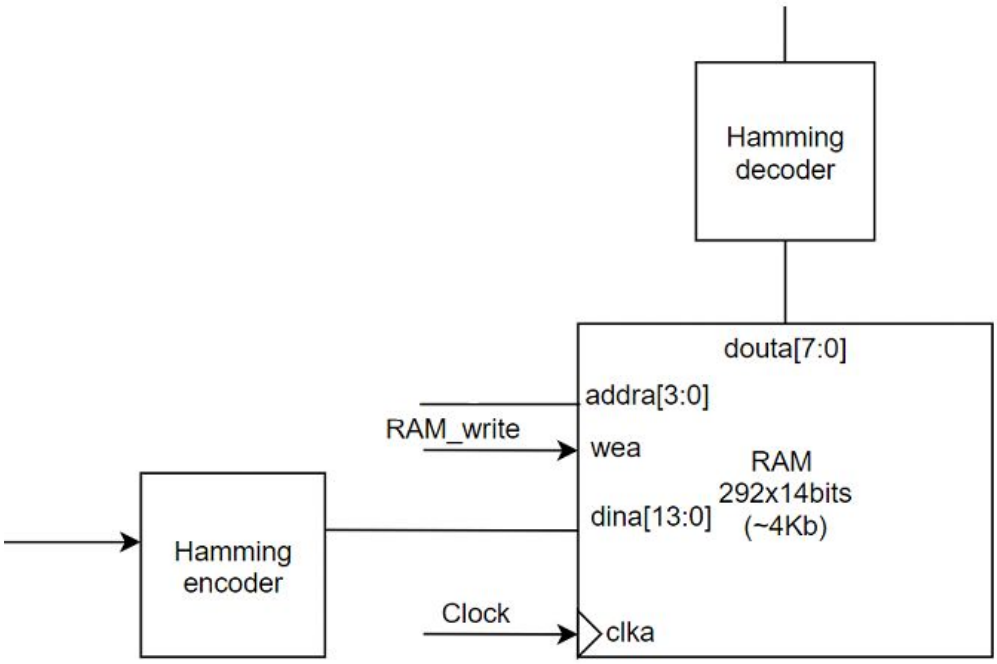
Projekat iz predmeta: Digitalni sistemi otporni na otkaz

Tema projekta:

Zaštita memorije u EC1 procesoru pomoću Hamingovog kodera i dekodera



Student: Aleksandar Komazec
Mentor: Rastislav Struharik

Dana _____ u Novom Sadu

Potpis studenta:_____

Potpis mentora:_____

Sadržaj:

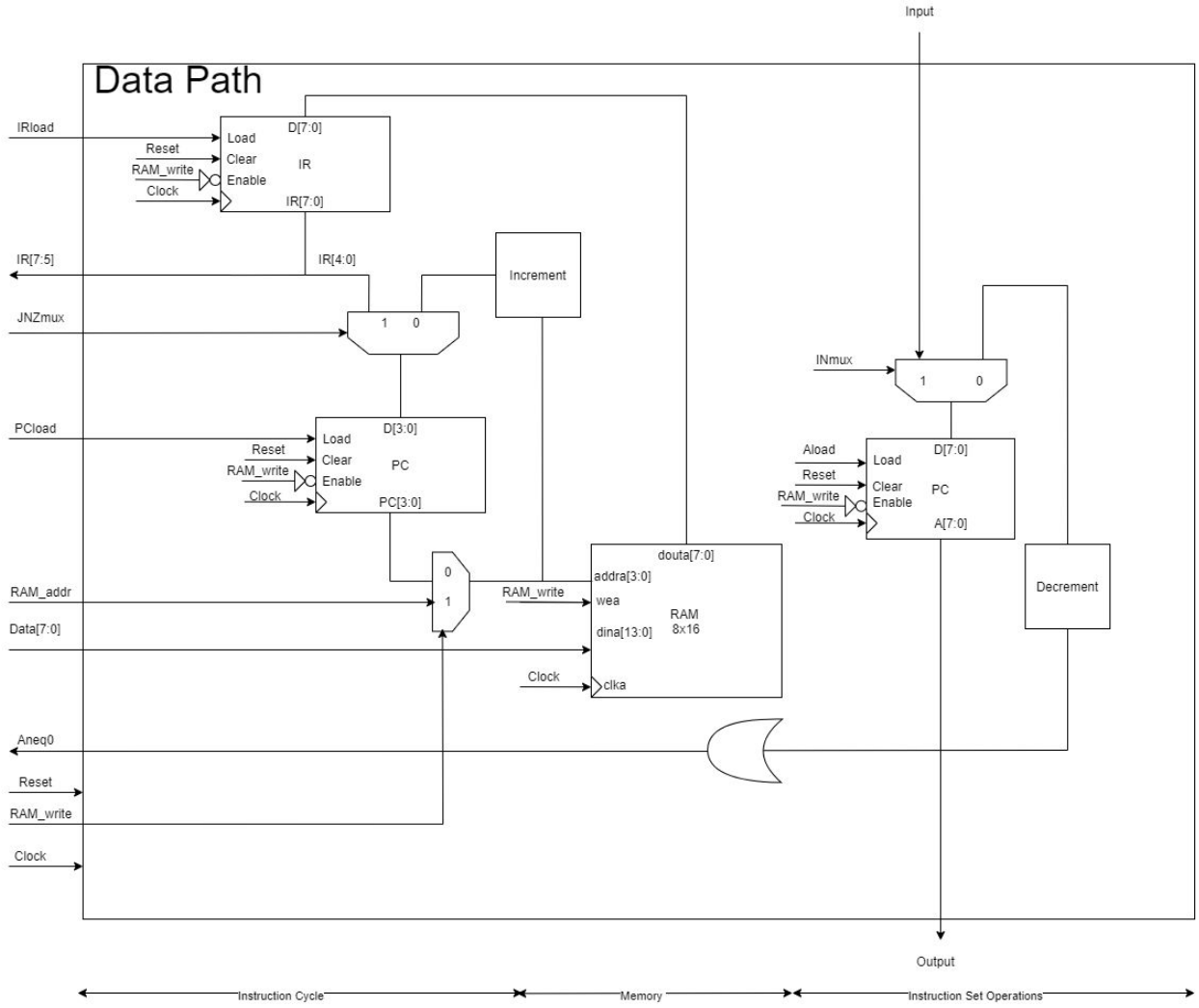
1.Uvod:

Na Easy computer 1 (ECU 1), jednostavanom osmobitnom procesoru je primenjeno kodovanje i dekodovanje podataka. Kako bi se ova tehnika primenila na najjefikasniji način, osmobitna magistrala je podeljena na dva dela po četiri bita I na taj način je predstavljen Hamingov kod (7,4).

1.1 Procesor “Easy computer one”

Easy computer one se sastoji iz dva glavna dela. Prvi deo obezbeđuje tokove podataka (eng. Datapath Slika 1).

1.1.1 Data Path



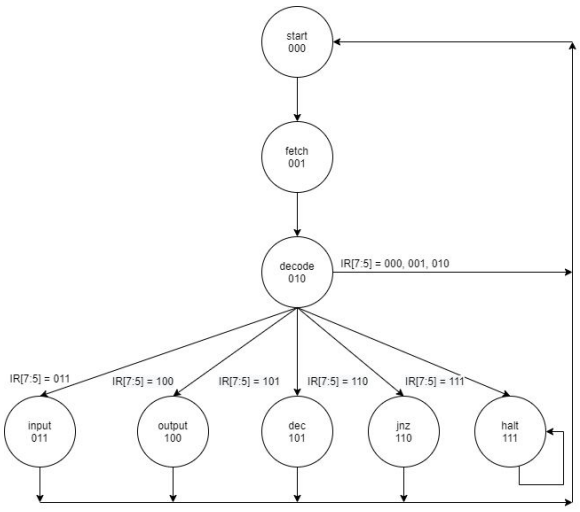
Slika 1: DataPath EC1 procesora

Data path se sastoji iz tri dela:

1)**Instruction Cycle Operations** (Krug instrukcija) deo koji obavlja pribavljanje instrukcije , inkrementiranje(uvećanje za 1) ili punjenje programskog brojača (PC).
Deo putanje podataka koji obavlja pribavljanje instrukcije i inkrementiranje ili punjenje PC-a sadrži Instrukcioni registar (IR) i PC. Veličina instrukcije u bitovima određuje veličinu IR-a, dok broj adresibilnih memorijskih lokacija određuje veličinu PC-a. Koristi se memorija sa 16 lokacija, svaka veličine 8-bita, pa je zbog toga potrebna 4-bitna adresa. To znači da je PC veličine 4 bita, a IR 8 bitova. 4-bitna inkrement jedinica se koristi za inkrementiranje PC-a za vrednost 1. PC je potrebno da se puni ili vrednošću koji je rezultat inkrement jedinice, ili adresom od instrukcije JUMP NO ZERO (JNZ). Za selekciju ove vrednosti se koristi multiplekser tipa 2 na 1.Jedan od ulaza u multiplekser je od jedinice za uvećanje, a drugi ulaz su četiri LSB bita registra IR, IR3-0. Radi jednostavnosti arhitekture EC1 ne koristi se spoljna memorija, već je ona sastavni deo procesora. Kapacitet memorije je 16 lokacija x 8 bitova i tipa je RAM. Izlaz PC-a je direktno povezan na 4-bitne memorijske adresne linije, s obzirom da se memorijska lokacija uvek određuje na osnovu sadržaja PC-a. 8-bitni izlaz memorije, Q7-0, se povezuje na ulaz registra IR radi izvršenja operacije zahvata instrukcija.
2) **Memory** - Memorija u koju su smeštene instrukcije koje se obavljaju. RAM memorija se popunjava tako što se RAM_Write podesi na 1. Tada ostatak sistema nije u funkciji, već instrukcije upisuju u RAM.
3)**Instruction Set Operatlons** (set instrukcija) deo koji obavlja operacije nad podacima za sve instrukcije iz skupa instrukcija.Sadrži akomulator koji predstavlja registar nad kojim se izvršavaju instrukcije.

1.1.2 Control Unit

Drugi deo se sastoji od registara i funkcionalnih jedinica i upravljačkog dela (eng. control unit Slika 2) koji obezbeđuje upravljanje radom mikroprocesora.



Slika 2: Control Unit

Dijagram stanja upravljačke jedinice je prikazan na slici 2. Prvo stanje Start, 000, se koristi kao inicijalno stanje. U toku ovog stanja ne obavlja se akcija. Pored toga, stanje Start poseduje jedan dodatni taktni ciklus za instrukcije kojima je potreban ekstra taktni interval da bi se operacija kompletirala. Nakon završetka svake instrukcije sledeće stanje je Start.Od pet instrukcija, samo JNZ(Jump no zero) instrukcija zahteva dodatni taktni interval da bi završila svoj rad. Razlog ovome je taj što PC mora da se napuni na novu vrednost adrese kada je uslov testa zadovoljen. Nova vrednost adrese, u suštini, se puni u PC na početku narednog taktnog intervala. Zbog toga, ako FSM mora da pređe u stanje **"Fetch"** sa narednim taktnim intervalom, tada će se IR puniti iz memorije sa stare adrese, a ne sa nove adrese. Za slučaj kada FSM prelazi nazad u stanje Start, PC će se ažurirati u ovom stanju na novu adresu, a memoriji će se pristupiti u toku narednog Fetch stanja sa nove adrese.Svako stanje je numerisano određenim bitma (Slika 2). Biti koji se koriste su pristigli iz Data Path-a.To su biti 7,6,5 koji se prosleđuju sa izlaza instrukcionog registra. **"Start"** predstavlja kao što je već pomenuto inicijalno stanje, dok **"Fetch"** predstavlja primanje podataka i **"Decode"** predstavlja stanje u kome se razmatra koje od 5 stanja (input,output,dec,jnz,halt) će biti posećeno.
Za kombinacije (000,001,010) se vraća u inicijalno stanje. Implementirano je 5 instrukcija: input, output, dec, jnz i halt (Tabela 1).

Tabela 1: Skup instrukcija

Instrukcija	Kodovanje	Operacija	Komentar
IN A	011 xxxxxxx	A<-Input	Na ulaz A se dovodi vrednost sa Input
OUT A	100 xxxxxxx	Output<- A	Čita se vrednost iz A (Akumulatora)
DEC A	101 xxxxxxx	A<-A-1	Umanje se vrednost A za jedan i ponovo šalje na ulaz A
JNZ address	110 xaaaa	if(A!=0) then PC=aaaa	Ako je vrednost A različita od nule tada PC dobija vrednost aaaa
HALT	111 xxxxx	Halt	Zasutavljanje

Predlog za nove mogućnosti:
S obzirom da vrednosti 000,001,010 neiskorištene, moguće je dizajn nadograditi novim instrukcijama tako da neiskorištene vrednosti predstavljaju nove instrukcije. Takođe četvrti bit na izlazu instrukcionog registra (IR), to jest IR(4) je ne iskorišten tako da je i taj bit moguće iskoristiti u budućim implementacijama instrukcija.

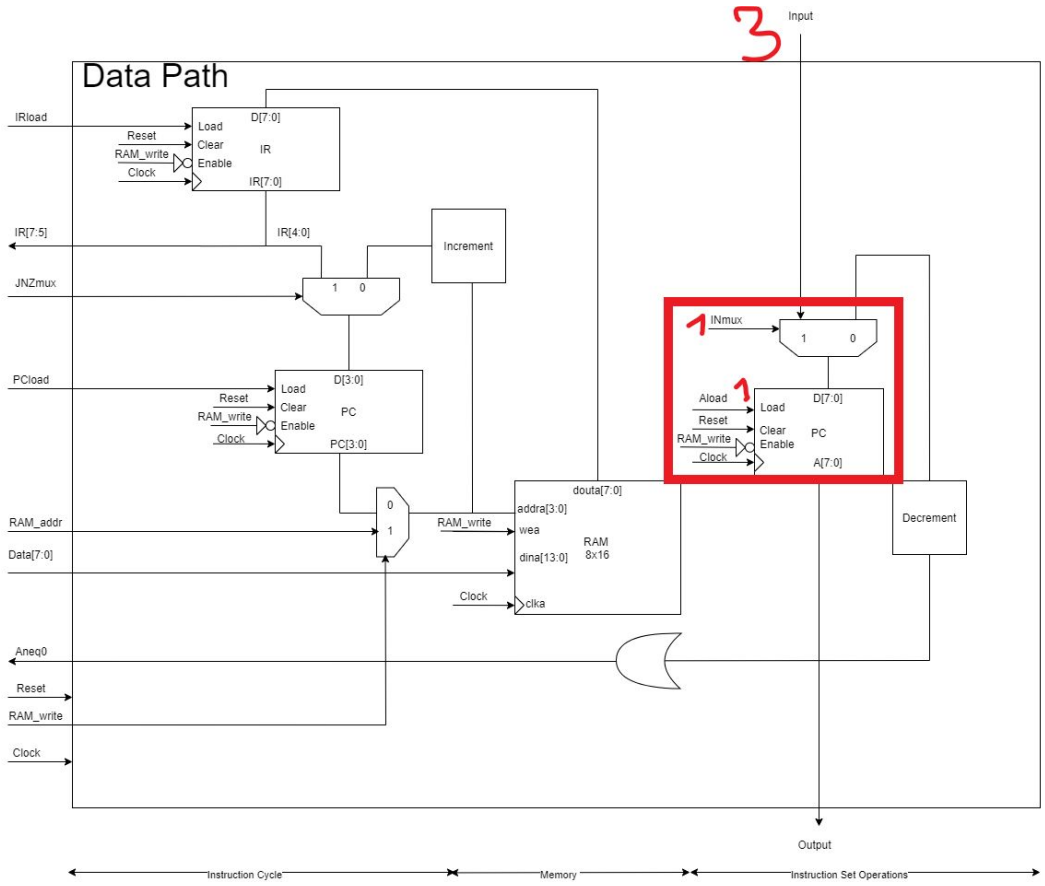
1.2 Primer rada EC1 procesora

Setovanjem RAM_Write na 1 omogućeno je da se instrukcije upišu u RAM.
Primer programa upisanog u RAM (Slika 3):

adresa	sadržaj	
0	01100000	-- IN A
1	10000000	-- OUT A
2	10100000	-- DEC A
3	11000001	-- JNZ loop (adresa labele loop je 1)
4	11111111	-- HALT

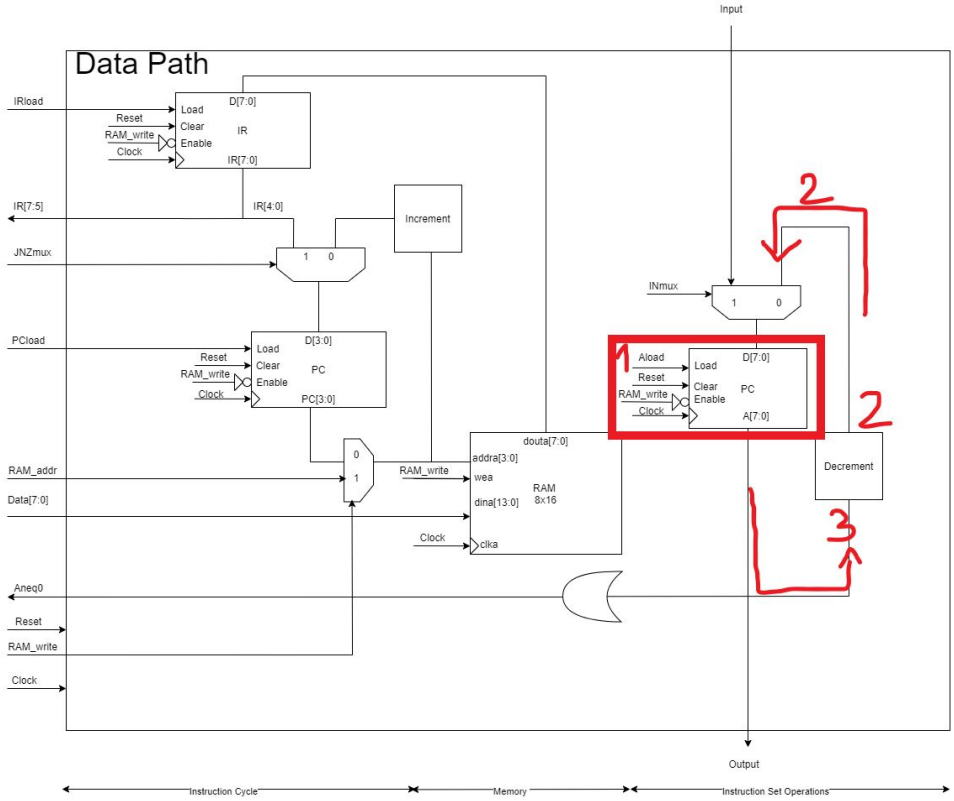
Slika 3: Program upisan u RAM

Izvršava se instrukcija na adresi 0,dakle IN A. Podatak koji je upisan na ulaz, Input je 3 (Korisnik opciono postavlja vrednost na ulaz). Dolazi do promene određenih komadnih signala, tačnije Control Unit postavlja vrednosti **INmux = 1, Aload = 1** (Slika 4)



Slika 4: Izvršavanje IN A instrukcije

Sledeća instrukcija je OUT A. Tako da će na izlazu biti pročitana vrednost 3.
Sledeća instrukcija je DEC A. Control Unit postavlja vrednosti **Aload = 0, INmux = 0** tako da se vrednost tri smanjuje za jedan i broj dva se upisuje u akomulator (Slika 5).



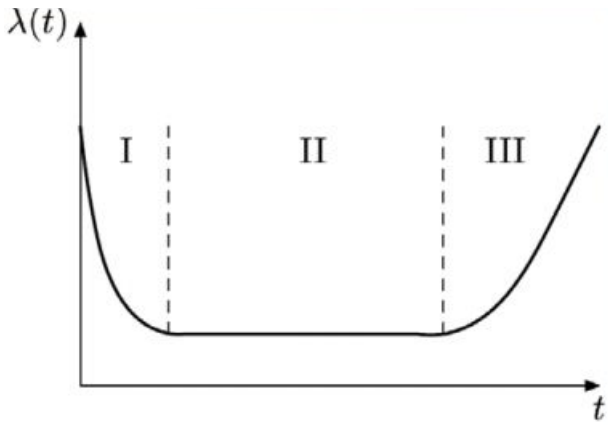
Slika 5: Izvršavanje DEC A instrukcije

Sledeća instrukcija je JNZ A, dakle ako A nije nula, a nije za sada (na izlazu je 02) kao što je gore objašnjeno. Vrednost brojača se vraća na adresu 1, to jeste instrukcija OUT A. Nakon JNZ A se ponovo izvršava OUT A, DEC A, pa JNZ A (Sada je vrednost 1 i ponovo se prelazi na adresu 1), ponovo se izvrši OUT A, DEC A, pa JNZ A (Vrednost je 0, tako da se neće skočiti na adresu 1 već će se preći na narednu adresu, adresa 4).
Poslednja instrukcija je HALT se izvršava. HALT predstavlja zaustavljanje celokupnog sistema.

Prednosti dizajna su

1. Jednostavnost
2. Cena
3. Pouzdanost (S obzirom na veličinu dizajna)

Glavna mana dizajna bi bila pouzdanost rada memorije na dužem vremenskom periodu.
Tačnije nakon određenog vremena dolazi do habanja komponenti (Slika 6, segment 3, lambda predstavlja Failure rate, to jest broj kvarova u jedinici vremena), tako da će u nekom treuntku vrlo verovatno doći do greške tokom upisa ili čitanja u memoriju. Drugi razlozi mogu biti nepogodna temperatura, zračnje i tako dalje.



Slika 6: Failure rate za hardverske sisteme

Tako da bi zaštita memorije donela veću pouzdanost, a na taj način i dužu upotrebnu vrednost, kao i novčanu uštedu. U zavisnosti od toga kakva memorija se koristi (U smislu kapaciteta, brzine rada i tako dalje) zavisi i ušteda. Potrebno je implementirati na zaštitu na neki način. Odgovor na taj problem u hardveru bi bila redudantnost. U sistemu gde postoji procesor, moguće je konstruisati sistem tako da postoje dva procesora tako da kada jedan prestane sa radom zbog kvara, drugi procesor može nastaviti da radi bez zaustavljanja sistema. Dok redudatnost kod softverskog sistema nema smisla zato što se softverski program uvek ponaša deterministički tako da bi se greška u jednoj verziji koda takođe pojavljivala i u ostalim kopijama koda.

1.2 Hamingovi kodovi

Hamingovi kodovi su prva klasa linearnih kodova osmišljeni za korekciju greške.
Našli su veliku primenu u kontroli greške u digitalnoj komunikaciji i sistemima za čuvanje podataka. Hamingovi kodovi mogu detektovati do 2 bita greške, to jeste korigovati jedan bit greške, za razliku od jednostavnog pariti kodovanja koje ne može korigovati grešku. Hamingovi kodovi su kodovi koji imaju najveci moguci rate za kodove određene dužine gde je minimum distance,d(c) = 3.

1.2.1 Opšti algoritam koda

Algoritam koji je ispod naveden generiše SEC (Single error correcting) kod
Algoritam po koracima:

1. Numerisanje bita počevši od 1: bit 1, 2, 3, 4, 5, 6, 7
2. Napisati brojeve u binarnom formatu: 1, 10, 11, 100, 101,...
3. Sve pozicije bita koje su stepen dvojke su biti parnosti 1,2,4,8 (1, 10, 100, 1000)
4. Sve druge pozicije bitova sa dve ili više bita čija je vrednost 1 su biti podataka.
5. Svaki bit podataka je ubačen u jedinstven set od odva ili više bita parnosti u odnosu na binarnu formu sopstvene pozicije:
    a. Bit parnosti 1 pokriva sve pozicije bita koji imaju LSB (least significant bit postavljen na 1) -> 3,5,7,9,... (011, 101, 111, 1001,...)
    b. Bit parnosti 2 pokriva sve pozicije bita koji kao drugi, to jeste sledeći bit u odnosu na LSB postavljen na 1 -> 3, 6, 7, 10, 11,... (011, 110, 111, 1010, 1011,...)
    c. Bit parnosti 4 pokriva sve pozicije bita koji kao treći, to jeste sledeći bit u odnosu na bit parnosti 2 ima vrednost postavljenu ja 1 -> 4–7, 12–15, 20–23.
    d. Bit parnosti 8 pokriva sve pozicije bita koji kao četrvi, to jeste sledeći bit u odnosu na bit parnosti 4 ima vrednost postavljenu ja 1 -> 8–15, 24–31, 40–47

Generalno svaki bit parnosti pokriva sve bite gde je bitwide AND operacija nad pozicijama parnosti i pozicija,a podataka različita od nula

1.2.2 Hamingova distanca

Hamingova distanca između dva binarna para predstavlja broj bita u kojima se razlikuju ti binarni parovi.

Na primer:
x = 0011
y = 0000

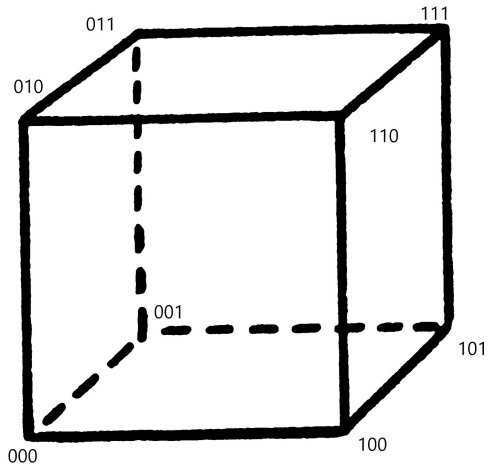
Hamingova distancia je δ(x, y) = 2 jer se su biti 0 i 1 od x jedinice dok su biti nula i jedan kod y jednaki nula.

Postoje tri aksioma koja moraju biti ispoštovana:

1. δ(x, y) = 0 ako i samo akox = y.
2. δ(x, y) = δ(y, x).
3. δ(x, y) + δ(y, z) ≥ δ(x, z).

Primer (Slika 7):

Podatak koji se šalje je tri bita i vrednost je 000.
Ako dođe greške u jednom bitu, recimo 010, moguća je detekcija greške na način da se predpostavi da je nula tačna vrednost i na taj način će se vrednost jedan korigovati na nula.



Slika 7: Trobitni prostor

Hamingov kod računa poziciju koja predstavlja bit na kome je došlo do greške računajući sindrom (Više o sindromu kasnije) tako da vrednost sindroma predstavlja poziciju na kojoj je došlo do greške. Kako je ta vrednost jednoznačna u slučaju da je Hamingova distanca veća od jedan, neće biti mogući detektovati grešku,a ni pravilno korigovati grešku.

1.2.3 Distanca koda (Code Distance(C))

Predstavlja minimalnu Hamingovu distancu između bilo koja dva različita para kodnih reči.

Primer:
Ispravna kodna reč je 000. Ako postoji mogućnost detektovanja i korigovanja jednog bita greške, to znači da su za ispravni kod, moguće greške 001, 010, 001, samim tim gledajući skup svih četiri vrednosti dolazi se do zaključka da je code distance jednak 1.

Primer za neki pozitivan broj m ≥ 3 postoji Hamingov kod sa određenim parametrima (Tabela 2)

Tabela 2: Parametri Hamingovog koda

Dužina koda	$n = 2^m - 1$
Broj informacionih simbola	$k = 2^m - m - 1$
Broj parity-check simbola	$m = n - k$
Sposobnost korigovanja greške	$t = 1\ (d_{min} = 3)$

Da bi se kod bio Hamingov kod, potrebno je ispoštovati sledeće pravilo:
Binarni linearni kod se naziva Hamingov kod za m ≥ 3 tako da parity-check matrica H ima m redova i 2m – 1 kolona tako da je svaka kolona jedinstvena I različita od 0.
Sa povećanjem bita podataka, povećava se i broj bita parnosti (Tabela 3).

Tabela 3: Neki od standardnih Hamingovih kodova

Parity bits(r)	Total bits (n)	Data bits (k)	Name	Rate
3	7	4	Hamming(7,4)	4/7 = 0.571
4	15	11	Hamming(15,11)	11/15 = 0.733
5	31	26	Hamming(31,26)	26/31 = 0.839

2. Konstrukcija kodera i dekodera

2.1 Koderi i dekoderi uopšteno

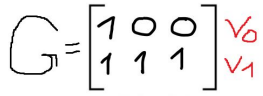
Uopšteno, kodovanje predstavlja proces računanja kodne reči za određene bite podataka. Drugim rečima svaki koder sadrži određeni mehanizam to jeste logiku koja se pridružuje pristiglim bitima podataka. Dodatni biti koji su pridruženi primljenim podacima se nazivaju check biti. Tako da se dobijena kodna reč sastoji od bita podataka i od check bita. Dekodovanje predstavlja proces vraćanja, to jest " izvlačenja" bita podataka iz kodne reči koja je rezultat kodovanja.

Primer koda:

(8, 7) Parity Check kod koji se sastoji od sedam bita podataka i jednim check bitom, to jeste bitom provere. Recimo da je A u ASCII kodu 1000001, reč nakon kodovanja bi bila 10000010. Dakle primenom određenog mehanizma dobio se bit provere i njegova vrednost je 0. Ako nije došlo do greške, dekodovana reč bi bila 1000001, to jeste ista kao reč koja se nalazi na ulazi kodera. Važna stvar jeste pojam information rate koji predstavlja odnos broja bita podataka i ukupnog broja bita u kodnoj reči. U ovom slučaju je to 7/8. Hamingov koder i dekoder je moguće konstruisati na osnovu parity check i generatorske matrice.

2.2 Koderi i dekoderi u slučaju linearnih kodova

Pomenute kodne reči predstavljaju proizvod reči podataka i generatorske matrice:
c = d*G, gde generatorska matrica (Slika 8a) predstavlja skup basis vektora (v0, v1) koji su poređani po redovima.

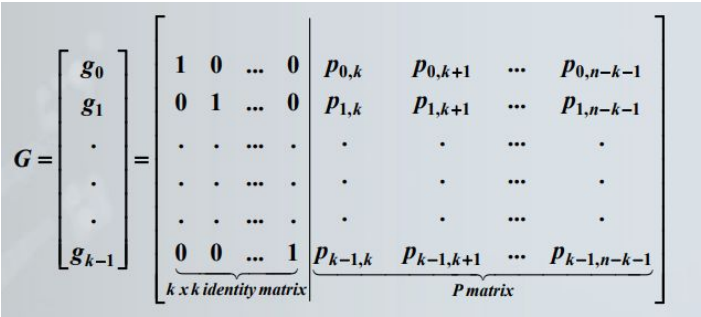


Slika 8a: Generatorska matrica

Takođe pri dekodovanju je potreno izračunati sindrom, $s = H \cdot c^T$ gde H predstavlja parity check matricu.

2.2.1 Implementacija kodera u slučaju linearnih kodova

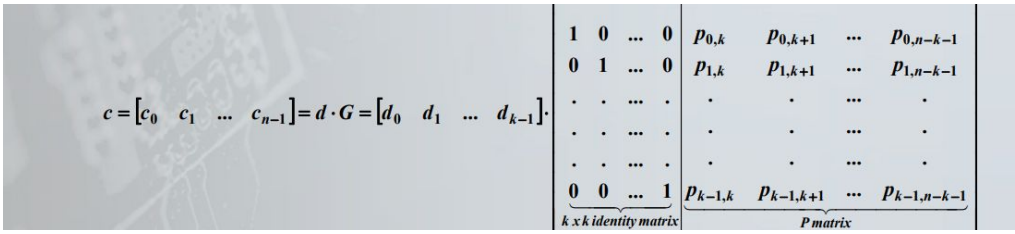
Matrica G je generatorska matrica i njen opšti oblik je predstavljen na slici 8b.



Slika 8b: Generatorska matrica u opštem slučaju linearnih kodova

pi,j = 0 ili 1, d = (d0, d1, …, d_{k-1}) poruka koja će biti kodovana,

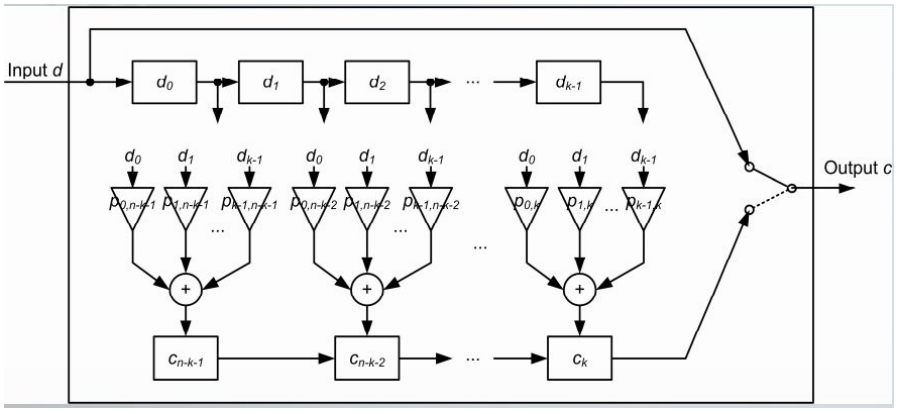
Tako da kodna reč dobijena nakon kodovanja je prikazana na slici 9.



Slika 9: Kodna reč u opštem slučaju linearnih kodova

Tako da važi:
$c_i = d_i$, for $0 \le i < k$, $c_j = d_0p_{0,j} + d_1p_{1,j} + \dots + d_{k-1}p_{k-1,j}$, za $k \le j < n-k-1$

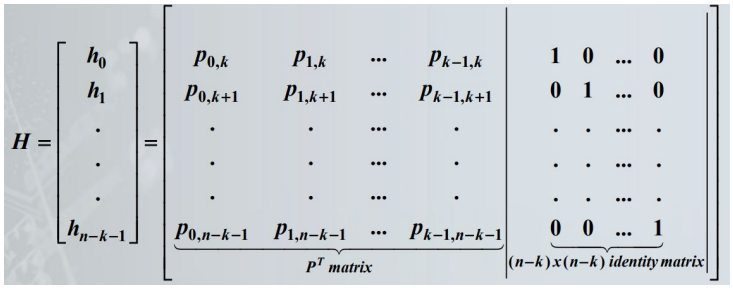
Tako da se opštne forme kodne reči c, može konstruisati koder u opštoj formi kao na slici 10.



Slika 10: Koder u opštem slučaju linearnih kodova

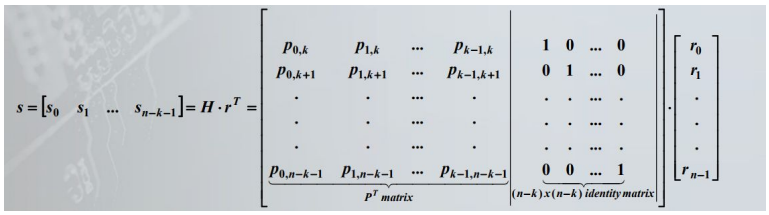
2.2.2 Implementacija dekodera u slučaju linearnih kodova

Gore pomenuta generatorska matrica ima formu G = [I $_k$A], tako da parity check matrica H ima formu H = [A_TI $_{n-k}$] i predstavljena je u opštem obliku (Slika 11).



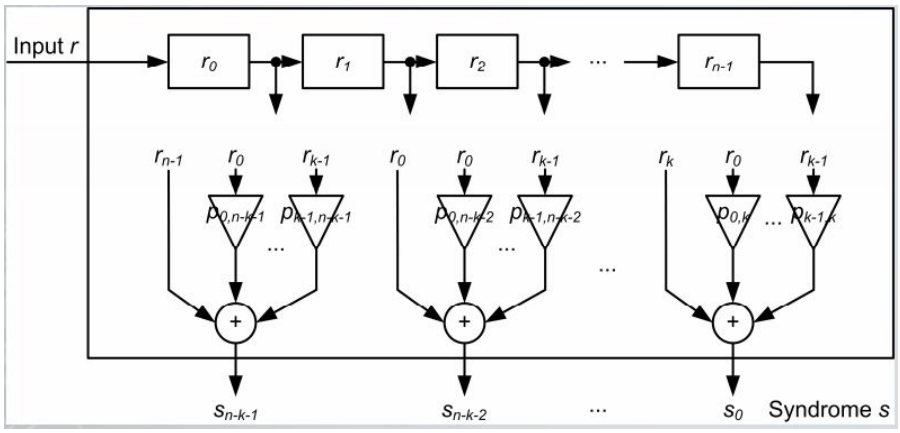
Slika 11: Parity check matrica u opštem slučaju linearnih kodova

Tako da se sindrom u opštem slučaju može izračunati (Slika 12).



Slika 12: Sindrom u opštem slučaju linearnih kodova

Na osnovu sindroma se može znati da li je došlo do greške ili nije. Ako je sindrom nula to znači da nije došlo do greše,a ako je različit od nule znači da je na nekom od bita došlo do greške. Dekoder u opštem obliku je prikazan na slici 13.



Slika 13: Dekoder u opštem slučaju linearnih kodova

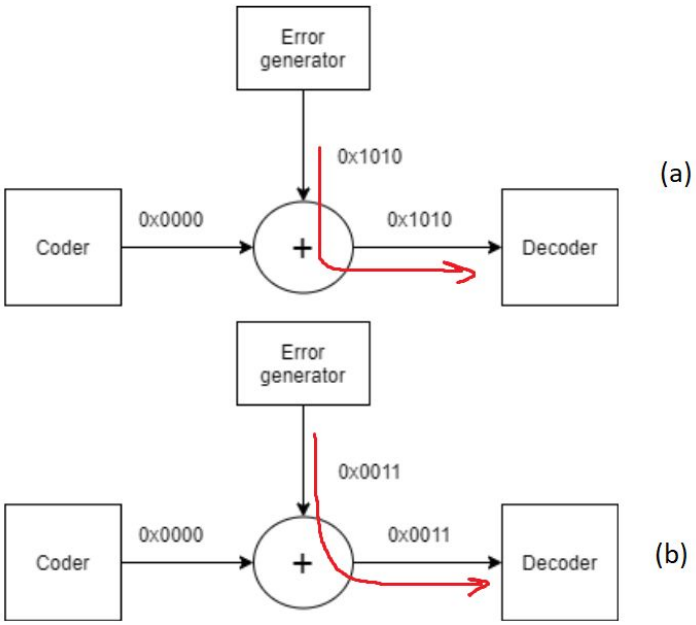
2.3 Odabir Hamingovog koda

S obzirom da je EC1 osmobitni procesor, tako da će reč koja se sastoji od bita podataka biti veličine 8 bita. Potrebno je pronaći adekvatan Hamingov kod koji će biti najefikasnije upotrebljen.

Prema tabeli 3, Hamming(7,4) ima 7 ukupno sedam bita, dok su 3 bita, biti provere.
Hamming(15,11) ima ukupno 15 bita, to jest 4 bita provere. Hamming(15,11) bi u potpunosti mogao da predstavi osmobitni podatak, tako da bi od 11 bita podataka tri najviša bita bila neiskorištena. Dok Hamming(7,4) koji ima svega četiri bita podataka nije dovoljan za predstavljane reči od osam bita podatka.

Izabrano rešenje:

Moguće je osmobitnu reč razdvojiti u dve nible (gornju i donju), gde bi gornja nibla predstavljala gornja četiri bita, dok bi donja nibla predstavljala donja četiri bita. Na taj način postoji mogućnost da se na ta dva para bita primeni Hamming(7,4), tako da će se kao krajnji rezultat dobiti reč od 14 bita gde su 6 bita biti parnosti (d7 d6 d5 d4 p2 p1 p0 d3 d2 d1 d0 p2 p1 p0). Mana ovog rešenja u odnosu na Hamming(15,11) je veći broj bita parnosti, dok je prednost Hamming(7,4) u odnosu na Hamming(15,11) je što je moguće detektovati i korigovati dva bita greške ali tako da se te greške pojavljuju u različitim niblama (Slia 14 (a)). U slučaju da se dva ili više bita greške dogode u istoj nibli, tada neće biti moguće detektovati i korigovati te greške (Slia 14 (b)).



Slika 14: Generisanje i propagacija greške a) 1 bit greške po nibli b) dva bita greške u prvoj nibli

2.3.1 Konstrukcija Hamingovog kodera(7,4)

Potrebno je izračunati parity check matricu $H = [P^T I_{n-k}]$

$$H = \left|\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right|$$

Parity check matrica u ovom obilku predstavlja separabilni oblik i na taj način omogućava da se generatorska matrica G jednostavno odredi.
Generatorska matrica G ima oblik $G = [I_k P]$

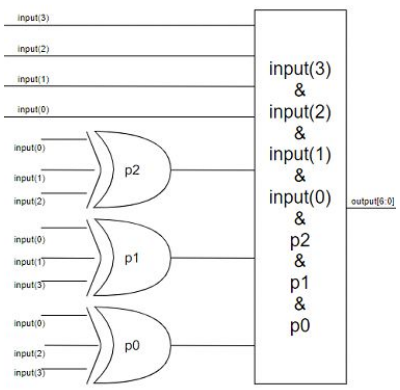
$$G = \left|\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right|$$

Iz generatorske matrice je moguće odrediti bite parnosti. Leva strana generatorske matrice predstavlja matricu identiteta. Kada se bit najviše važnosti u jediničnoj matrici podigne na 1 (Prva vrsta), tada se nulti i prvi bit podignu na jedan (Nulti, prvi i drugi bit predstavljaju bite parnosti) što znači da bit parnosti 0 i 1 pokrivaju šesti bit G matrice, to jeste treći bit u matrici identiteta (Biti u jediničnoj matrici su obeleženi sa 0, 1, 2 ,3, sa leva na desno). Kada se drugi bit matrice identiteta podigne, tada je drugi i nulti bit parnosti 1, što znači da drugi i nulti bit pokriva drugi bit matrice identiteta i tako dalje.

Tako da su biti parnosti:

p0 = input(0) xor input(2) xor input(3)
p1 = input(0) xor input(1) xor input(3)
p2 = input(0) xor input(1) xor input(2)

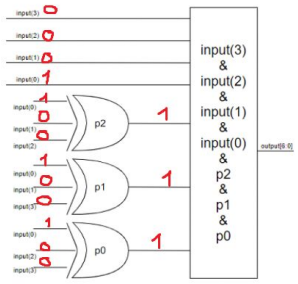
Tako da se na ulaz Hamingovog kodera dovode biti podataka spojeni sa bitima parnosti (Slika 15a).



Slika 15a: Hamingov koder(7,4)

Primer računanja kodne reči:

Na ulaz se dovede d = 0001, tako da se na osnovu dobijene logike za bite parnosti mogu izračunati njihove vrednosti (Slika 15b).



Slika 15b: Hamingov koder(7,4) primer za d = 0111

Tako da kodna reč c ima vrednost 0001111, gde su biti 0, 1 i 2 biti parnosti. Na osnovu konstruisanog Hamingovog kodera mogu se izračunati biti parnosti za sve moguće kombinacije ulaza (Tabela 4)

Tabela 4: Biti parnosti za sve kombinacije bita podataka

Bit(dec)	d3	d2	d1	d0	p2	p1	p0
0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1
2	0	0	1	0	1	1	0
3	0	0	1	1	0	0	1
4	0	1	0	0	1	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	0	1	1
7	0	1	1	1	1	0	0
8	1	0	0	0	0	1	1
9	1	0	0	1	1	0	0
10	1	0	1	0	1	0	1
11	1	0	1	1	0	1	0
12	1	1	0	0	1	1	0
13	1	1	0	1	0	0	1
14	1	1	1	0	0	0	0
15	1	1	1	1	1	1	1

To je moguće dokazati računom kao na slici 16. Recimo za primer koji je pomenut iznad, to jeste za d = 0001. Prva vrsta G matrice se množi sa prvim elementom d matrice, to je nula, druga vrsta G matrice se množi sa drugim elementom d matrice, to je takođe nula. Isto je i za treću vrstu. Dok kada se množi četvrta vrsta, dobije se da je kodna reč, c1 = 0 0 0 1 1 1 1

$$C_0 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}0000\end{bmatrix} = 0000000 \qquad C_8 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}1000\end{bmatrix} = 1000011$$

$$C_1 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}0001\end{bmatrix} = 0001111 \qquad C_9 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}1001\end{bmatrix} = 1001100$$

$$C_2 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}0010\end{bmatrix} = 0010110 \qquad C_{10} = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}1010\end{bmatrix} = 1010101$$

$$C_3 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}0011\end{bmatrix} = 0011001 \qquad C_{11} = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}1011\end{bmatrix} = 1011010$$

$$C_4 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}0100\end{bmatrix} = 0100101 \qquad C_{12} = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}1100\end{bmatrix} = 1100110$$

$$C_5 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}0101\end{bmatrix} = 0101010 \qquad C_{13} = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}1101\end{bmatrix} = 1101001$$

$$C_6 = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}0110\end{bmatrix} = 0110011 \qquad C_{14} = \begin{bmatrix}1000011\\0100101\\0010110\\0001111\end{bmatrix}\begin{bmatrix}1110\end{bmatrix} = 1110000$$

Slika 16: Računanje kodne reči za sve kombinacije podataka d

2.3.1 Konstrukcija Hamingovog dekodera(7,4)

Hamingov dekoder se može konstruisati koristeći ne separabilnu formu H matrice.

$$H = \begin{vmatrix} 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{vmatrix}$$

Potrebno je izračunati sindrom. S obzirom da je H matrica postavljena tako da bude poređena po vrednosti od 1 do 7 (Kolone, sa leva na desno) na taj način vrednost sindroma predstavlja poziciju bita gde se desila greška. Ako je vrednost sindroma 001, to znači da je prvi bit pogrešan, dok vrednost sindroma 000 znači da ne postoji greška u primljenoj reči.

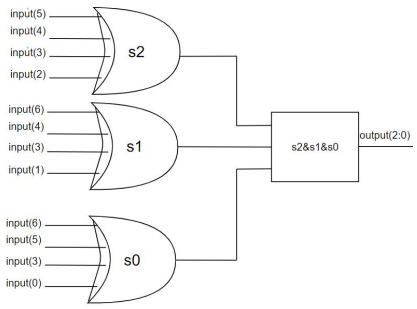
Primer:
Kodna reč c je 1 1 0 1 0 1 0 koja je nastala kodovajne podatka 0 1 0 1. Tokom transimsije se dogodi greška tako da je primljena reč je 0 1 0 1 0 1 0.
Sindrom se računa na sledeći način:

$$s = H * r^T = \begin{vmatrix} 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{vmatrix} * \begin{vmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{vmatrix} = [1\ 1\ 1]$$

Što znači da se greška desila na sedmom bitu, što je tačno.

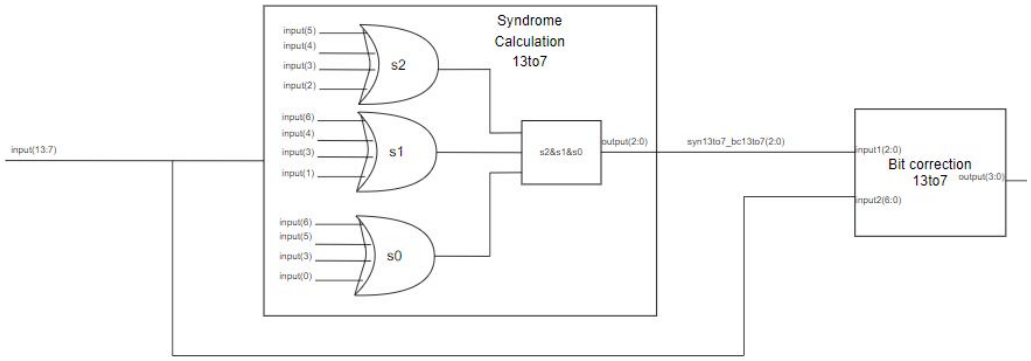
Gledajući H matricu u ne separabilnoj formi, moguće je predstaviti sindrom koristeći ekskluzivno ili operaciju (Slika 17)

s2 = input(5) xor input(4) xor input(3) xor input(2)
s1 = input(6) xor input(4) xor input(3) xor input(1)
s0 = input(6) xor input(5) xor input(3) xor input(0)



Slika 17: Impelemntacija sindroma za Hamming (7,4)

Izračunati sindrom se prosleđuje na blok Bit correction (Slika 18) koji invertuje bit greške (U slučaju da je sindrom različit od nule). U primeru iznad, sindrom je 7 što znači da je bit correction blok preuzeti reč sa ulaza i invertovati sedmi bit (Sedmi bit je prvi put invertovan i to se smatra greškom, sada kada je pogrešan bit pronađen, on se invertuje još jednom i na taj način se uklanja greška). Izlaz bit correction bloka je četvorobitna, to jeste biti podataka. Biti parnosti su odbačeni, tako da se samo biti podataka dalje šalju na magistralu.



Slika 18: Implementacija dekodera za Hamming (7,4)

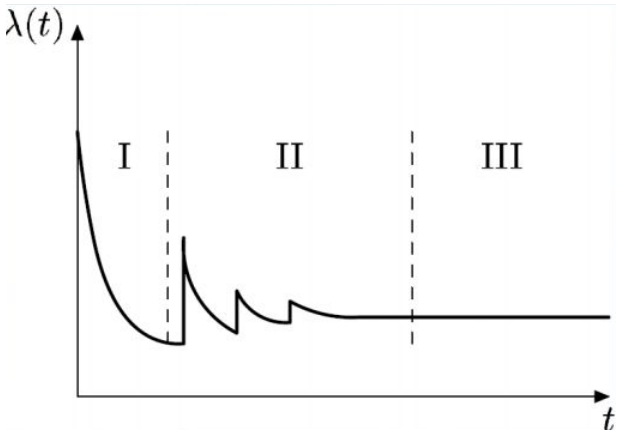
Tako da su parametri za odabrani Hamingov kod 7,4 prikazani u tabeli 5.

Tabela 5: Parametri za Hamming (7,4)

Dužina koda	7
Broj informacionih simbola	4
Broj parity-check simbola	3
Sposobnost korigovanja greške	1

3. Testiranje i Integracija Hamingovog kodera i dekodera

Nakon implementacije Hamingovog kodera i dekodera potrebno je uraditi testiranje istih. Jer ako bi se integracija komponenti uradila, a da pre toga nije urađeno temeljno testiranje, postoji šansa da se uvedu nove greške, to jeste novi problemi u sistem.
To je karakteristično kod razvoja softvera (Slika 19).

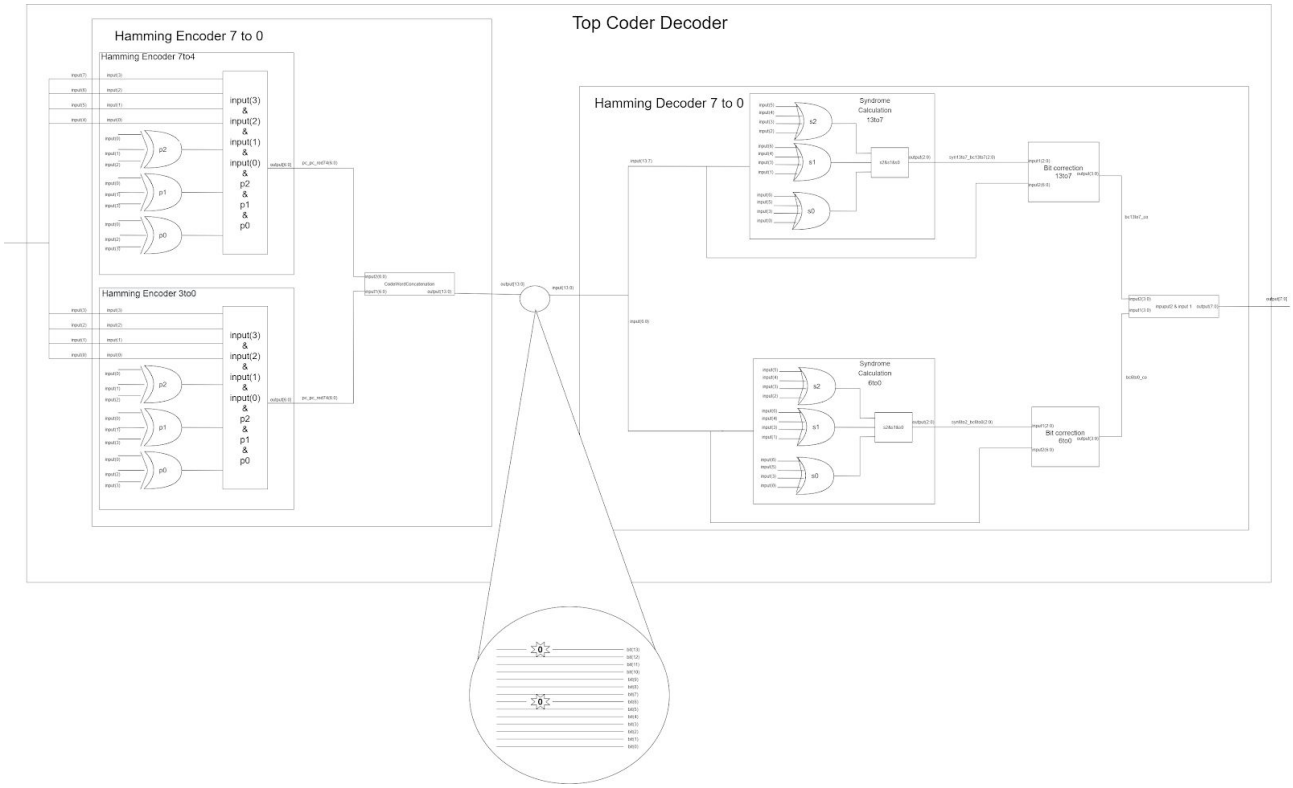


Slika 19: Failure rate za softverske sisteme

Softverski failure rate u prvoj sekciji predstavlja period kada se inicijalno implementiran softverski sistem debaguje i testira i na taj način se smanje greške. Druga sekcija se naziva “useful life”, u tom vremenskom intervalu se uvode nove mogućnosti i samim tim uvode se nove greške, zatim se te greške ispravljaju takozvanim zakprama (eng. “patch”) što predstavlja “besplatan” to jeste jeftin način da se otklone problemi. Što se tiče hardvera, tu nije moguće ispraviti greške na isti način. Ako korisnici dobiju hardver koji ima određene defekte, to umanjuje ugled kompanije, korisnici neće kupovati tu seriju tog proizvoda, taj model i slično. Što samim tim dovodi do ogromnih troškova. Takođe bi takva situacija uslovila kompanije da naprave novu verziju hardvera što dodatne resurse, kao i ostale stvari koje predstavljaju velike troškove.
Dakle temeljno testiranje hardvera je od velikog značaja.

3.1 Testiranje Hamingovog kodera i dekodera

Koder i dekoder su spojeni u jedan zajednički sistem (Slika 20).



Slika 20: Stuck at 0 između Hamingovog kodera i dekodera

U realnom svetu greška može da nastane na mnogo načina i u odnosu na to koliko traje, to jeste kako se manifestuje može da se uvrstiti u određenu kategoriju:

1. Moguće je da se greška ponavlja u određenom periodu vremena. Recimo greška se desi jednom u sat vremena zato što neki drugi element u sistemu ponaša neočekivano na kratak vremenski period. Recimo da se napon na napajanju iz nepoznatog razloga na trenutak spusti na vrednost ispod praga i taj signal se registruje i prouzrokuje vrednost 0 na bitu 13 i bitu 6. Ovo je dobar primer gde se može videti da integracija Hamingovog kodera i dekodera povećava pouzdanost sistema.
2. Moguće je da greška bude prisutna svo vreme. Recimo da usled habanja komponenti dođe do defekta (Dođe do otvorene veze vodova). Na Slici xx, su vodovi 13 i 6 otvoreni usled habanja.

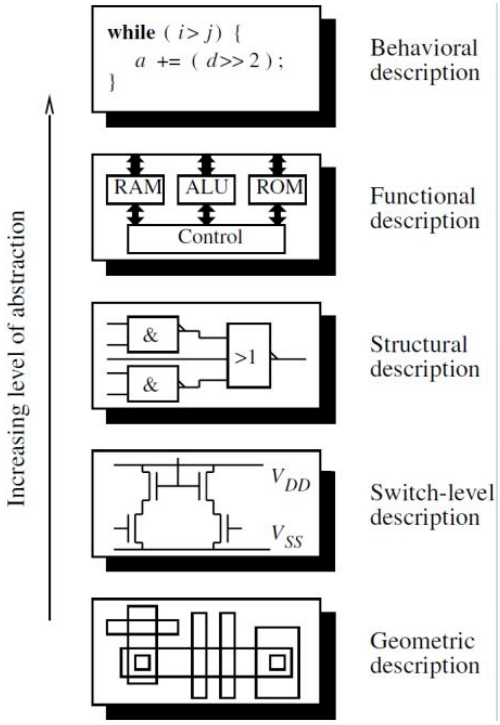
Popravljanje, to jeste menjanje hardvera zavisi od određenih faktora. Vreme potrebno da se sistem popravi se naziva MTTR (Mean Time to Repair) i predstavlja se formulom:

$$MTTR = \frac{1}{\mu}$$

Gde je repair rate μ predviđen broj popravki u jedinici vremena.
MTTR zavisi od fault recovery mehanizma korišćen u sistemu, lokacija sistema, lokacija rezervnih delova, zakazanog održavanja i tako dalje.
Nizak MTTR zahteva high operational cost sistema, na primer:

1. MTTR = 30 minuta - pod uslovom da se taj sistem nadgleda 24 sata, da je operativni radnik konstanto na lokaciji gde je sistem i da su rezervni delovi na istoj lokaci gde i sistem. To bi primer nekog sistema za koji je od velike važnosti da uvek bude dostupan.
2. MTTR = 2 nedelje - pod uslovom da se sistem nadgleda sa određene daljine, to jeste operativni radnik nije fizički prisutan na mestu gde se nalazi sistem već preko računara nadgleda stanje sistema sa neke druge geografske lokacije. Tako da kvar zahteva putovanje operativnog radnika na teren, to jeste lokaciju gde se sistem nalazi.

Modelovanje greške radi simulacije je moguće izvršiti na više abstraknih nivoa (Slika 21). Postoji pet apstraktnih nivoa (Behavioral, Functional, Structural, Switch - level, Geometric).



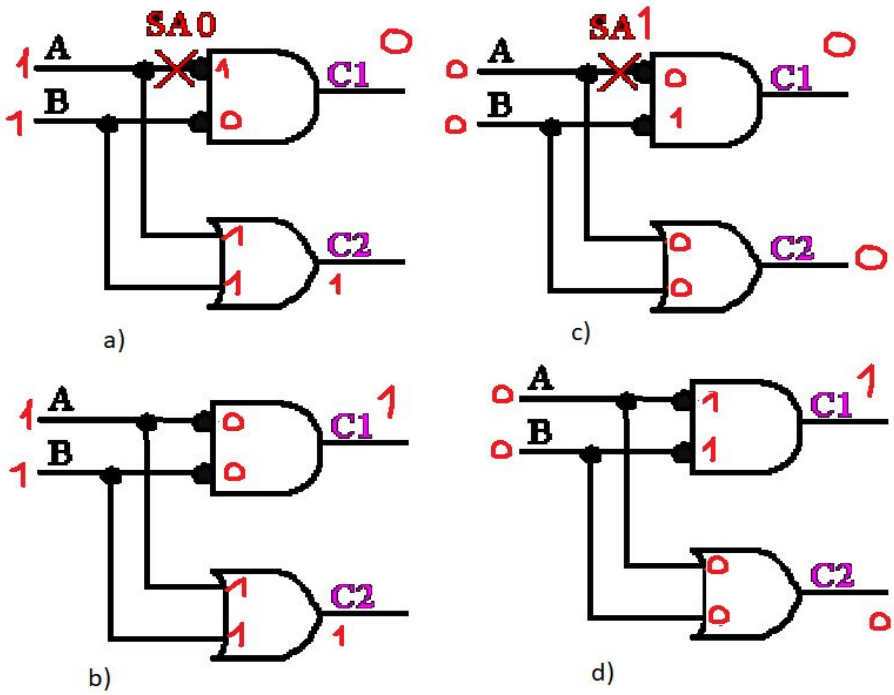
Slika 21: Abstrakcija nivoa

Modelovanje defekata će se vršiti na struturalnom nivou i takvi modeli se još nazivaju i “Structural Fault Models”. Kvarovi na ovom nivou utiču na interkonekcije između elemenata sistema tako da je to veran način da se simuliraju slučajevi pomenuti iznad u ovom poglavlju.

Jedan od najpoznatijih modela u ovoj kategoriji je **stuck-at** fault model. Ovaj model je poprilično popularan u industriji zbog dugovečnosti.

Postoji stuck-at 0 (SA 0) to je slučaj kada vod, to jeste linija stalno ima vrednost nula, tok je stuck-at 1 (SA 1) situacija gde linija uvek ima vrednost 1, bez obzira koja ja vrednost došla na ulaz.
Ako se ovaj model nalazi samo na jednoj liniji u sistemu, to se naziva **single SAF model**, dok ako je ovaj model primenjen na više linija u sistemu, tada se naziva **multiple SAF model**.

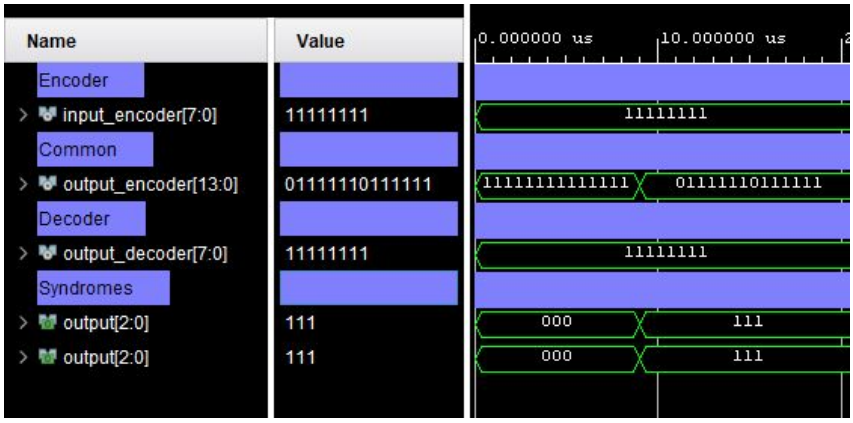
Primer SA0 i SA1:



Slika 22: a) Kolo sa SA0 za ulaze 11, b) Kolo bez SA0 za ulaze 11, c) Kolo sa SA1 za ulaze 00, d) Kolo sa bez SA1 za ulaze 00

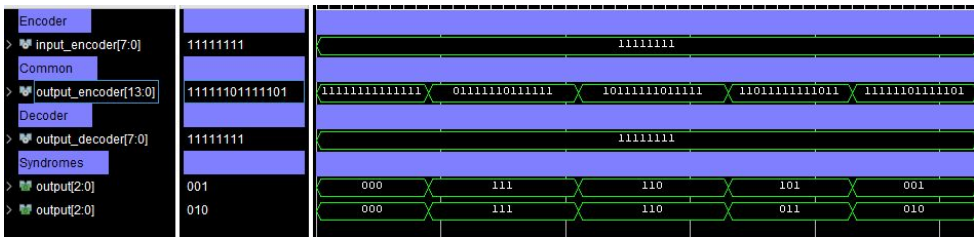
Kod SA0 (Slia 22, a)) rezultat I kola čiji su ulazi invertovani je nula, dok je vrednost u slučaju kada nema kvara (Slika 22, b)) je jedan. Kod SA1 (Slika 22, c)) izlaz invertora je nula, dok je za istu kombinaciju bez fault modela izlaz invertora jedan Kod SA1 (Slika 22, d)).

Vratimo se na testiranje top modula sa Slike xx. Na ulaz kodera se primi reč d = 11111111 i kao rezultat kodera dobija se kodna reč 11111111111111. Koristeći SA0 model na bitovima 13 i 6, na ulaz dekoder dođe vrednost 01111110111111 (Slika 22).



Slika 22: Testiranje kodera/dekodera, 1 bit greške za primer sa slike 20

Sindromi su u ovom slučaju 111, to jest sedam što predstavlja poziciju bita na kom se desila greška.
Kao što je već pomenuto sve dok je jedan bit greške u jednoj od nibli (gornjih ili donjih četiri bita podataka), greška će biti detektovana i korigovana (Slika 23). Ako dođe do greške u bitu parnosti, ta greška neće uticati na bite podataka, zato što se biti parnosti odbacuju i na izlaz dekodera se prikazuj osmobitna vrednost.

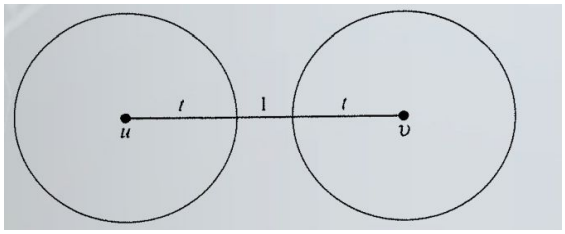


Slika 23: Testiranje kodera/dekodera, 1 bit greške više primera

Ako dođe do greške u dva ili više bita, u takvom slučaju greška neće moći biti detektovana, a ni korigovana pravilno (Slika 24). Recimo da su biti 13,12,11,10, 6 i 5 su biti greške, to jeste invertovani su sa jedinice na nulu i kao rezultat na dekoderu će biti vrednost 00000011, dok je ulaz na koderu bio 11111111.

Slika 24: Testiranje kodera/dekodera, više od jednog bita greške

Kao što je rečeno code distance, d(C) za Hamingov kod je 3 tako da važi d(C) ≥ 2t+1 (Slika 25), gde je t broj bita greške. Ako je uslov ispunjen broj bita greške će biti pravilno detektovan i korigovan.



Slika 25: Korekcija za t bita greške

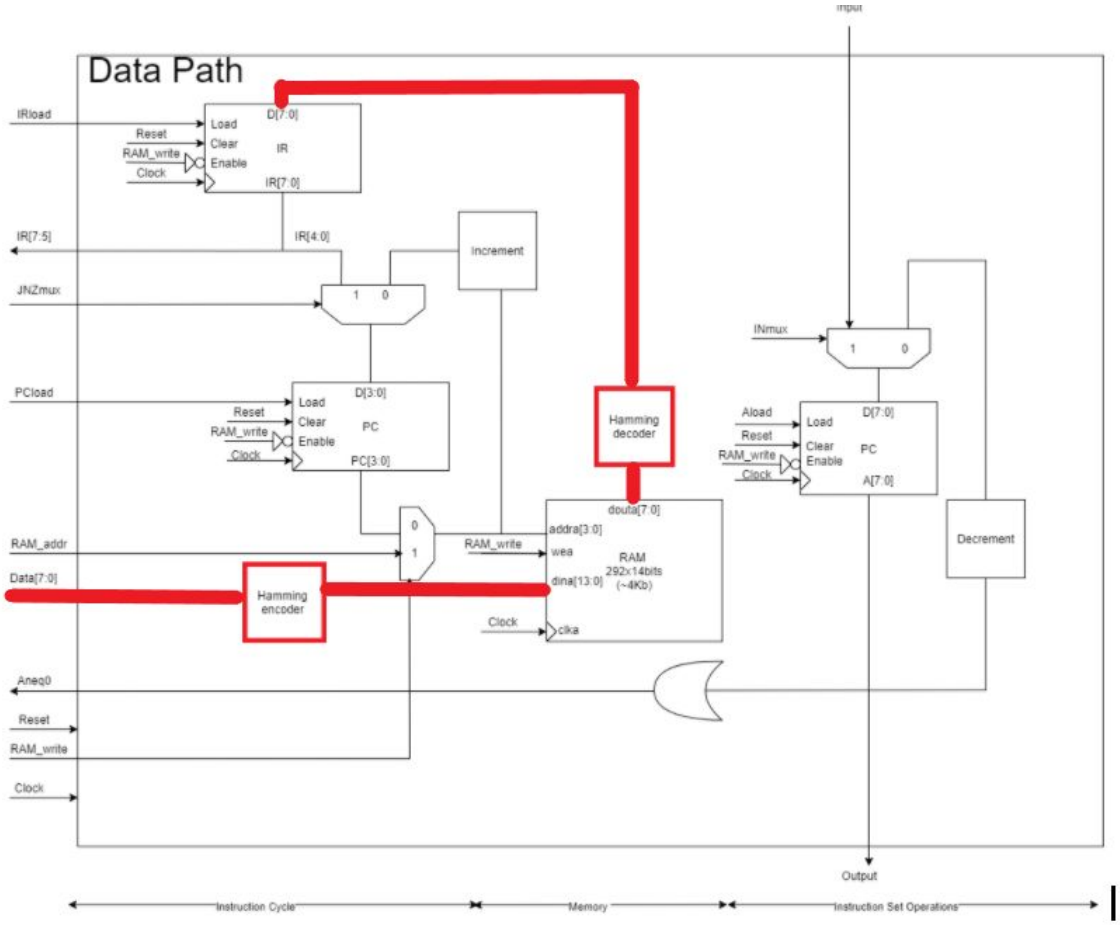
Na primeru sa slike xx, gde je kodna reč koja predstavlja izlaz iz enkodera, x = 11111111111111, a kodna u kojoj se desila greška je 0000111001111. U ovom slučaju je t = 6 tako da uslov:
d(C) ≥ 2t+1 -> 3 ≥ 2*6+1 -> 3 ≥ 13, to jeste uslov nije ispunjen i greška neće biti korigovana kako treba.

3.1 Integracija i testiranje Hemingovog kodera i dekodera u EC1 procesor

Nakon temeljnog testiranja Hemingovog kodera i dekodera potrebno je integrisati te komponente u EC1.

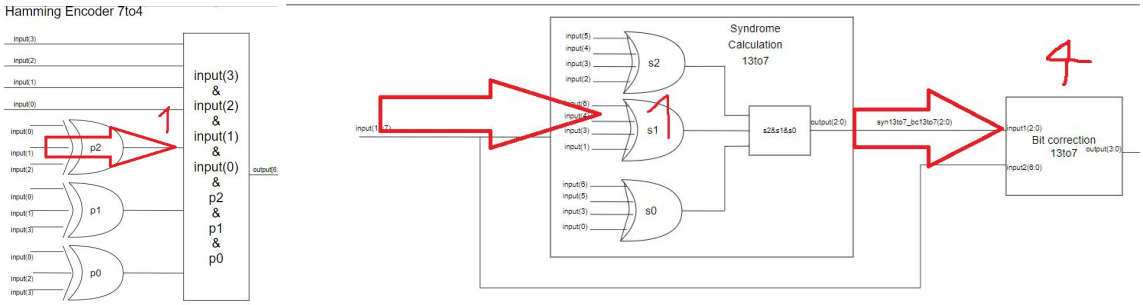
3.1.1 Integracija Hemingovog kodera i dekodera u EC1 procesor i analiza kašnjenja logičkih kapija

Data path EC1 procesora izgleda kao na slici 26.



Slika 26: Data Path nakon integracije Hamingovog kodera i dekodera

Kontrolna jedinica (Control Unit) je ostala nepromenjena. Tako da će podaci koji se upisuju u memoriju prvo proći kroz koder. Integracijom kodera kašnjenje se povećava za jedno xor kolo (Slika 27).



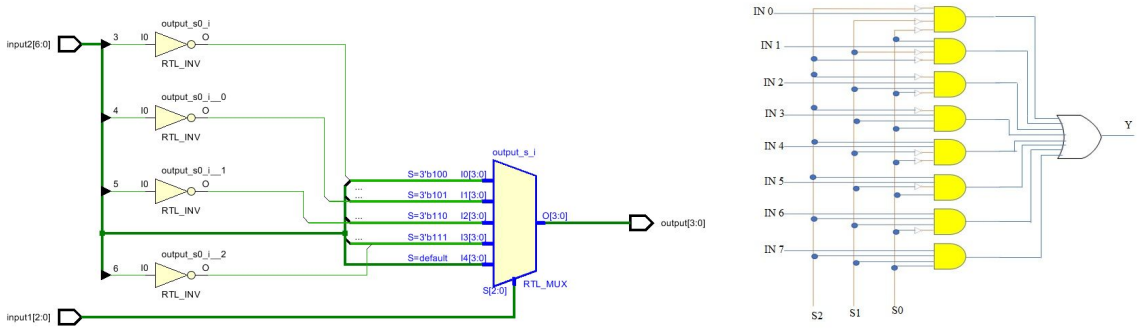
Slika 27: Analiza kašnjenja na osnovu broja logičkih kola

Kodna reč širine 7 bita jedne od nibli se spaja sa kodnom reči druge nible i 14 bita se upisuje u memoriju. Ceo proces se dešava dok je sistem neaktivan. Nakon što se memorija popuni instrukcijama, počinje čitanje tako kodovanih instrukcija. Pre nego što kodovana instrukcija bude primljena u instrukcioni registar, prvo prolazi kroz dekoder gde se detektuje, ispravlja bit greške kao i odbacuju biti parnosti. Izlaz dekodera je osmobitna reč koja se zatim upisuje u instrukcioni registar. Integracijom dekodera u sistem unosi se kašnjenje od 5 logička kola. Na putanji u syndrome calculation bloku se nalazi jedno xor kolo dok se u Bit correction-u nalazi četiri logička kola (not i mux8to1(not, and, or)). Iz raloga što je bit correction blok implementiran koristeći when case (Slika 28 a i b)

```
case input1 is

        when "100" =>
            output_s      <=  input2(6)&input2(5)&input2(4)&(not(input2(3)));
        when "101" =>
            output_s      <=  input2(6)&input2(5)&(not(input2(4)))&(input2(3));
        when "110" =>
            output_s      <=  input2(6)&(not(input2(5)))&input2(4)&(input2(3));
        when "111" =>
            output_s      <=  (not(input2(6)))&input2(5)&input2(4)&(input2(3));
        when others =>
            output_s      <=   input2(6 downto 3);
        end case;
```

Listing koda 1: Implementacija Bit correction bloka u vhdlu



Slika 28: Implementacija a) Implementacija Bit correction bloka u hardveru
b) Implementacija multipleksera 8 na 1 u hardveru

Da je Bit correction bio implementiran korišćenjem if else najduža putanja je 7 mux 2to1 (not, and, xor), dakle 21 logička kapija, tako da je korišćenje when case donosi smanjenje kašnjenja za 17 logičkih kapija.

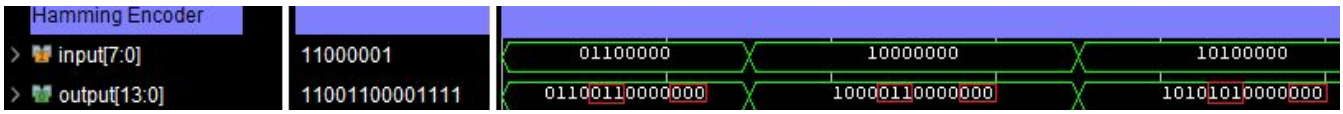
Integracijom kodera i dekodera se povećao broj logičkih kapija za 5, na putanji koja je obeležena crvenom linijom (Slika 26). Tako da se povećalo i kašnjenje.

3.1.2 Testiranje EC1 procesora nakon integrisanja Hamingovog Kodera i dekodera

Testiranje će se vršiti na isti način kao i testiranje Hamingovog kodera i dekodera.
Testiranje kodera će se svoditi na to da se upišu biti podataka u vidu EC1 instrukcija (Listing koda 1) gde biti podataka redom predstavljaju input, output, dec, jnz, halt instrukciju.
Tako da su uokvirene vrednosti biti parnosti (Slika 29).

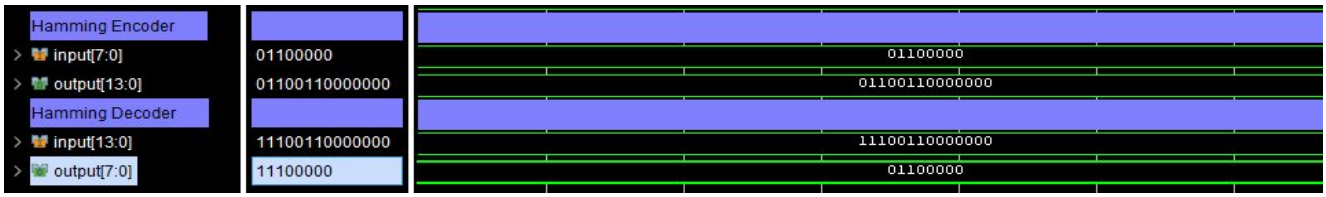
Listing koda 2: EC1 instrukcije koje se upisuju u RAM

RAM_Data <= "01100000","10000000" after 150 ns, "10100000" after 350 ns,"11000001" after 550 ns,"11111111" after 750 ns



Slika 29: Biti parnosti Hamingovog kodera

Testiranje dekodera će se izvršiti na isti način osim što će se dogoditi greška. Kodna reč sa greškom će biti upisana u memoriju. Kodna reč će biti ispraljvena i biti parnosti će biti odbačeni. (Slika 30)

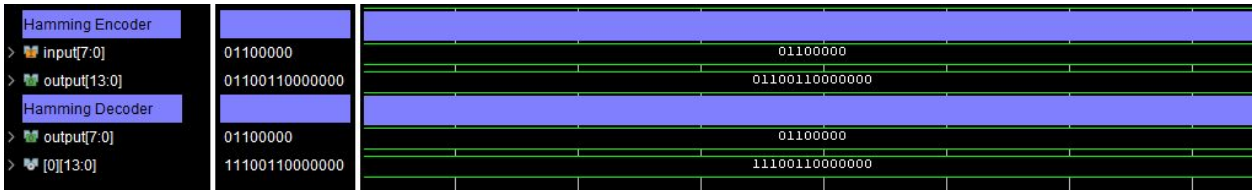


Slika 30: Rezultat Hamingovog dekodovanja

Moguće je da se u memoriju upiše tačna kodna reč širine 14 bita, i da se u međuvremenu neki bit u upisanoj kodnoj reči invertuje i na taj način predstavlja bit greške (Slika 31).

Na primer:

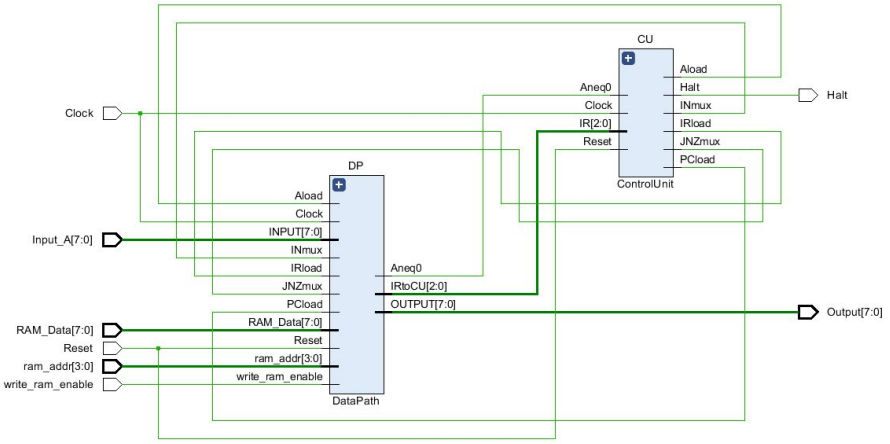
Ako je osmobitna reč upisana 01100000, kodna reč, c = 01100110000000 i ta vrednost se upisuje na nultu adresu u memoriju. Nakon vremena vrednost na adresi nula se promeni na 01100110000000. Dekoder će detektovati bit greške i korigovati taj bit. Takođe će se biti parnosti odbaciti i na izlazu dekodera će biti 01100000



Slika 31: Detektovanje i korekcija greške koja je nastala u memoriji.

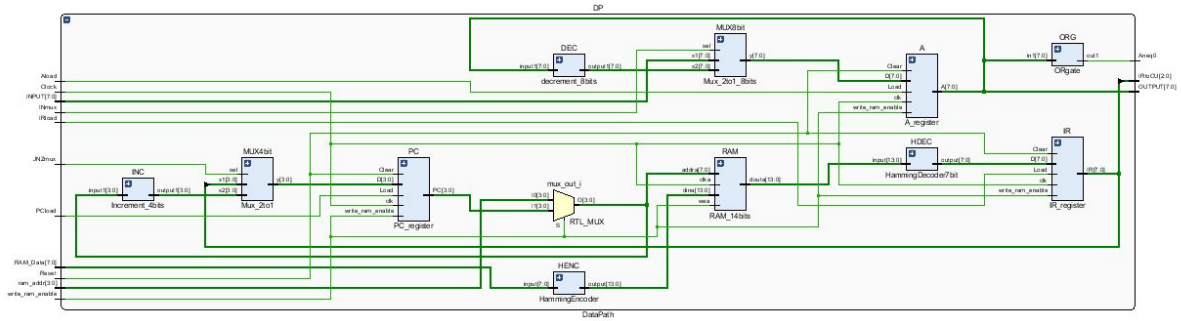
4.RTL analiza, sinteza, integracija

Part korištene ploče je xc7k70tfbv676-1, iz familije Kintex-7.
Nakon izvršene RTL analize, šematik EC1 procesora izgleda kao na slici 32



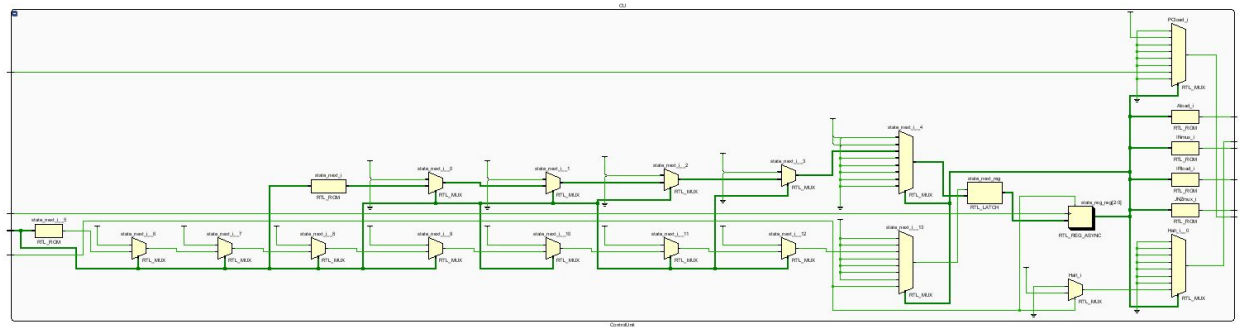
Slika 32: EC1 šematik

Data path šematik nakon RTL analize izgleda kao na slici 33



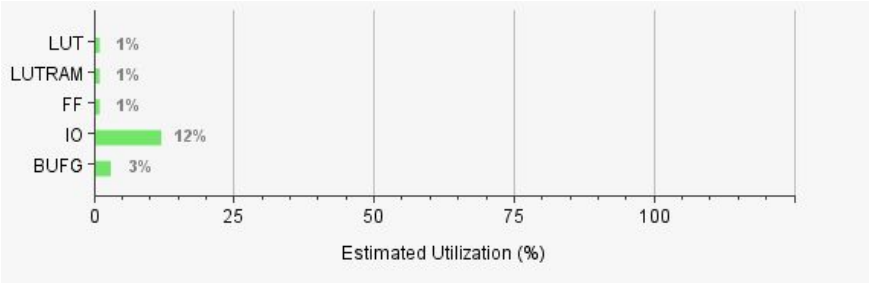
Slika 33: Data path šematik

Control unit šematik nakon RTL analize izgleda kao na slici 34



Slika 34: Control unit šematik

Iskorišteni resursi su prikazani na slici 35



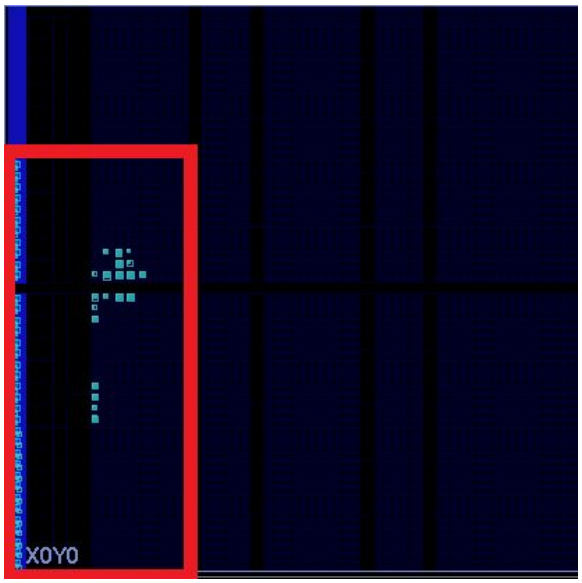
Slika 35: Iskorišćeni resursi

Temperature i potrošnja na su prikazani na slici 36

Total On-Chip Power:	**0.461 W**
Junction Temperature:	**25.9 °C**
Thermal Margin:	59.1 °C (31.2 W)
Effective ϑJA:	1.9 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low
Implemented Power Report	

Slika 36: Temperatura i potrošnja

Nakon impelemntacije, dizajn se namapirao na elemente ploče u sektoru X0Y0 (Slika 37)



Slika 37: Implementacija EC1 dizajna na ploči

5.Zaključak

Iako je EC1 jednostavan procesor, postoji potreba za zaštitom sistema, jer se na taj način povećava pouzdanost sistema. Najkritičniji element u procesoru je bila memorija, tako da je bilo potrebno impelemntirati mehanizam zaštite. Izabrani model zaštite je bio Hamingov kod, i to Hamingov kod 7,4 iz razloga što primenom na osmobitnu magistralu omogućava detektovanje i korekcija greške. Na ovaj način je sistem postao otporniji na greške. Takođe i blok koji radi korekciju greške je implementiran na način tako da se smanji propagacija u vidu logičkih kola. U tom slučaju je upotrebljen manji broj lokičkih kola i samim tim pored smanjenja kašnjenja, smanjeni su i troškovi.