojeve tako što MSB (bitove koji određuju predznak broja) negiramo pomoću XOR kola.

2.4 Dodatni moduli

2.4.1 Overflow modul

A3	B3	S3	OF
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Tabela 13. – tablica istinitosti za overflow bit



Slika 13. – šema overflow modula

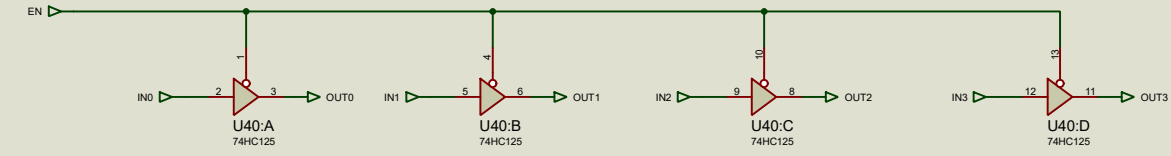
Uz NOT, AND i OR kola proveravamo da li je izlaz kod predznačnog sabiranja i oduzimanja validan. Overflow bit je 1 ako je predznak (MSB) ulaza isti, ali se predznak izlaza ne poklapa sa njihovim. Ako je overflow 1 znamo da je rezultat izvan opsega od 4 bita.

Overflow bit ima stanje visoke impedanse kada se ne bavimo aritmetičkim funkcijama ili za drugi komplement.

2.4.2 Buffer modul

IN3	IN2	IN1	IN0	EN	OUT3	OUT2	OUT1	OUT0
X	X	X	X	1	hi-Z	hi-Z	hi-Z	hi-Z
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1
1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1
1	0	1	0	0	1	0	1	0
1	0	1	1	0	1	0	1	1
1	1	0	0	0	1	1	0	0
1	1	0	1	0	1	1	0	1
1	1	1	0	0	1	1	1	0
1	1	1	1	0	1	1	1	1

Tabela 14. – tablica istinitosti za buffer modul



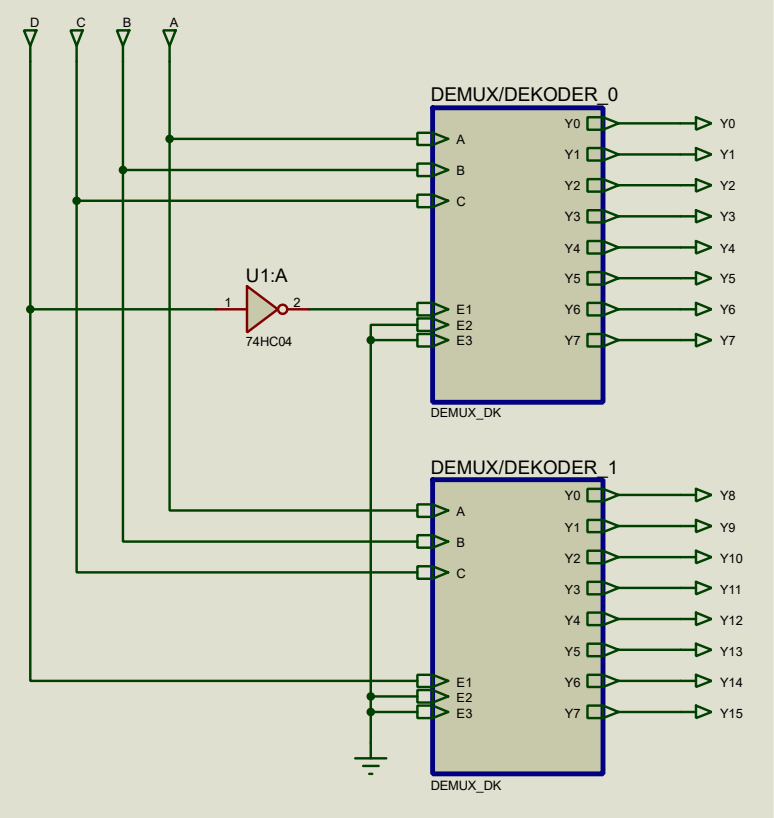
Slika 14. – šema buffer modula

Pošto imamo mnogo funkcija koristimo buffer modul za gašenje određenih izlaza koji nam ne trebaju. Pošto ima negativnu logiku on omogućava izlaze kada je EN = 0, a gasi ih kada je 1. Ovaj modul koristimo posle svakog drugog modula (osim komparatora) i u EN stavljamo izlaz dekodera sa negativnom logikom, tako možemo da biramo koju funkciju želimo na izlazu.

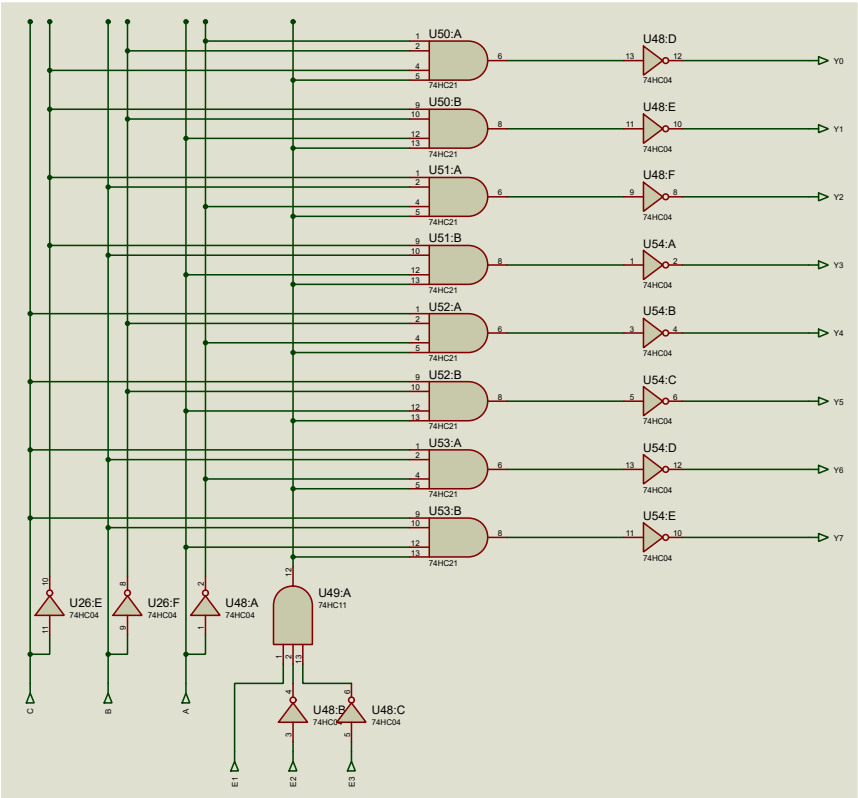
2.4.2 Dekoder sa negativnom logikom

0000 => A + B
0001 => A - B
0010 => -A
0011 => -B
0100 => A++
0101 => A--
0110 => A AND B
0111 => A OR B
1000 => A XOR B
1001 => NOT A
1010 => NOT B
1011 => A NAND B
1100 => A NOR B
1101 => NONE
1110 => NONE
1111 => NONE

Slika 15. – izbor funkcije kod 4 na 16 dekodera sa negativnom logikom



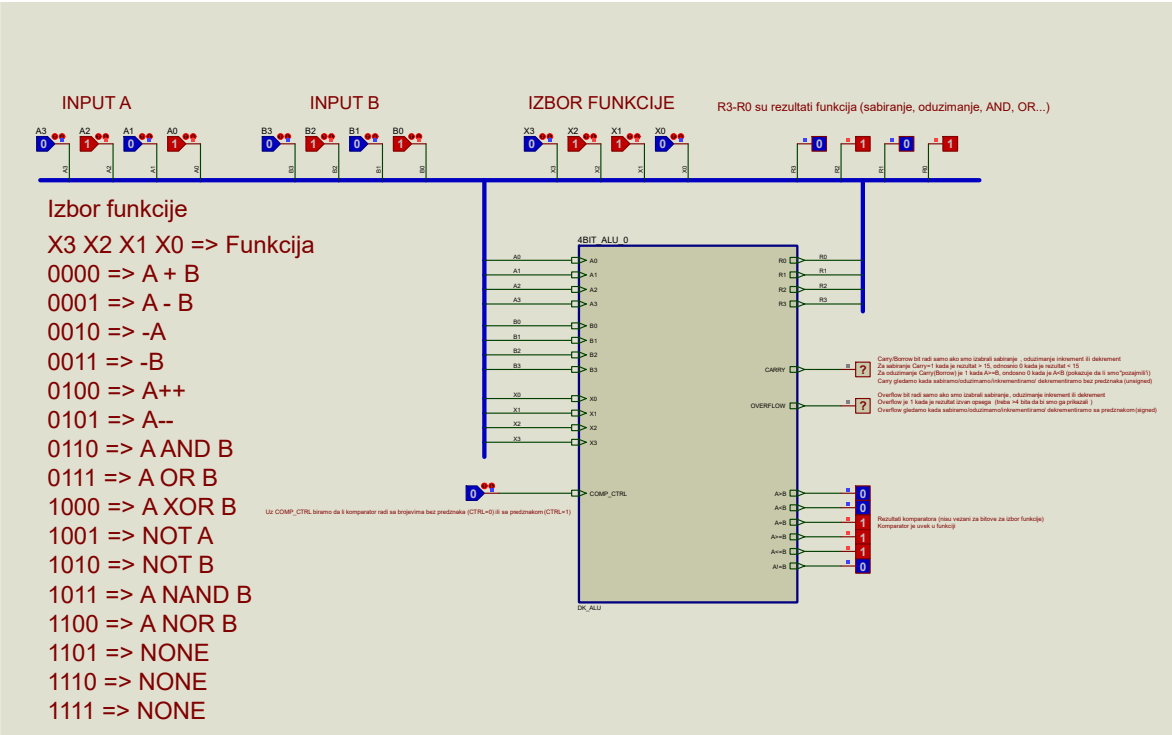
Slika 16. – šema 4 na 16 dekodera sa negativnom logikom



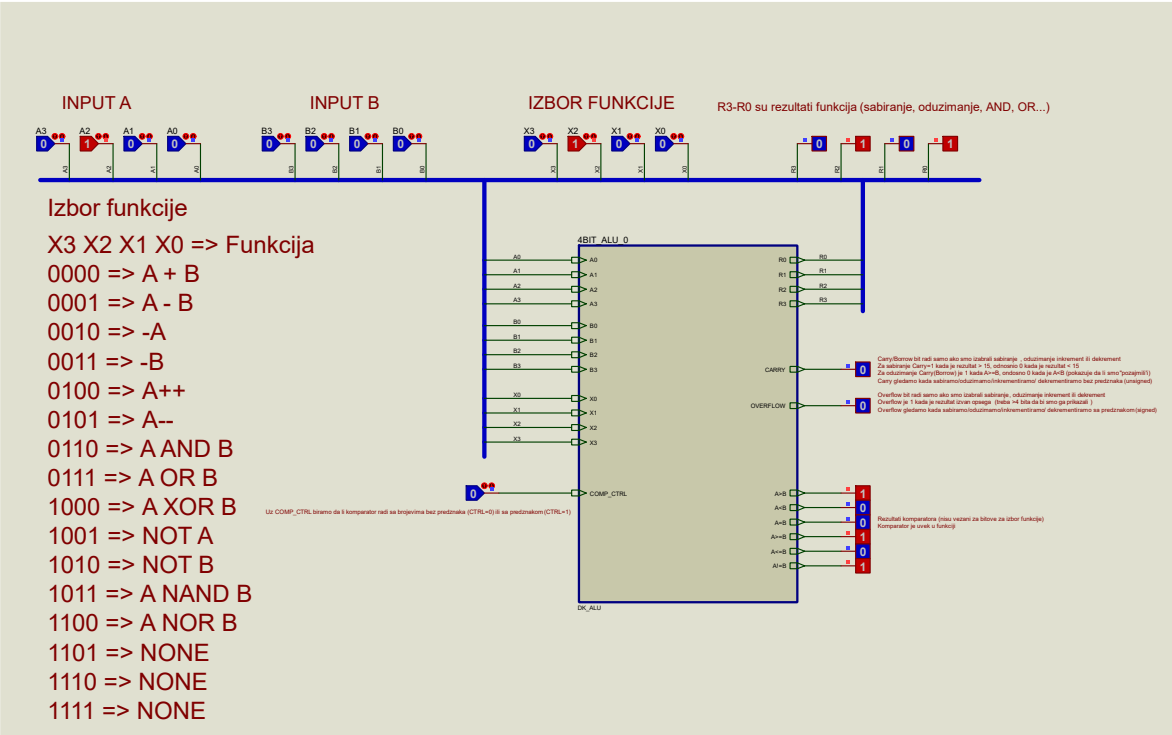
Slika 17. – šema 3 na 8 dekodera sa negativnom logikom

Subotica, 2024.

3. PRIKAZ SIMULACIJE

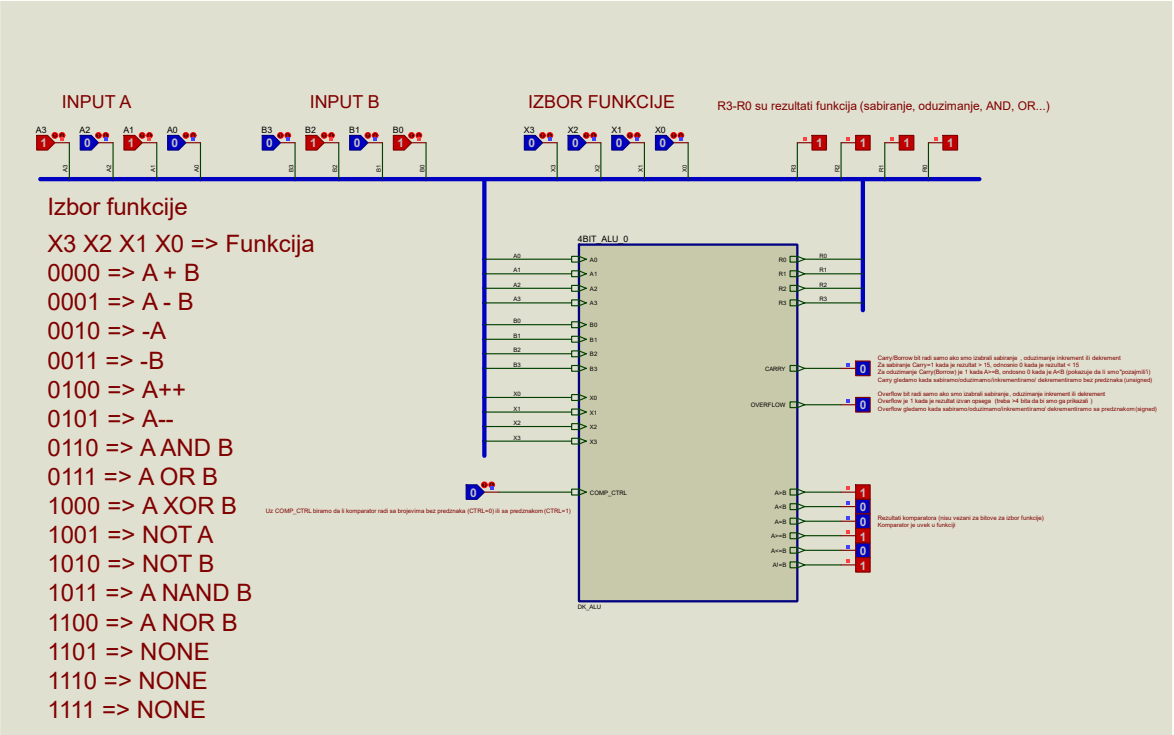


Slika 18. – A AND B

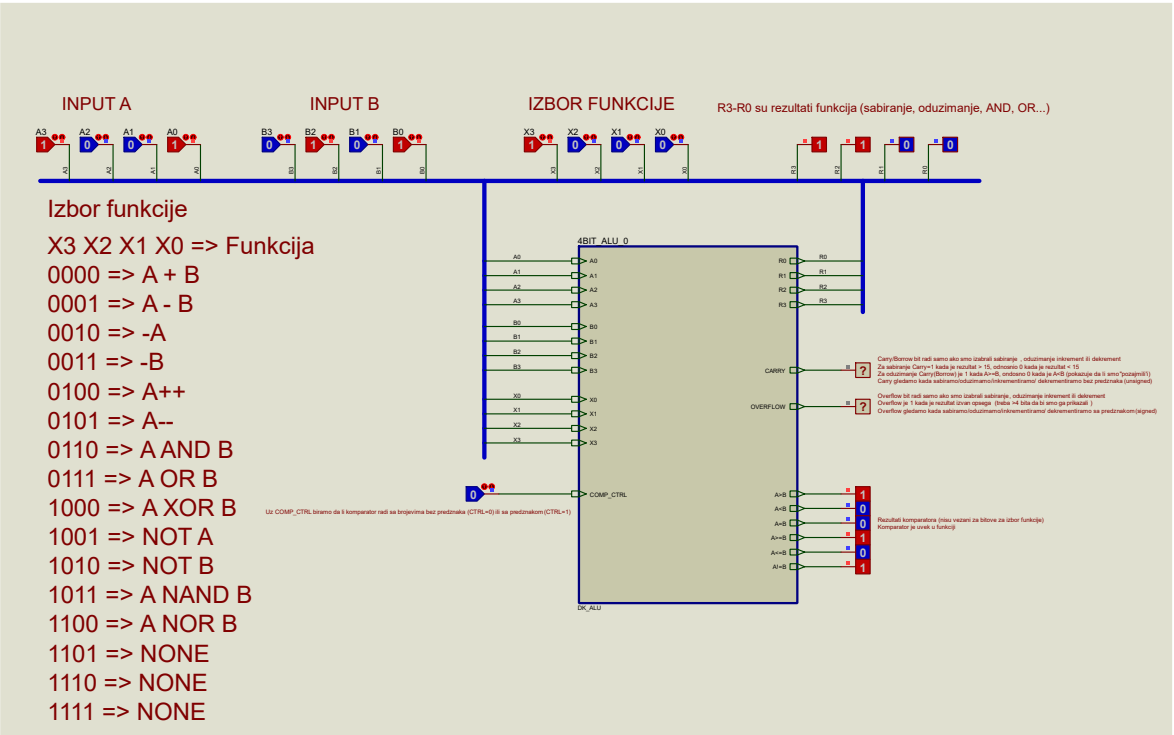


Slika 19. – A++

Subotica, 2024.

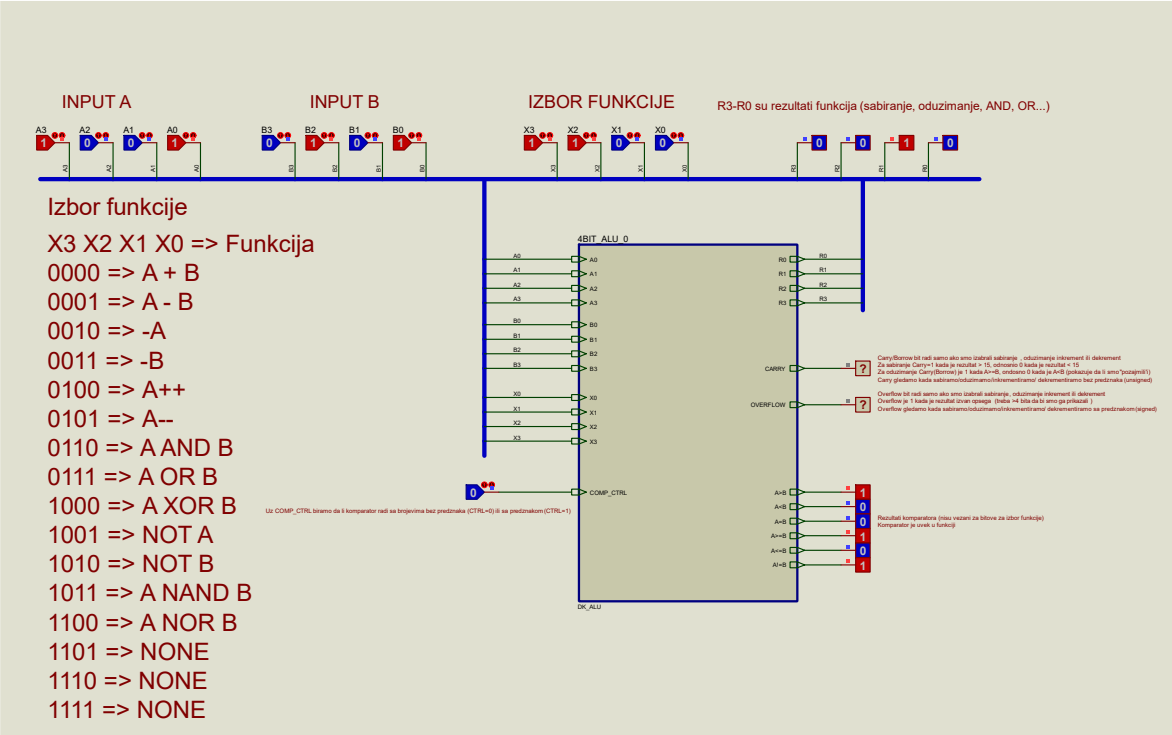


Slika 20. – A + B

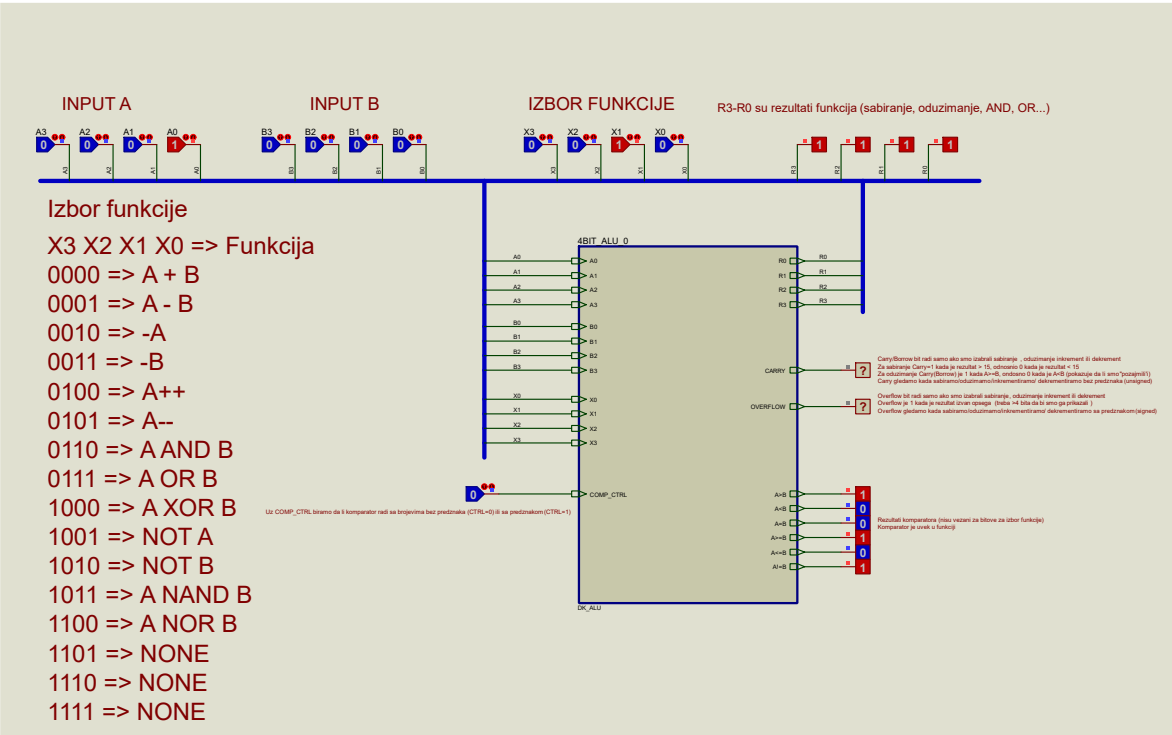


Slika 21. – A XOR B

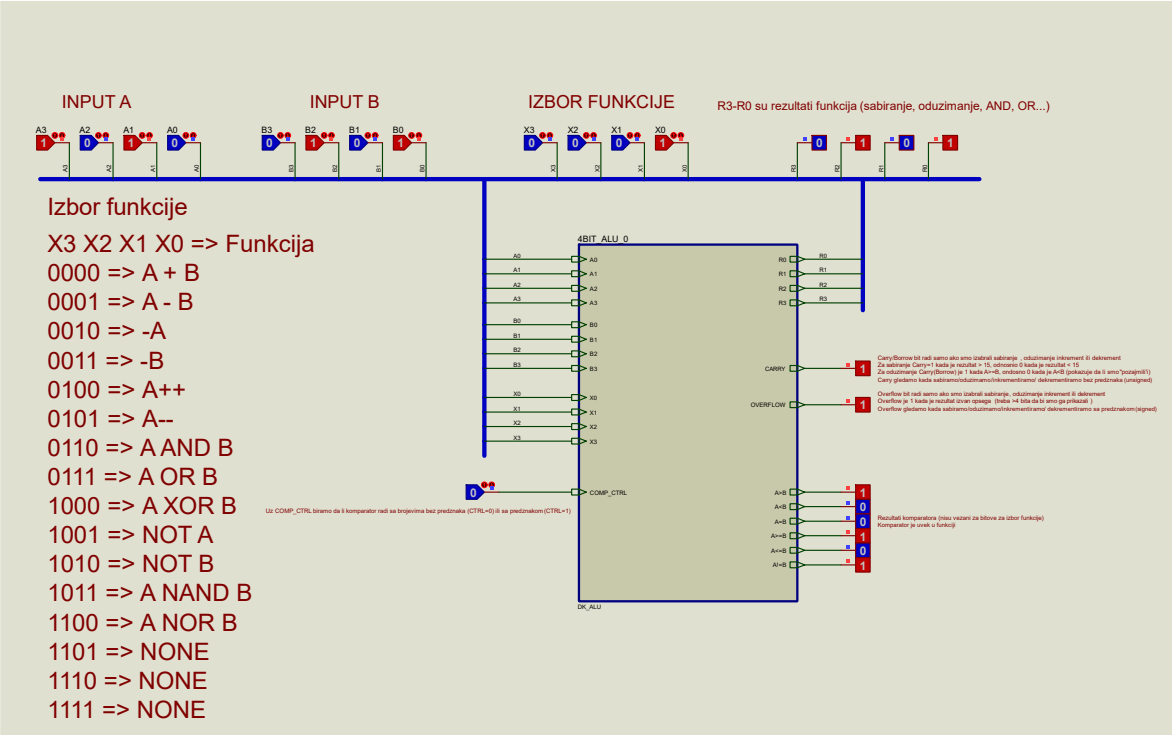
Subotica, 2024.



Slika 22. – A NOR B

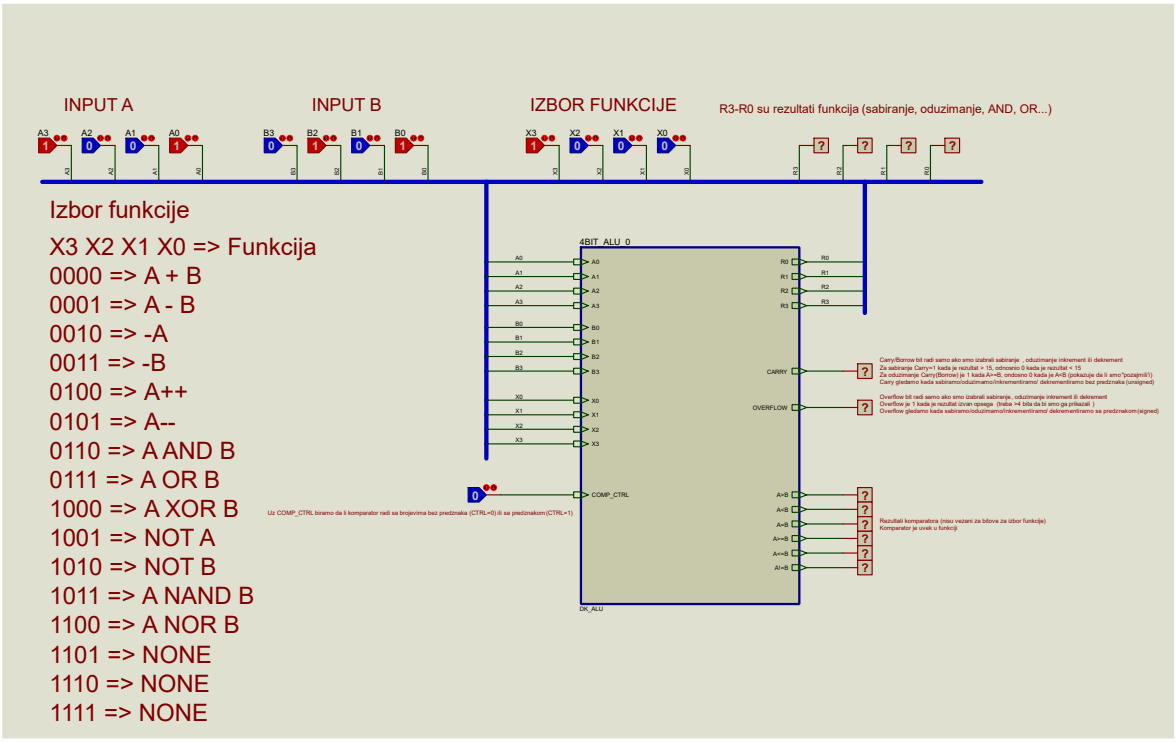


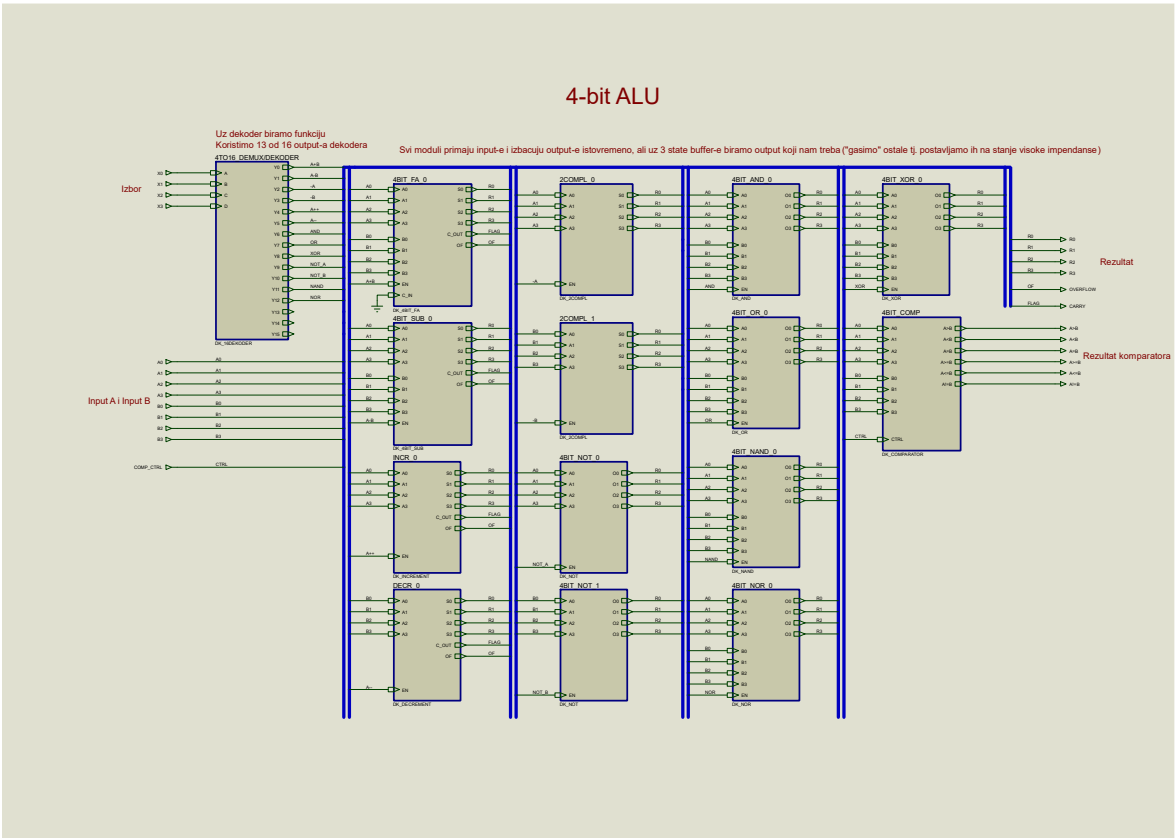
Slika 23. – -A


INPUT A
INPUT B
IZBOR FUNKCIJE
R3-R0 su rezultati funkcija (sabiranje, oduzimanje, AND, OR...)
Izbor funkcije
X3 X2 X1 X0 => Funkcija
0000 => A + B
0001 => A - B
0010 => -A
0011 => -B
0100 => A++
0101 => A--
0110 => A AND B
0111 => A OR B
1000 => A XOR B
1001 => NOT A
1010 => NOT B
1011 => A NAND B
1100 => A NOR B
1101 => NONE
1110 => NONE
1111 => NONE
4BIT_ALU_0
COMP_CTRL
CARRY
OVERFLOW
DK_ALU

Slika 24. – A - B

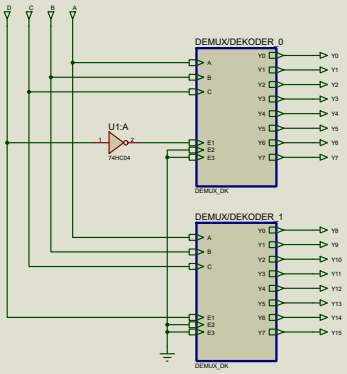
4. PRILOG

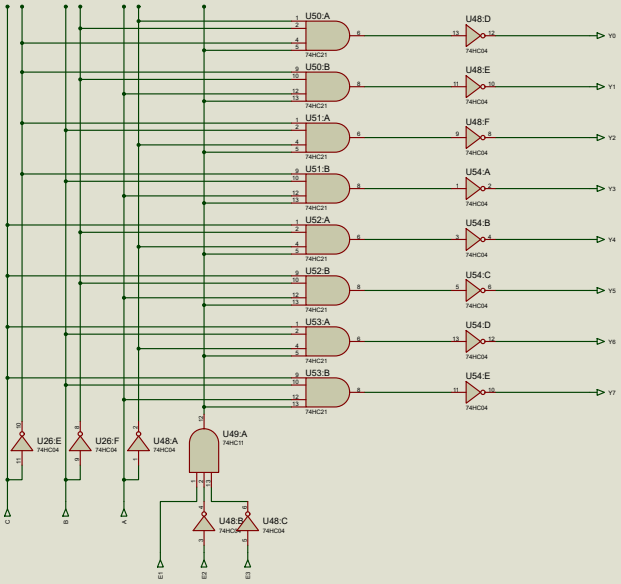




Subotica, 2024.

16-bit MUX/DECODER

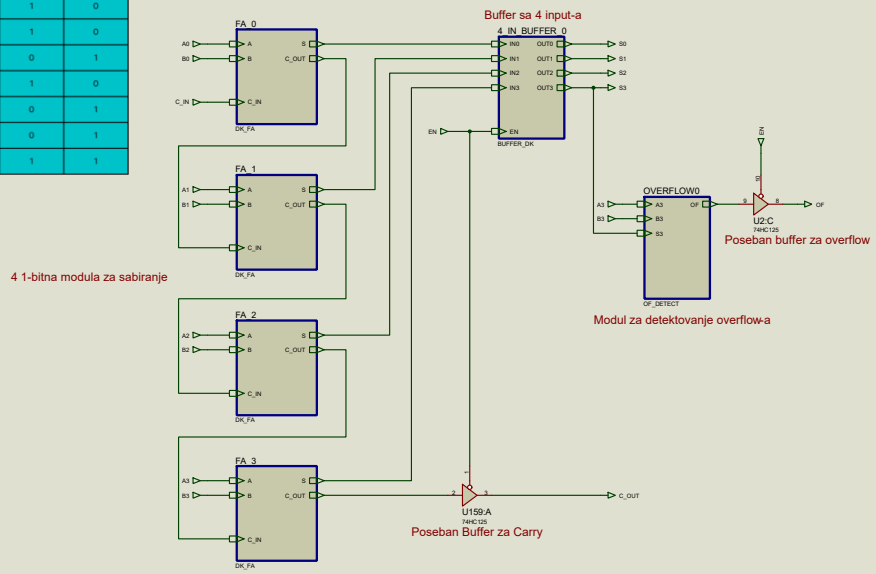




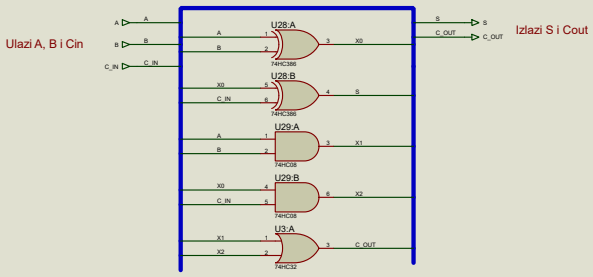
Subotica, 2024.

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

4-bit modul za sabiranje sa carry i overflow

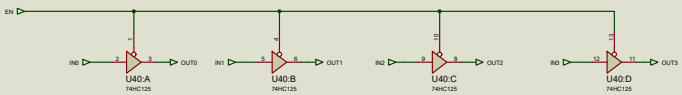


1-bit FA sa carry



Subotica, 2024.

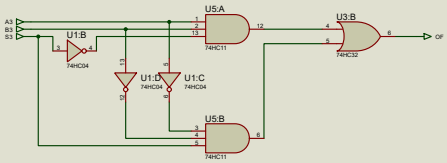
4 bit tri-state buffer sa negativnom logikom (EN=0-enabled EN=1-disabled)



Uzima 4-bitni ulaz i izbacuje ga nepromenjenog ako je EN =0 tj. u stanju visoke impedanse ako je EN=1

IN3	IN2	IN1	IN0	EN	OUT3	OUT2	OUT1	OUT0
X	X	X	X	1	hi-Z	hi-Z	hi-Z	hi-Z
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1
1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1
1	0	1	0	0	1	0	1	0
1	0	1	1	0	1	0	1	1
1	1	0	0	0	1	1	0	0
1	1	0	1	0	1	1	0	1
1	1	1	0	0	1	1	1	0
1	1	1	1	0	1	1	1	1

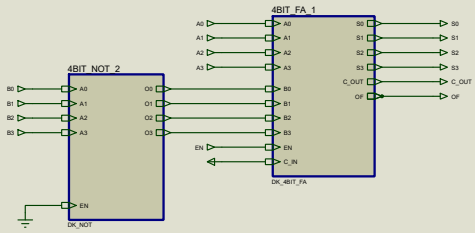
OVERFLOW CHECK



Proveravamo MSB-ove ulaza i rezultata
Ako znak rezultata nije isti kao znak input-a OF=1

A3	B3	S3	OF
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

4-bit modul za oduzimanje sa borrow (c_out) i overflow



A	B	S	Cout
0	0	0	1
0	1	0	0
1	0	1	1
1	1	0	1

Modul za oduzimanje smo napravili od modula za sabiranje

Prvi ulaz ostaje nepromenjen

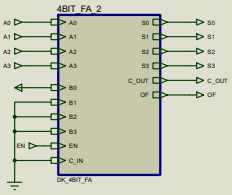
Drugi ulaz prebacimo u 1. komplement (4-bitni NOT) i uz Cin iz 1. u 2. komplement

Imamo A + (-B), koje je jednako sa A - B

Koristimo OF iz modula za sabiranje

Borrow (c_out) je 1 kada je A >= B tj. 0 kada je A < B

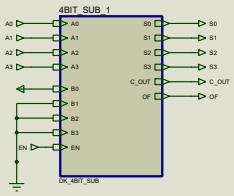
4-bit inkrement sa carry i overflow



Koristimo modul za sabiranje u koga ubacujemo broj koji inkrementiramo u A input i u Binput-u LSB postavlajmo na 1 a ostale na 0

A3	A2	A1	A0	R3	R2	R1	R0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

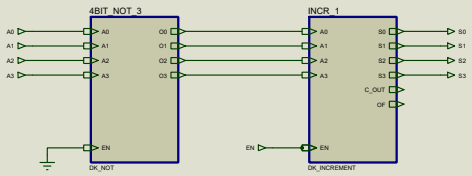
4-bit dekrement sa borrow i overflow



Koristimo modul za oduzimanje u koga ubacujemo broj koji dekrementiramo u A -input i u B-input-u LSB postavlajmo na 1 a ostale na 0

A3	A2	A1	A0	R3	R2	R1	R0
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

4-bit drugi komplement sa carry i overflow



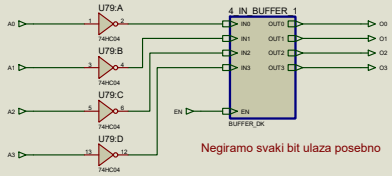
Prvo negiramo ulaz (1. komplement) Pa inkrementiramo 1. komplement i dobijamo 2. komplement

A3	A2	A1	A0	R3	R2	R1	R0
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

Za svaki bit posebno

A	R
0	1
1	0

4-bit NOT

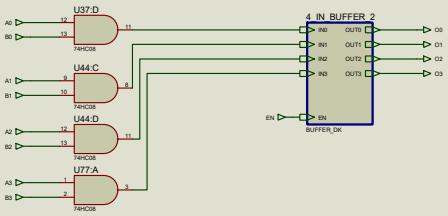


Negiramo svaki bit ulaza posebno

Za svaki bit posebno

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

4-bit AND

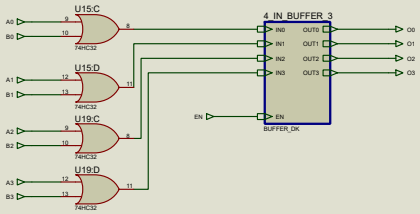


Radimo AND svakog bita A i B posebno

Za svaki bit posebno

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

4-bit OR

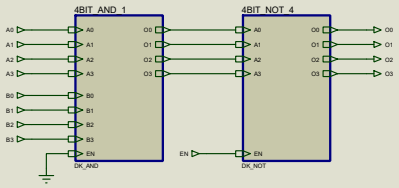


Radimo OR svakog bita A i B posebno

Za svaki bit posebno

A	B	R
0	0	1
0	1	1
1	0	1
1	1	0

4-bit NAND

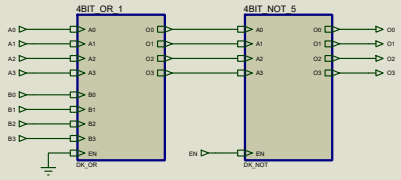


Negiramo izlaz AND modula

Za svaki bit posebno

A	B	R
0	0	1
0	1	0
1	0	0
1	1	0

4-bit NOR

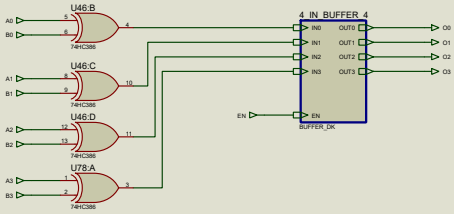


Negiramo izlaz OR modula

Za svaki bit posebno

A	B	R
0	0	0
0	1	1
1	0	1
1	1	0

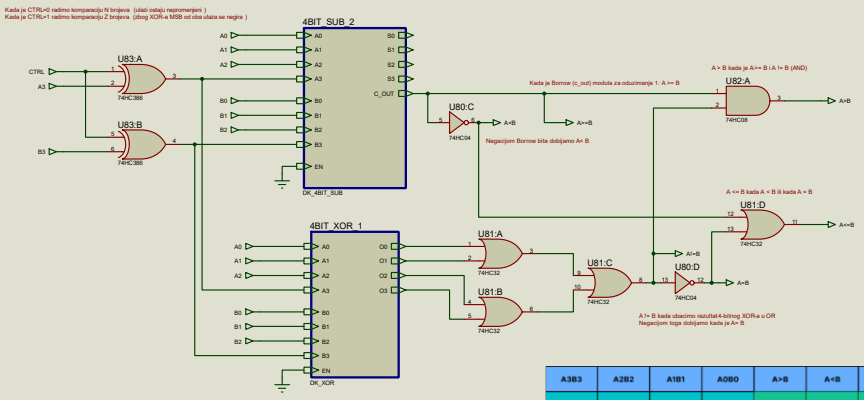
4-bit XOR



Radimo XOR svakog bita A i B posebno

4-bit komparator za N i Z brojeve

Za komparator koristimo modul za sabiranje i XOR modul



A	B	A>B	A<B	A=B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

A3B3	A2B2	A1B1	A0B0	A>B	A<B	A=B	A>=B	A<=B	A!=B
A3>B3	X	X	X	1	0	0	1	0	1
A3<B3	X	X	X	0	1	0	0	1	1
A3=B3	A2>B2	X	X	1	0	0	1	0	1
A3=B3	A2<B2	X	X	0	1	0	0	1	1
A3=B3	A2=B2	A1>B1	X	1	0	0	1	0	1
A3=B3	A2=B2	A1<B1	X	0	1	0	0	1	1
A3=B3	A2=B2	A1=B1	A0>B0	1	0	0	1	0	1
A3=B3	A2=B2	A1=B1	A0<B0	0	1	0	0	1	1
A3=B3	A2=B2	A1=B1	A0=B0	0	0	1	1	1	0

Subotica, 2024.

LITERATURA

[W1] Tablica istinitosti za komparator - https://technobyte.org/2-bit-4-bit-comparator/