ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

HARDVERSKA IMPLEMENTACIJA BLAKE ALGORITMA ZA HEŠIRANJE

Kandidat:

Goran Ognjanović 2011/3297

Mentor:

doc. dr Zoran Čiča

Beograd, Septembar 2014.

SADRŽAJ

1. UVOD

U današnje vreme postoji neprestana potreba za što bržim heširanjem kod aplikacija koje zahtevaju proveru integriteta, digitalni potpis, autentičnost podataka, *cloud* skladištenja, detekciju napada (*host-based intrusion*), sisteme za kontrolu verzija i dr.

Kriptografski BLAKE heš algoritam je bio jedan od pet finalista poslednje treće runde na takmičenju za novi SHA-3 standard koje se završilo u novembru 2012. godine. Iako je pobednik bio Keccak heš algoritam, Blake se smatra kao veoma snažna heš funkcija koja ima veliku sigurnosnu marginu i veoma dobre performanse, a takođe je podobna i za softversku i za hardversku implementciju.

Zvanično postoje 4 verzije BLAKE algoritma čija se razlika ogleda u dužini izlaznog heš podatka u skladu sa zahtevima sa SHA-3 takmičenja. Dužine heš izlaza su 224, 256, 384 i 512 bita pa su imena verzija algoritma BLAKE-224, BLAKE-256, BLAKE-384, BLAKE-512, respektivno. U ovom radu fokus će biti na BLAKE-256 algoritmu.

U ovom radu biće prikazane specifikacija i performanse BLAKE algoritma. Biće prikazane tri različite hardverske implementacije, pri čemu će biti analizirane njihove prednosti i mane u odnosu jedna na drugu, kao i iskorišćenje resursa u odnosu na brzinu.

Sama funkcionalna verifikacija BLAKE-256 algoritma sprovedena je u testbenč okruženju napisanom u VHDL jeziku. Za simulaciju se koristio ModelSim 10, dok se za generisanje ulaznih i očekivanih izlaznih podataka koristio referentni softverski model napisan u C++ koji je priložen od strane autora u okviru konkursa za SHA-3 standard.

Ostatak rada je oragnizovan na sledeći način. U drugom poglavlju biće opisane osnove heš algoritama, razlike između opštih i kriptografskih heš funkcija i istoriju standardizacije kriptografskih heš algoritama. Takođe će biti opisani najpoznatiji kriptografski heš algoritmi kao i napadi na njih. Detaljan opis verzije Blake heš algoritma BLAKE-256 biće opisan u trećem poglavlju. U četvrtom poglavlju biće opisana implementacija BLAKE-256 modula, pri čemu će biti opisane tri različite verzije implementacije. U petom poglavlju biće prikazana verifikacija BLAKE-256 modula, opisan detaljan postupak verifikacije i validnosti svake od tri verzije implemetacije BLAKE-256 algoritma. Performanse BLAKE-256 modula za svaku implementaciju u vidu hardverskih zahteva i brzine rada biće date u šestom poglavlju. U sedmom poglavlju biće data zaključna razmatranja u kojima se govori o trenutnoj primeni, kao i o budućnosti i naslednicima BLAKE algoritma. Sav izvorni kod kao i sam tekst rada je dostupan u elektonskoj formi.

2. KRIPTOGRAFSKI HEŠ ALGORITMI

U ovom poglavlju biće opisane osnove heš algoritama, razlike između opštih i kriptografskih heš funkcija, istorija standardizacije, a biće opisani i najpoznatiji kriptografski heš algoritmi kao i napadi na njih.

2.1. HEŠ ALGORITMI

Heš funkcije transformišu ulazni podatak promenjive dužine u numerički heš ili sažetak (kratak pregled) određene fiksne dužine. Često se koriste u digitalnim sistemima radi provere integriteta poslatih podataka, brzog traženja poruka u bazi podataka ili mapiranja poruka sa indeksima tabele. U polju kriptografije, heševi imaju primenu u procesu autentifikacije korisnika, digitalnom potpisivanju dokumenata i generisanju statistički slučajnih nizova podataka.

Jak heš algoritam može brzo da generiše heš iz ulaznih podataka i da pritom bude praktično nemoguće napadačima da otkriju originalne ulazne podatke na osnovu poznavanja vrednosti izlaznog heša. Takođe, verovatnoća kolizije, gde se za dve ili više različitih poruka generiše identičan heš na izlazu, mora biti zanemarljivo mala. Svaka promena ulazne poruke, ma koliko bila mala, treba da izazove drastičnu promenu izlaznog heša. Ukoliko heš funkcija ne sadrži ove osobine smatra se slabom, i te osobine prave razliku između kriptografskih heš algoritama i heš algoritama opšte namene. Kriptografski heš algoritmi stoga moraju imati veliku kompleksnost, vreme računanja i veličinu izlaznog podatka.

2.2. NIST I SHA

NIST (*National Institute of Standards and Techology*) je ustanovio set standardnih heš funkcija nazvanih SHA (*Secure Hash Algorithm*). Prva verzija SHA-0 objavljena je 1993. godine. Godine 1995. objavljena je nova SHA-1 verzija koja je postala prihvaćena širom sveta kao standard za autentifikaciju digitalnih podataka i digitalnog potpisa. Godine 2001. izlazi SHA-2 standard kako bi se uklonile određene slabosti kod SHA-1 algoritma.

NIST organizacija je 2007. godine objavila globalno takmičenje za pronalaženje nove SHA heš funkcije. Algoritmi koji su kandidovani za novi standard bili su strogo analizirani u pogledu sigurnosti, performansi i samog dizajna. Godine 2010. objavljena je konačna lista od pet finalista među kojima je BLAKE algoritam zauzeo mesto kao jedan od najvećih favorita za novi SHA-3 algoritam. Međutim, 2012. godine BLAKE nije postao pobednik takmičenja za SHA-3 standard, ali njegova primena i potencijal su i dalje veliki.

2.3. POZNATI HEŠ ALGORITMI

Danas se među najpoznatijim i najčešće korišćenim heš algoritmima smatraju MD5, SHA-1 i SHA-2.

2.3.1. Heš algoritam MD-5

MD5 (*Message-Digest algorithm 5*) je kriptografski algoritam koji spada u grupu najpopularnijih i najčešće korišćenih heš algoritama. Dužina sažetka odnosto izlaza heš algoritma je 128 bita.

MD5 algoritam je razvio 1991. godine Ronald Rivest. Baziran je na MD4 algoritmu, i iako je nešto sporiji od MD4 algoritma, MD5 je mnogo sigurniji. Avgusta 2004. otkriveno je da MD5 algoritam nije otporan na koliziju. Za razbijanje ovog algoritma bio je potreban samo jedan sat na IBM p690 klasteru.

Dugo je bio veoma primenjen u mnogim oblastima zaštite podataka. Danas se smatra da je podložan kriptografskim napadima i smanjena mu je primena u kriptografiji poput digitalnog potpisa, skladištenja šifri itd, dok i dalje nalazi veoma čestu primenu u proveri integriteta većih fajlova zbog svoje brzine.

2.3.2. Heš algoritam SHA-1

SHA-1 je objavljen 1995. godine. Iako veoma sličan SHA-0 algoritmu imao je veću sigurnost i primenu. Dužina SHA-1 izlaza heš algoritma je 160 bita odnosno 20 bajtova. Iako je algoritam razbijen i ima naslednike SHA-2 i SHA-3, trenutno je SHA-1 heš algoritam još uvek najraspostranjeniji SHA algoritam u svetu. Takođe se koristi u nekoliko masovno korišćenih aplikacija i protokola.

2.3.3. Heš algoritam SHA-2

SHA-2 predstavlja set od 6 kriptografskih heš funkcija (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256) dizajniran od strane U.S. National Security Agency (NSA) i NIST ga je objavio 2001. godine kao U.S. Federal Information Processing Standard (FIPS). SHA-2 je uveo značajne razlike u odnosu od SHA-1 heš algoritam. Dužine izlaza SHA-2 heš funkcija su 224, 256, 384 ili 512 bita. Do danas algoritam još uvek nije razbijen.

2.3.4. Heš algoritam SHA-3/Keccak

Keccak predstavlja novi SHA-3 standard. SHA-3 je stvoren kao alternativa u slučaju da SHA-2 bude razbijen.

2.4. RANJIVOSTI I NAPADI

2.4.1. Napad „Collision attack“

Napad kolizijom (*collision attack*) na kriptografske heš funkcije pokušava da pronađe dva ulaza koja daju istu heš vrednost (heš kolizija ili sudar). Za razliku od inverznog napada, heš vrednost nije određena. Otpornost heš funkcije na napad kolizijom znači da je praktično neizvodljivo da se pronađu dve poruke m1 i m2, takve da je $m1 \neq m2$, i da važi hash(m1) = hash(m2).

Postoje dve vrste napada kolizijom:

- Pronalaženje dve različite poruke m1 i m2 takve da važi hash(m1) = hash(m2) predstavlja „Osnovni napad kolizijom“.

- „Napad kolizijom tipa izbor-prefiksa“ predstavlja napad tako da se za dva zadata različita prefiksa p1 i p2 pronađu dva dodatka m1 i m2 takva da važi hash(p1 // m1) = hash(p2 // m2) (gde je // operacija spajanja nizova).

2.4.2. Napad „Preimage attack“

Napad orginala (*preimage attack*) pokušava da pronađe poruku koja daje zadatu heš vrednost. Otpornost heš funkcija na ovu vrstu napada u suštini znači da je za računarski neizvodljivo da se za poznatu vrednost izlaza nađe ulaz koji daje generiše takav heš izlaz, odnosno da se nađe x takvo da je h(x) = y gde je y poznati izlaz.

2.4.3. Napad „Second Preimage attack“

Napad drugog orginala (*second preimage attack*) pokušava da pronađe drugu poruku koja daje istu heš vrednost kao zadata poruka. Otpornost na ovaj napad podrazumeva da je računarski neizvodljivo pronaći neki drugi ulaz koji daje isti heš izlaz kao originalni ulaz, odnosno za zadato x, neizvodljivo je pronaći $x' \neq x$ za koje važi hash(x) = hash(x′). Otpornost na napad drugog orignala ne garantuje otpornost na napad orginala.

2.4.4. Napad „Dictionary attack“

Napad pomoću rečnika (*dictionary attack*) koristi rečnik uobičajenih reči kod pronalaženja odgovarajućeg ulaza. Ovaj napad zasniva se na činjenici da su korisnici skloni izboru uobičajenih lozinki, na primer često upotrebljavane reči i cifre dodate na kraju. Tako napadač može da izgradi listu zajedničkih, najčešće korišćenih, lozinki i njihovih heš vrednosti (rečnik) i kod napada sistematski pokušava svaku kombinaciju iz ovog skupa. Ovaj pristup može da uštedi vreme, jer ne mora da se pokrije ceo prostor mogućih heš vrednosti kao kod napada brutalnom silom. Nedostatak je što ne postoji nikakva garancija da će se traženi ulaz naći u upotrebljenom rečniku.

Napad pomoću rečnika retko je uspešan protiv sistema koji koristi u poruci više reči, a neuspešan je protiv sistema koji upotrebljava nasumične kombinacije velikih i malih slova pomešanih sa brojevima. Napad se može otežati dodavanjem slučajnog niza znakova (*salt* vrednost) na lozinku pre izvršavanja heš funkcije.

2.4.5. Napad „Birthday attack“

Rođendanski napad (*birthday attack*) koristi matematičku osnovu problema rođendana iz teorije verovatnoće da se smanji složenost otkrivanja kolizije heš funkcije. Problem rođendana razmatra verovatnoću da se u slučajnom skupu nalaze dve osobe sa istim datumom rođenja. Pokazuje se da je verovatnoća jednaka 100% kada broj članova skupa dosegne 367 (365 dana i 29. februar), 99.9 % za 70 osoba i 50% kada broj osoba u skupu dosegne 23. Pronalaženje dve osobe sa istim slučajnim rođendanom ekvivalentno je pronalaženju dve različite poruke koje daju istu heš vrednost. Za pronalaženje poruke sa zadatom heš vrednosti dužine m bita potrebno je hešovati 2m slučajnih poruka, dok pronalaženje dve slučajne poruke sa istom heš vrednosti zahteva hešovanje svega 2m/2 slučajnih poruka. Ova vrsta napada može da se koristi za zloupotrebu komunikacije između dva ili više učesnika.

2.4.6. Napad „Brute-force attack“

Napad brutalnom silom (*brute-force attack*) sastoji se od sistematske provere svih mogućih ulaza. Prednost ove metode je da se garantuje uspeh u pronalaženju ispravnog ulaza, međutim ovaj

napad je vremenski izuzetno zahtevan proces, i u najgorem slučaju podrazumeva prolaz kroz ceo prostor rešenja. Ovaj napad se uvek može primeniti kada nije moguće iskoristiti slabosti kriptografskog sistema i upotrebiti drugu vrstu napada.

2.4.7. Napad „Rainbow table“

Tabela duge je redukovana tabela preračunatih heš vrednost koja se koristi za pronalaženje ulaza za zadatu heš vrednost. To nije kompletna heš tabela sa svim ulazima i njihovim heš vrednostima koja po pravilu zahteva preveliki memorijski prostor. Tabela duge je daleko manja i predstavlja kompromis između memorijskog prostora i vremena pretraživanja.

Svaka vrsta tabele se sastoji od početnog ulaza i finalne heš vrednosti koja je dobijena posebnim postupkom ulančavanja. Svaki korak ovog postupka sastoji se od primene jednosmerne heš funkcije i primene redukcione funkcije koja dobijenu heš vrednost ponovo preslikava u skup mogućih ulaza (poruka). Redukciona funkcija nije inverzna heš funkcija, jedino daje jednu od mogućih poruka koja je ulaz za heš funkciju u narednom koraku. Posle uzastopne primene velikog broja koraka na zadnji ulaz primenjuje se heš funkcija i dobija se finalna heš vrednost koja se ugrađuje u tabelu.

Na osnovu početne poruke mogu se reprodukovati sve ulazne poruke primenom određenog broja koraka postupka hešovanja i redukcije. Na taj način u svaku vrstu tabele je praktično ugrađen veliki broj mogućih ulaza, a sama tabela je značajno smanjena.

Pretraživanje tabele koristi finalne vrednosti u vrstama tabele i identičan postupak koji je korišćen kod formiranja vrsta tabele je sada primenjen na zadatu heš vrednost za koju se traži odgovarajući ulaz.

Prvo se pokušava pretraživanje heš vrednosti u tabeli, ako se pronađe vrednost onda se primeni postupak reprodukcije ulaza na osnovu početne vrednosti iz tabele i posle odgovarajućeg broja koraka dobija se traženi ulaz.

Ako nije pronađena heš vrednost u tabeli, na nju se primenjuje redukcija i heširanje i za novu heš vrednost ponavlja se prethodni postupak, s tom razlikom da je broj koraka reprodukcije manji za jedan. Ceo postupak se ponavlja do pronalaženja traženog ulaza ili do neuspešnog iscrpljivanja tabele.

Osnovni problem kod tabele duge je mogućnost kolizije ulaza što u nekim slučajevima može dovesti do petlje u postupku formiranja ili pretraživanja tabele. Kod tabele duge koriste se u jednoj tabeli različite redukcione funkcije, i ako bi njima dodelili boje dobila bi se slika koja podseća na dugu (od početnog ulaza do finalne heš vrednosti), što je motiv za davanje naziva ovoj tabeli.

2.4.8. Napad „Side-channel attack“

Napad bočnim kanalom (*side-channel attack*) je svaki napad koji se zasniva na informacijama dobijenim iz fizičke implementacije kriptografskog sistema, umesto da se koristi brutalna sila ili teorijske slabosti primenjenih algoritama. Na primer vremenski redosled, potrošnja energije, elektromagnetno zračenje ili akustički signali mogu predstavljati dodatni izvor informacija za probijanje kriptografskih mehanizama. Neki put je potrebno tehničko znanje o unutrašnjem funkcionisanju sistema ili je dovoljno primeniti diferencijalnu analizu snage i tretirati sistem kao crnu kutiju.

2.4.9. Napad „Length extension attack“

Napad proširenjem dužine (*length extension attack*) poruke primenjuje se na heš funkcije koje omogućuju uključivanje dodatnih informacija. Ovaj napad se može primeniti na konstrukcije hash(tajna // poruka) kada se zna poruka i dužina tajnog dela.

3. BLAKE HEŠ ALGORITAM

Heš funkcija BLAKE-256 radi sa rečima dužine 32 bita i kao izlaz vraća heš vrednosti dužine 32 bajta (256 bita). BLAKE je iterativni algoritam i bazira se na veoma poznatoj i prihvaćenoj HAIFA(*Hash Iterative Framework*) strukturi:

$$h^0 \leftarrow IV$$

$$\textbf{\textit{for}}\ i = 0, \dots, N-1$$

$$h^{i+1} \leftarrow \textbf{\textit{compress}}(h^i, m^i, s, t^i)$$

$$\textbf{\textit{return}}\ h^N$$

Pre početka heširanja ulazna poruka *m* dužine $l < 2^{64}$ bita se deli na *N* blokova dužine 512 bita $m^0, \dots, m^{N-1}$ (svaki blok sadrži 16 reči dužine po 32 bita). Blokovi se zatim prosleđuju blok po blok kroz kompresionu funkciju zajedno sa dotadašnjim hešom. Pre podele na blokove ako je potrebno radi se *padding*[1]. Rezultat procesiranja jednog bloka je novi heš i proces se ponavlja dok svih *N* blokova ne prođu kroz kompresiju posle čega heš vrednost predstavlja finalni heš rezultat.

BLAKE je *full diffusion* algoritam što podrazumeva da promena makar i jednog bita na ulaznoj poruci, saltu ili inicijalizacionom vektoru može da utiče na svaki izlazni bit posle samo dve transformacione runde.

3.1. FUNKCIJA KOMPRESIJE

BLAKE-256 funkcija kompresije kao ulaz uzima 4 vrednosti:

- Niz trenutne heš vrednosti $h = h_0, \dots, h_7$
- Jedan blok ulazne poruke $m = m_0, \dots, m_{15}$
- Salt $s = s_0, \dots, s_3$
- Brojač $t = t_o, t_1$

Ova četiri ulaza predstavljena su sa ukupno 30 reči dužine po 32 bita. Izlaz ove funkcije je nov lanac vrednosti $h' = h'_0, \dots, h'_7$ dužine 8 reči tj. 256 bita.

[1] *Padding* (dopuna) predstavlja dodavanje ekstenzije poruci da bi tako novonastala poruka mogla da se podeli na celobrojni broj blokova dužine 512 bita. Prilikom dodavanja na poruke se dodaje bit 1 praćen određenim brojem bitova vrednosti 0. Na kraju dodajemo poslednji bit 1. *Big-endian* (MSB…LSB) reprezentacija poruke dužine *l*:

$$m \leftarrow m \parallel 1000 \dots 0001\langle l \rangle_{64}$$

Na početku vrednosti heša h^0 odgovara istim inicijalnim vrednostima:

$$IV_0 = 6A09E667$$
$$IV_1 = BB67AE85$$
$$IV_2 = 3C6EF374$$
$$IV_3 = A54FF53A$$
$$IV_4 = 510E527F$$
$$IV_5 = 9B05688C$$
$$IV_6 = 1F83D9AB$$
$$IV_7 = 5BE0CD19$$

Brojač t predstavlja koristan broj bita koji je do tog trenutka bio obrađen. On obezbeđuje da identični blokovi na različitim mestima u ulaznoj poruci vraćaju različitu heš vrednost.

Salt vrednost s povećava sigurnost jer korisnik na taj način može da za isti podatak napravi više različitih heš vrednosti. Podrazumevana (difolt) vrednost salta je nula.

Funkcija kompresije čiji su ulazi h, m, s, t a izlaz h' se predstavlja izrazom:

$$h' = \boldsymbol{compress}\ (h, m, s, t)$$

Procedura heširanja poruke m sa BLAKE-256, gde je m originalna poruka bez dopune (*padding*), a s predstavlja salt, se predstavlja sa:

$$BLAKE\_256(m, s) = h^N$$

Blake-256 koristi 16 konstanti (prve cifre od π) :

$$C_0 = 243F6A88 \qquad C_1 = 58A308D3$$
$$C_2 = 13198A2E \qquad C_3 = 03707344$$
$$C_4 = A4093822 \qquad C_5 = 299F31D0$$
$$C_6 = 082EFA98 \qquad C_7 = EC4E6C89$$
$$C_8 = 452821E6 \qquad C_9 = 38D01377$$
$$C_{10} = BE5466CF \qquad C_{11} = 34E90C6C$$
$$C_{12} = C0AC29B7 \qquad C_{13} = C97C50DD$$
$$C_{14} = 3F84D5B5 \qquad C_{15} = B5470917$$

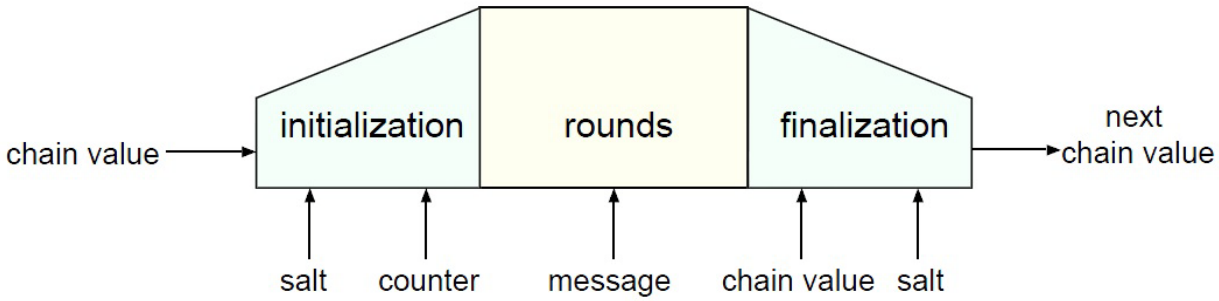
i deset permutacija od $\{0, \dots, 15\}$ se koriste u svim BLAKE funkcijama i prikazane su u Tabeli 3.1.1.

Tabela 3.1.1. - Prikaz vrednosti permutacija

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

Funkcija kompresije je podeljena na 3 faze izvršavanja (slika 3.1.1):

- faza inicijalizacije
- faza izvršavanja rundi
- faza finalizacije



Slika 3.1.1. - Funkcija kompresije

3.2. FAZA INICIJALIZACIJE

Podrazumevana vrednost salta je nula. Stanje v je veličine 16 reči dužine 32 bita. Stanje v inicijalizujemo tako da različiti ulazi generišu različita inicijalna stanja. Stanje je predstavljeno kao matrica 4x4 i popunjava se:

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

3.3. FAZA RUNDI

Dizajn bilo kojeg kriptografskog heša uključuje balansiranje između sigurnosti, performansi i potrošnje u hardverskoj implementaciji. Parametar koji utiče na te osobine je broj rundi po kompresiji. BLAKE-256 sadrži 14 rundi.

Jednom kada je stanje inicijalizovano, funkcija kompresije započinje da izvršava seriju od 14 rundi. Runda predstavlja transformaciju stanja v i sastoji se od 8 G funkcija:

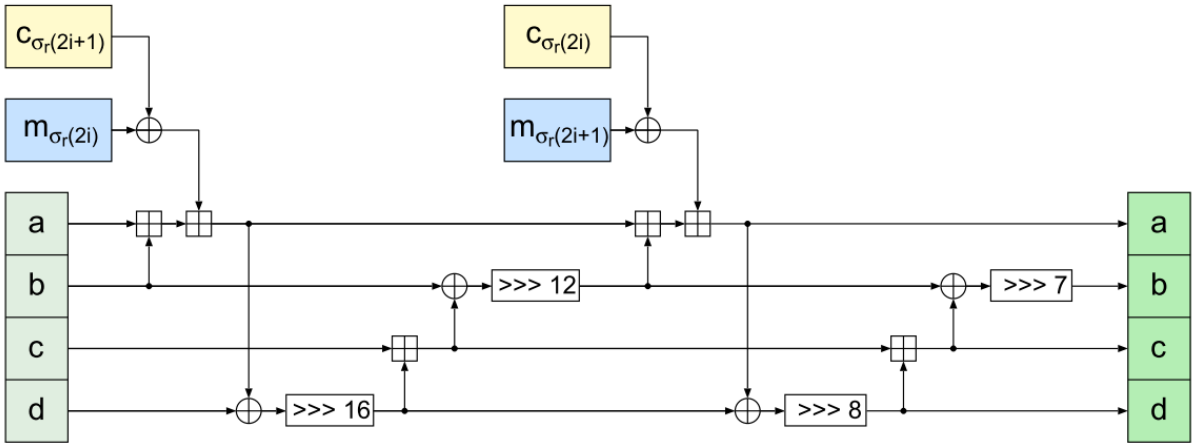
$$G_0(v_0, v_4, v_8, v_{12})$$
$$G_1(v_1, v_5, v_9, v_{13})$$
$$G_2(v_2, v_6, v_{10}, v_{14})$$
$$G_3(v_3, v_7, v_{11}, v_{15})$$
$$G_4(v_0, v_5, v_{10}, v_{15})$$
$$G_5(v_1, v_6, v_{11}, v_{12})$$
$$G_6(v_2, v_7, v_8, v_{13})$$
$$G_7(v_3, v_4, v_9, v_{14})$$

Za rundu r, $G_i(a, b, c, d)$ izvršava sledeći niz operacija:

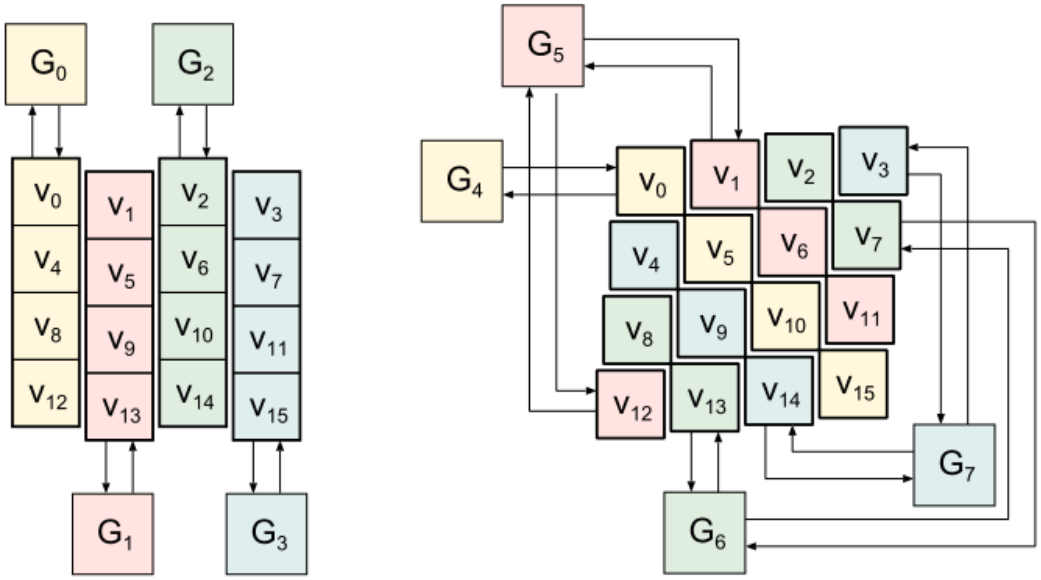
$$a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)})$$
$$d \leftarrow (d \oplus a) \ggg 16$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) \ggg 12$$
$$a \leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)})$$
$$d \leftarrow (d \oplus a) \ggg 8$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) \ggg 7$$

G transformacije su dizajnirane tako da maksimalno umanje verovatnoću lokalne kolizije, koja nastaje kada dve različite poruke iniciraju isto interno stanje posle istog broja rundi.

Prve četiri G funkcije G_0, G_1, G_2, G_3 se mogu izvršavati u paraleli jer svaka od njih menja matricu po kolonama i nema poklapanja sa između njih. Ova faza se naziva *column step*. Posle ove faze G_4, G_5, G_6, G_7 funkcije se takođe izvršavaju u paraleli jer svaki od njih menja matricu po dijagonalama i nema poklapanja sa između njih. Ova faza se naziva *diagonal step*. Za runde gde je $r > 9$, koristimo permutacije u obliku $\sigma_{r \bmod 10}$. Slike 3.3.1 i 3.3.2 ilustruju G_i funkciju kao i faze njihovog izvršavanja.



Slika 3.3.1. - G_i funkcija



Slika 3.3.2. - Faze izvršavanja

3.4. FAZA FINALIZACIJE

Posle sekvence rundi, novi lanac vrednosti $h'_0, \dots, h'_7$ je proračunat iz stanja $v_0, \dots, v_{15}$ pri čemu u proračun ulaze i ulazni inicijalni lanac vrednosti $\mathrm{h}_0, \dots, \mathrm{h}_7$ i salt $\mathrm{s}_0, \dots, \mathrm{s}_3$:

$$\mathrm{h}'_0 \leftarrow \mathrm{h}_0 \oplus \mathrm{s}_0 \oplus \mathrm{v}_0 \oplus \mathrm{v}_8$$

$$\mathrm{h}'_1 \leftarrow \mathrm{h}_1 \oplus \mathrm{s}_1 \oplus \mathrm{v}_1 \oplus \mathrm{v}_9$$

$$h'_2 \leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10}$$

$$h'_3 \leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11}$$

$$h'_4 \leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12}$$

$$h'_5 \leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13}$$

$$h'_6 \leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14}$$

$$h'_7 \leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}$$

4. BLAKE-256 MODUL

U ovom poglavlju biće data opšte napomene o upotrebi i značaju IP (*Intellectual Property*) jezgara, biće opisan modul (tj. IP jezgro) BLAKE-256 algoritma kao i verzije njegovih implementacija koje balansiraju parametre zauzeća hardverskih resursa i brzine radnog takta.

4.1. IP jezgra

Najpoznatiji metod za povećanje produktivnosti dizajniranja sistema na ASIC ili FPGA platformama je korišćenje komponenata za višekratnu upotrebu tzv. IP jezgara. Ove komponente su prethodno verifikovane što smanjuje vreme verifikacije celog sistema. IP jezgra se dele u tri kategorije:

- Hard IP - *layout* maske
- Firm IP - sintetizovane netliste za specifičnu tehnologiju
- Soft IP - sintetizibilan HDL (Hardware Description Language) kod

Soft IP jezgra predstavljaju HDL verzije jezgara, koji imaju ugrađenu skalabilnost i parametrizaciju. Dizajnirani su tako da mogu da budu sintetizovani u hardver za veliki opseg funkcionalnosti i procesa. Parametri određuju funkcionalnost i prilikom sintetizacije utiču na pravljenje željenog hardvera.

Parametrizacija Soft IP jezgra nudi najveću fleksibilnost, skalabilnost i portabilnost, obezbeđuje najveći stepen iskorišćenja za višekratnu upotrebu. Sa druge strane verifikacija i testiranje su znatno kompleksniji. Takođe, u odnosu na Hard IP jezgra imaju degradirane performanse u pogledu zauzeća površine, potrošnje i brzine.

4.2. BLAKE MODULI

BLAKE-256 modul tj. IP jezgro predstavlja implementaciju visokih performansi BLAKE kriptografskog heš algoritma, kandidata za SHA-3 standard. BLAKE IP jezgro nalazi primenu u različitim aplikacijama koje koriste digitalni potpis, neki drugi vid autentifikacije porekla poruke ili zaštite od neautorizovanih pristupa. To su aplikacije koje se koriste za e-trgovinu, zaštitu integriteta podataka, masovno enkriptovanje, za zaštitu bežičnih mreža, za autentifikaciju i autorizaciju, kao i integritet komunikacije između mrežnih uređaja u mreži itd.

BLAKE IP jezgro na ulazu prihvata dugačku poruku promenjive dužine i kao izlaz daje heš rezultat fiksne dužine. Jezgro se sastoji od dva glavna modula - ulaznog interfejs modula i BLAKE centralne jedinice tj. funkcije kompresije. BLAKE centralna jedinica izvršava BLAKE petlju nad blokom poruke dužine 512 bita, dok ulazni interfejs modul sadrži brojač koji broji koliko je do tog trenutka bita poruke heširano.

IP jezgru se mora poslati na početku dužina poruke bez *padding* bita. Dužina je predstavljena kao 64-bitni podatak. Zatim, jezgro prima na ulazu reč po reč dužine 64 bita dok ne primi ceo blok poruke od 512 bita (signal *input_enable* je aktivan tokom prijema bloka). Kada se

primi ceo blok poruke, ulazni tok podataka se pauzira i počinje njihova kompresija. Posle obrade bloka jezgro prima novi blok i radi kompresiju. Na kraju obrade poslednjeg bloka poruke se generiše rezultujući heš poruke (izlazni signal *output_enable* se aktivira i time se označava da je postavljen rezultujući heš na izlazne linije).

4.3. POSTAVKA I CILJEVI

U ovom radu biće prikazane tri implementacije BLAKE-256 Soft IP jezgara:

- Minimalna implementacija koja troši minimalne hardverske resurse (ista instanca runde sa samo jednom G funkcijom se koristi).
- Maksimalna implementacija koja ostvaruje visok protok tj. gde se svaka runda instancira zasebno, i gde postoje registri između rundi (radi omogućavanja rada u *pipeline* režimu ako bi hteli da omogućimo više istovremenih heširanja različitih poruka).
- Maksimalna implementacija u kojoj su sve runde implementirane kao jedna kombinaciona logika.

Pretpostavke prilikom realizacije:

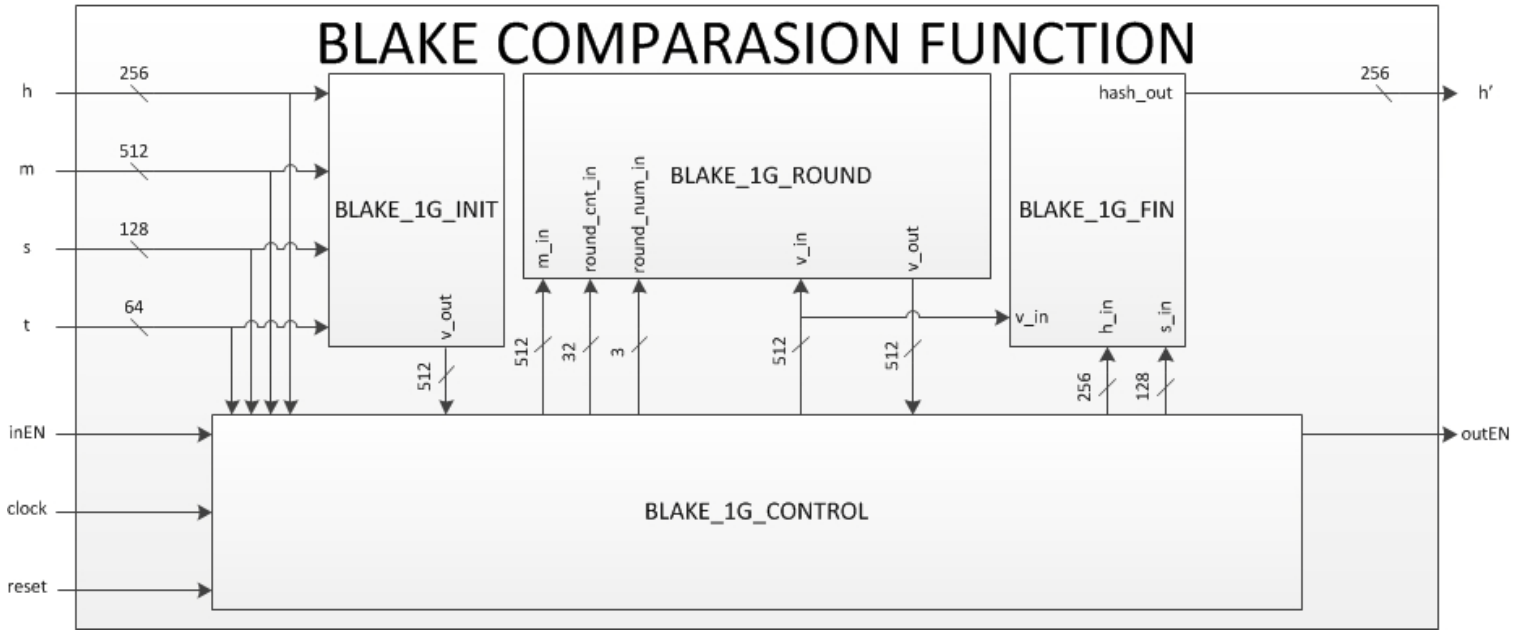
- Poruka se nalazi izvan IP jezgra u FIFO memoriji iz koje treba čitati blok po blok dužine 512 bita i procesirati u bloku za heširanje.
- Poruka je već prilagođena BLAKE algoritmu, urađen je eventualan *padding* i sl.
- Dužina poruke u bitima nalazi se u istoj FIFO memoriji kao i sama poruka, s tim što je dužina smeštena prva, a zatim ide poruka. Dužina poruke se koristi prilikom kreiranja vrednosti brojača radi određivanja kraja heširanja. Pri tome se smatra da poruka nije veća od 2^{64} bita.

4.4. BLAKE-256 IP JEZGRO - MINIMALNA IMPLEMENTACIJA

Minimalna implementacija IP BLAKE jezgra predstavlja dizajn IP jezgra koji bi trebalo da zauzima najmanju površinu tj. da iskorišćava najmanje resursa na čipu u odnosu na ostale implementacije.

Da bi se ovo ostvarilo sama runda se sastoji od samo jedne G funkcije, pa će se svaka runda izvršiti u 8 ciklusa signala takta. Ovim pristupum, heširanje jednog bloka poruke od 512 bita traje 112 ciklusa takta. Pritom samo jedna poruka može da se procesira istovremeno. Pošto se maksimalno koristi jedna instanca G funkcije, postiže se minimalna zauzetost resursa, ali i minimalan protok heširanja kao posledica ovog pristupa. Ova realizacija zahteva nešto kompleksniju kontrolnu logiku koja podrazumeva uvođenje mašine stanja koja će da broji runde, kao i stanja prilikom svake od rundi. Međutim, i pored ove dodatne kontrolne logike ušteda koja se ostvari implementacijom samo jedne instance G runde je značajna.

Kompletna šema funkcije kompresije za „Minimalnu“ implementaciju je prikazana na slici 4.4.1. Na njoj se jasno mogu videti inicijalni i finalizacioni kombinacioni blokovi, kao i kombinaciona logika runde koja se sastoji od samo jedne G funkcije. Svi registri, kao i mašina stanja nalaze se u kontrolnom bloku.



Slika 4.4.1. – Funkcija kompresije „Minimalne“ implementacije IP BLAKE jezgra

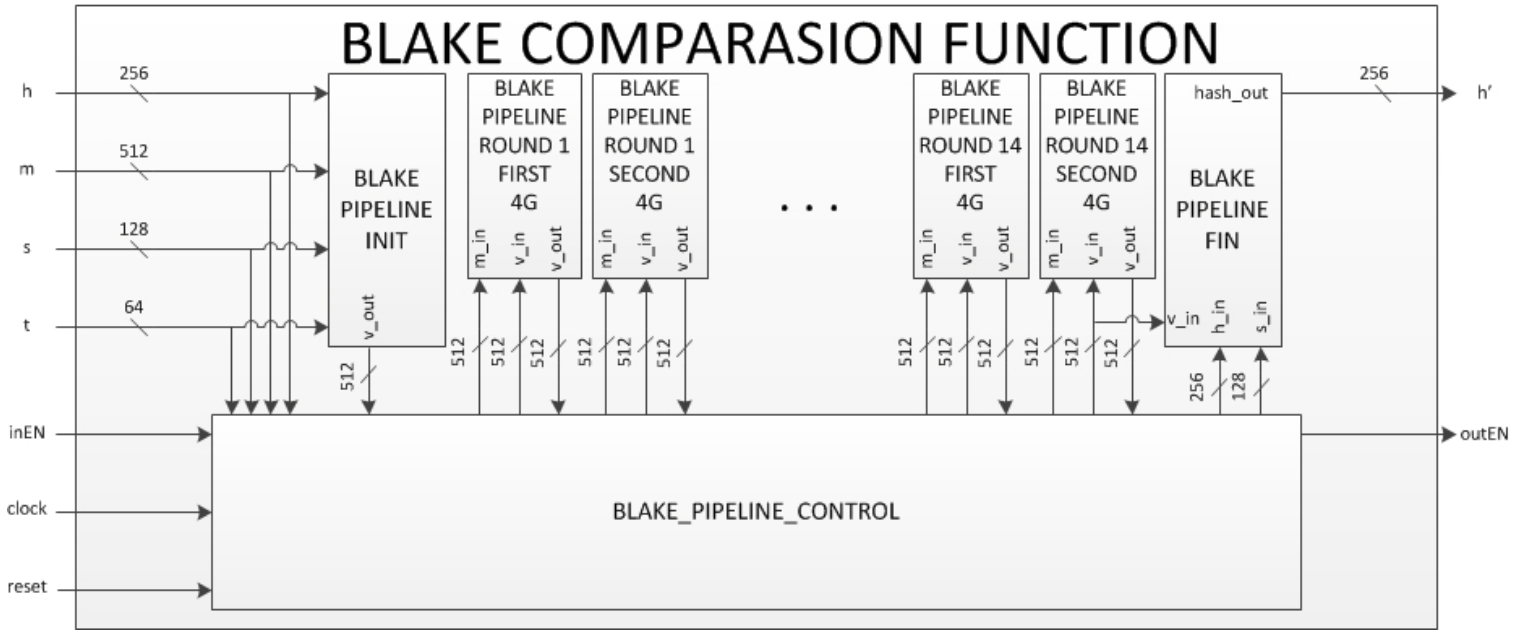
4.5. BLAKE-256 IP JEZGRO MAKSIMALNA REALIZACIJA (PAJPLAJN)

Pajplajn implementacija IP BLAKE jezgra predstavlja dizajn IP jezgra koji bi trebalo da zauzima najveću površinu čipa tj. da koristi najveću količinu resursa na čipu u odnosu na ostale dve implementacije realizovane u ovoj tezi. S druge strane, ovo IP jezgro bi trebalo da omogući visok stepen paralelizacije, a samim tim i visok protok. Takođe, pošto se koristi pajplajn tehnika, kombinaciona logika pojedinih etapa je manja, i samim tim bi trebala da se postigne viša radna frekvencija što takođe doprinosi većem protoku ove realizacije BLAKE heširanja.

Da bi se ostvarila što veća frekvencija signala takta, runda je podeljena na dve podrunde. Svaka podrunda se sastoji od po četiri G funkcije koje mogu međusobno da se izvršavaju u paraleli. Kao što sam BLAKE algoritam nalaže, postojaće 14 rundi. I za razliku od “Minimalne” realizacije kontrolni blok neće sadržati mašinu stanja već samo registre između faza pajplajna (registri su neophodni za kreiranje pajplajna). Stoga je sama kontrolna logika jednostavnija u odnosu na “Minimalnu” implementaciju.

Pajplajn implementacija istovremeno će moći da procesira 28 paralelnih tokova podataka i time omogućava visoku protočnost heširanja na nivou celog sistema. Pajplajn pristup omogućava da ovo IP jezgro postigne najveću radnu frekvenciju takta u odnosu na ostale implemantacije, ali i najveće paralelno procesiranje više podataka koji se heširaju. Takođe, kao posledica cepanja dizajna u 28 blokova tj. kreiranje 28 pajplajn faza znatno se povećava količina potrebin registara, pa će ova implementacija zahtevati znatno više resursa čipa u odnosu na druge dve realizacije opisane u ovoj tezi.

Kompletna šema funkcije kompresije za „Pipeline“ implementaciju je prikazana na slici 4.5.1. Na njoj se jasno mogu videti inicijalni i finalizacioni kombinacioni blokovi, kao i kombinaciona logika podrundi koje se sastoje od po četiri G funkcije.



Slika 4.5.1. – Funkcija kompresije „Pipeline“ implementacije IP BLAKE jezgra

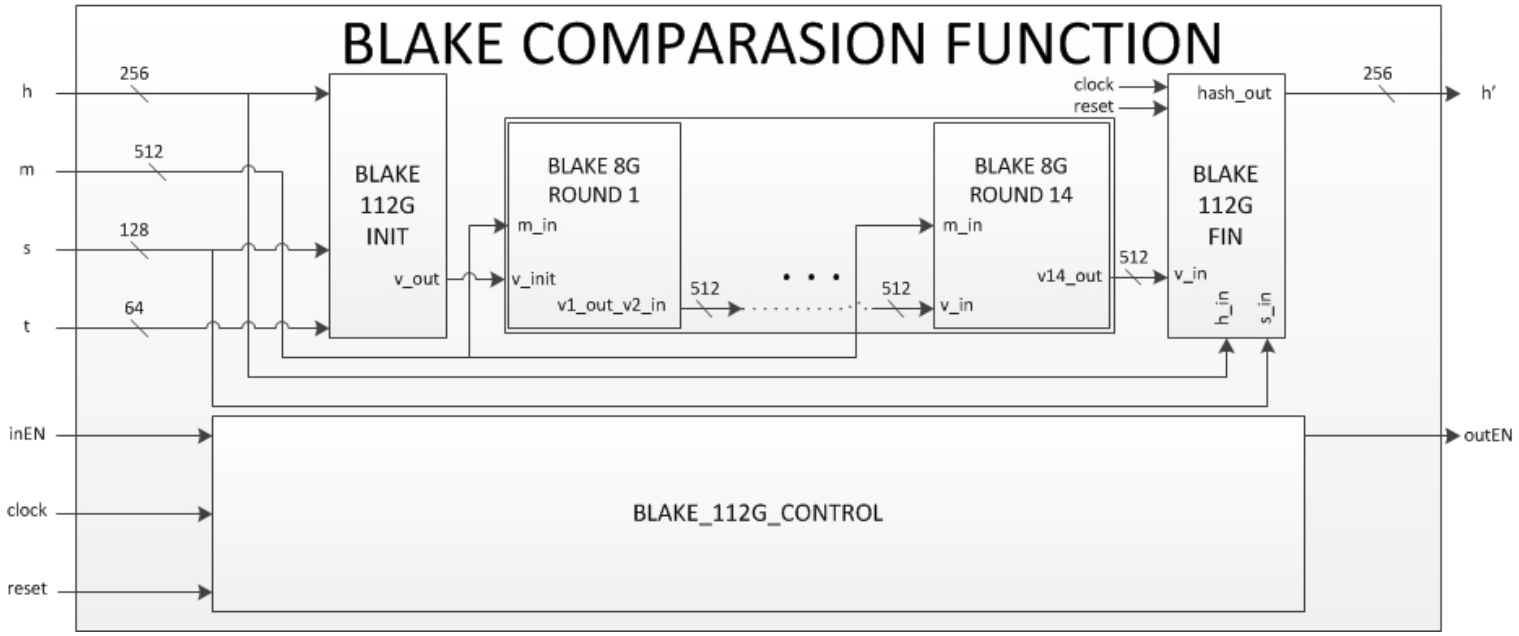
4.6. BLAKE-256 IP JEZGRO MAKSIMALNA REALIZACIJA (KOMBINACIONA LOGIKA)

Potpuno kombinaciona implementacija IP BLAKE jezgra predstavlja dizajn IP jezgra koji bi trebao da iskorišćava resurse na čipu između „Minimalne“ i „Pipeline“ implementacije.

Ovakva realizacija BLAKE algoritma zahteva da su sve runde, gde se svaka sastoji od 8 G funkcija, povezane na red i bez registrovanja podataka između tj. kao jedna velika kombinaciona logika. Podaci će biti registrovani samo na izlazu finalizacionog bloka. Posledica ovakve implementacije bi trebala da bude niska frekvencija signala takta kao i nemogućnost paralelizacije više tokova podataka jer je dizajn jedna velika kombinaciona logika. S druge strane, trebalo bi da se znatno smanji iskorišćenje resursa u odnosu na „Pipeline implementaciju“ sa stanovišta količine potrebnih registara, kao i da poveća brzina obrade podatka u odnosu na „Minimalnu“ implementaciju. Pošto se kompletno heširanje obavlja praktično u jednom taktu, latencija koju unosi ova realizacija je najmanja u odnosu na ostale dve realizacije opisane u ovoj tezi.

Kontrolna logika kod ove impementacije je najprostija u odnosu na prethodno opisane dve implementacije. Služi samo da zakasni tj. da registruje *enable* signal na izlazu.

Kompletna šema funkcije kompresije za „Potpuno kombinaciona“ implementaciju je prikazana na slici 4.6.1. Na njoj se jasno mogu videti inicijalni i finalizacioni kombinacioni blokovi, kao i kombinaciona logika bloka koji sadrži 14 rundi koje sadrže po 8G funkcija. Svo registrovanje obavlja se posle finalizacije.



Slika 4.6.1. – Funkcija kompresije „Potpuno kombinaciona“ implementacije IP BLAKE jezgra

5. FUNKCIONALNA VERIFIKACIJA BLAKE IP JEZGRA

Ovo poglavlje daje detaljan opis celokupne verifikacije IP jezgra BLAKE funkcije. Prikazuje tehnike verifikovanja kao i detalje koji su rađeni u svakom koraku. Opisuje alate koji su korišćeni, arhitekturu test okruženja, kao i indikatore koji govore da je verifikacija uspešno urađena.

5.1. UVOD

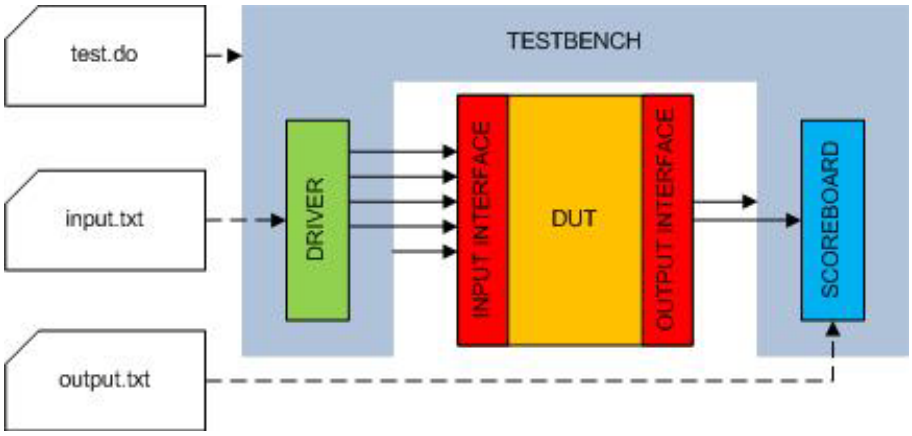
Verifikacija IP jezgra u ovom radu biće sprovedena kao direktna verifikacija sa konkretnim test scenarijima. Neće biti randomizacije jer sam dizajn modula ima jasne interfejse i stoga se lako mogu pokriti sve funkcionalnosti kao i moguće greške puštanjem jednostavnih test vektor vrednosti kroz dizajn.

5.2. TESTBENČ

Testbenč predstavlja logiku, napisanu u VHDL jeziku za opis hardvera, koja okružuje dizajn koji se testira i koja vrši određene pobude na ulazu u dizajn i prati da li izlazi imaju predviđeno ponašanje.

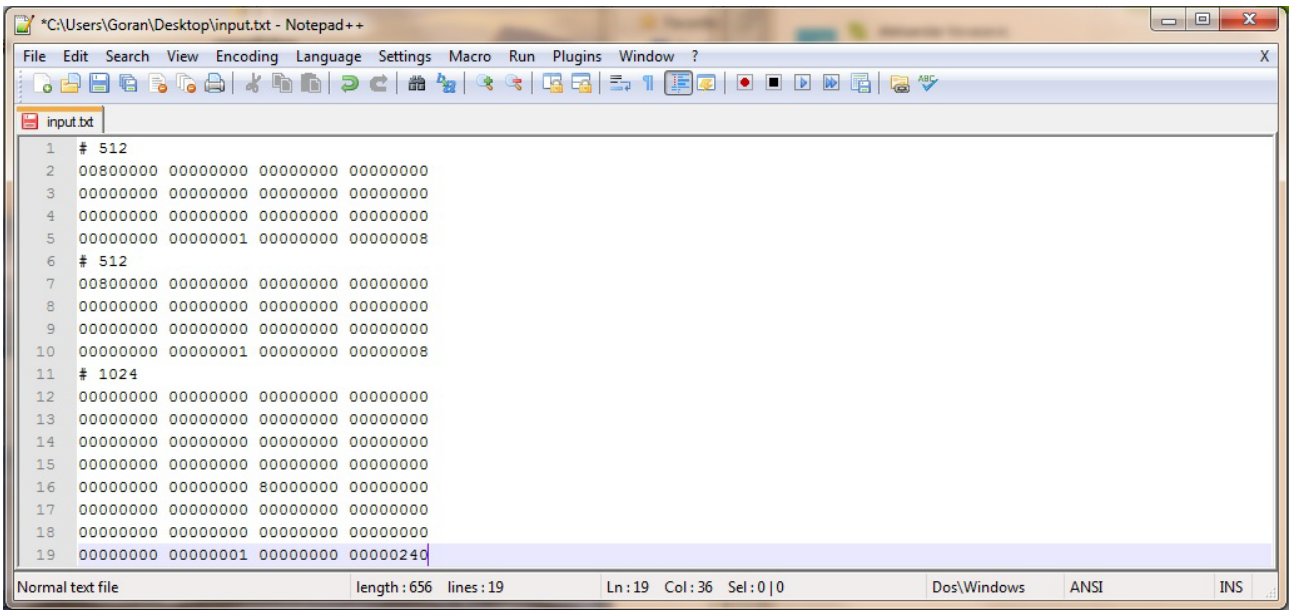
U ovom radu test okruženje čine dva glavna elementa *driver* i *scoreboard*. *Driver* je zadužen za dve stvari, konstantno čitanje sadržaja iz ulaznog fajla koji predstavlja poruke koje će biti heširane, raspoređivanje i slanje ka DUT-u (*design under test*). *Scoreboard* na početku izvršavanja otvara fajl sa predviđenim heš vrednostima koje bi trebalo DUT da generiše tokom trajanja testa, smešta ih u listu i prilikom svakog generisanja heša od strane dizajna poredi dobijenu vrednost sa odgovarajućom vrednošću iz liste.

Slika 5.2.1. detaljno prikazuje testbenč arhitekturu. Prikazuje komponente test okruženja kao i interfejse prema DUT-u.



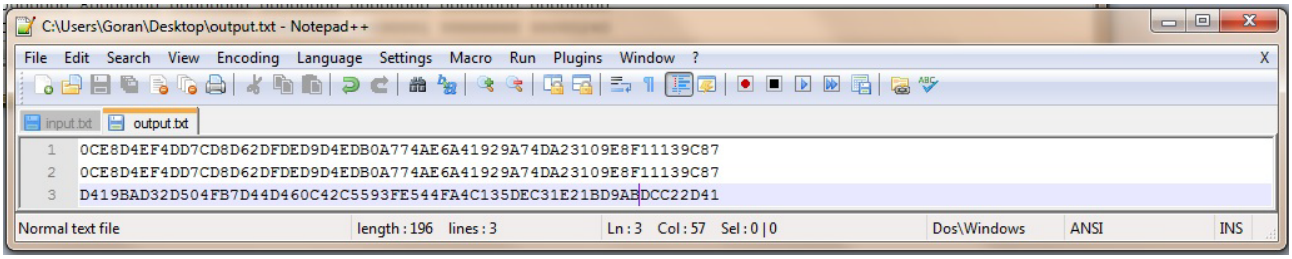
Slika 5.2.1. – Arhitektura test okruženja

Fajl koji *driver* koristi kao ulazni je „input.txt“. Ovaj fajl sadrži sekvence poruka koje trebaju biti prosleđene dizajnu. Takođe, pre svake poruke naveden je i broj korisnih bita u toj poruci koji DUT-u govori kako će slati samu poruku i iz koliko će iteracija to uraditi . Slika 5.2.2 prikazuje jednostavan primer jednog „input.txt“ fajla.



Slika 5.2.2. – Primer input.txt fajla

Fajl koji *scoreboard* koristi kao ulazni je „output.txt“. Ovaj fajl sadrži predviđene vrednosti BLAKE heš funkcije kompresije koje se očekuju da će ih dizajn izbaciti istim redosledom kako su i navedene u fajlu. Slika 5.2.3 prikazuje primer jednog „output.txt“ fajla sa predviđenim vrednostima koje bi DUT trebalo da generiše na prilikom heširanja poruka iz „input.txt“ fajla prikazanim na slici 5.2.3.

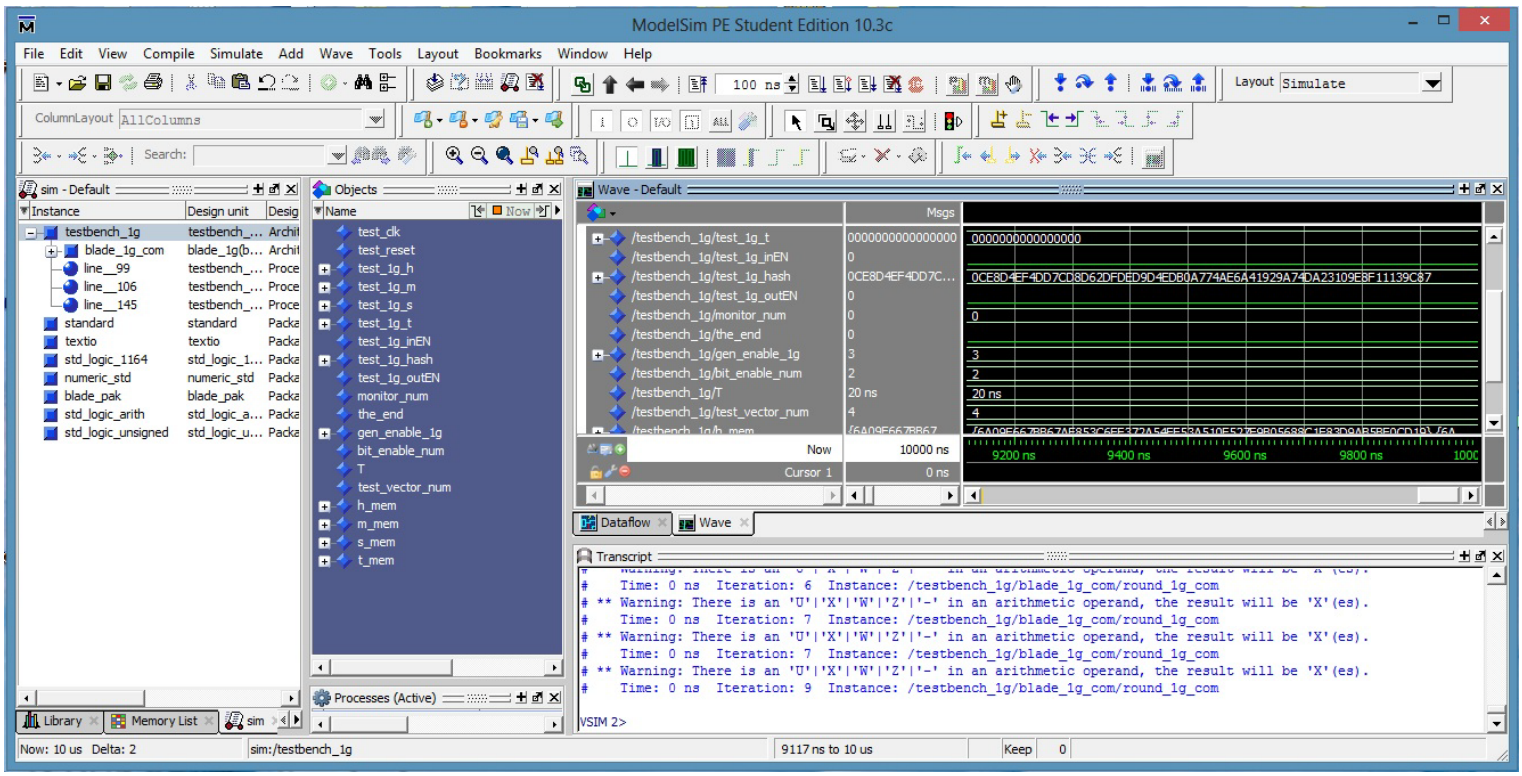


Slika 5.2.3. – Primer output.txt fajla

Za generisanje samih „input.txt“ i „output.txt“ fajlova koriste se hex editor i softverski referentni model BLAKE IP heš algoritma skinut sa sajta [7]. Kod ovog referentnog modela kao i skripte za njegovo pokretanje biće priloženi u digitalnom obliku uz sam rad.

5.3. ALATI

Alat koji se koristi za kompajliranje dizajna i test okruženja, simulaciju i debagovanje u ovom radu je ModelSim PE Student Edition 10.3c kompanije Mentor Graphics. Na slici 5.3.1 prikazan je grafički prikaz ModelSim-a.

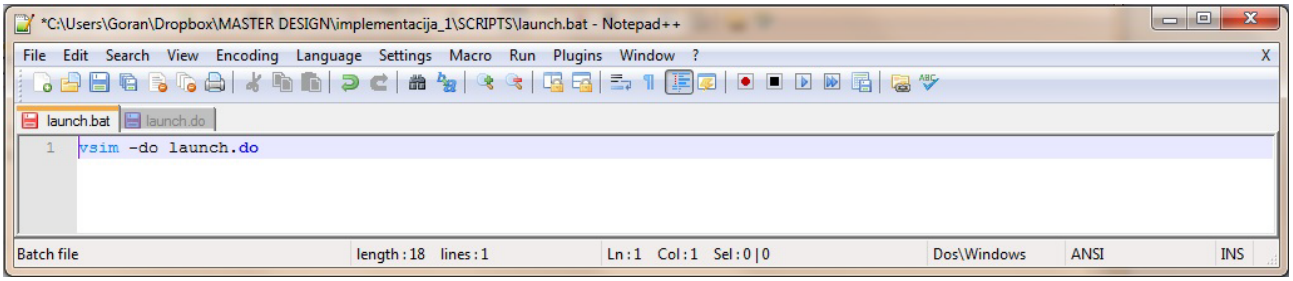


Slika 5.3.1. – Prikaz ModelSim alata

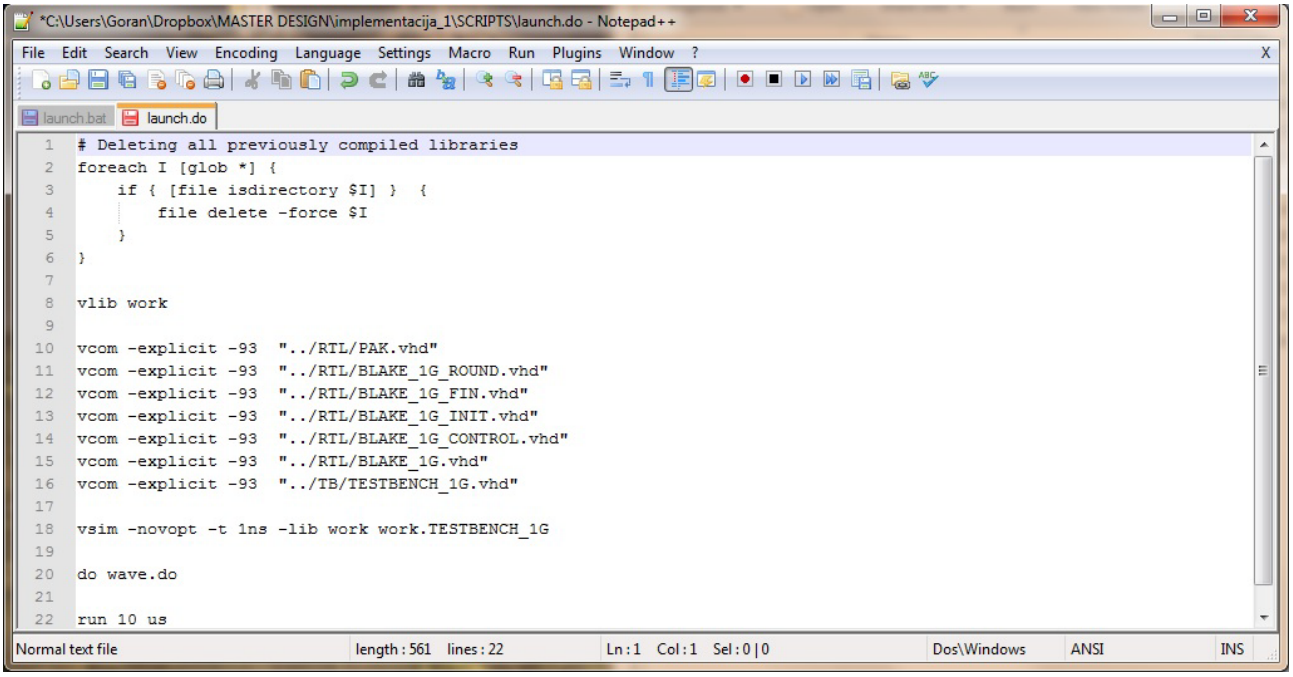
Samo pokretanje ModelSim-a kao i pravljenje projekta biće automatski rađeni prilikom pokretanja .bat fajla koji izvršava .do skriptu koja poziva i izvršava ModelSim-ove TCL skript komande. Detaljan opis ovih fajlova biće predstavljen u narednom potpoglavlju.

5.4. SIMULACIJA I PUŠTANJE TESTOVA

Kompajliranje, pokretanje ModelSim-a i prikazivanje vremeskih dijagrama signala kao i samo pokretanje simulacije radi se automatski selektovanjem „lounch.bat“ fajla. Ovaj fajl, čiji se sadržaj vidi na slici 5.4.1 pokreće „lounch.do“ skriptu. Skripta se sastoji od TCL komandi koje na početku brišu izgenerisane fajlove za vreme prethodnog kompajliranja, pokreće ModelSim, ubacuje sve dizajn i testbenč fajlove u projekat i kompajlira ih. Zatim ubacuje sve signale koje hoćemo da pratimo u vremenski dijagram signala i pokreće izvršavanje simulacije. Slika 5.4.2. prikazuje sadržaj ovog .do fajla.



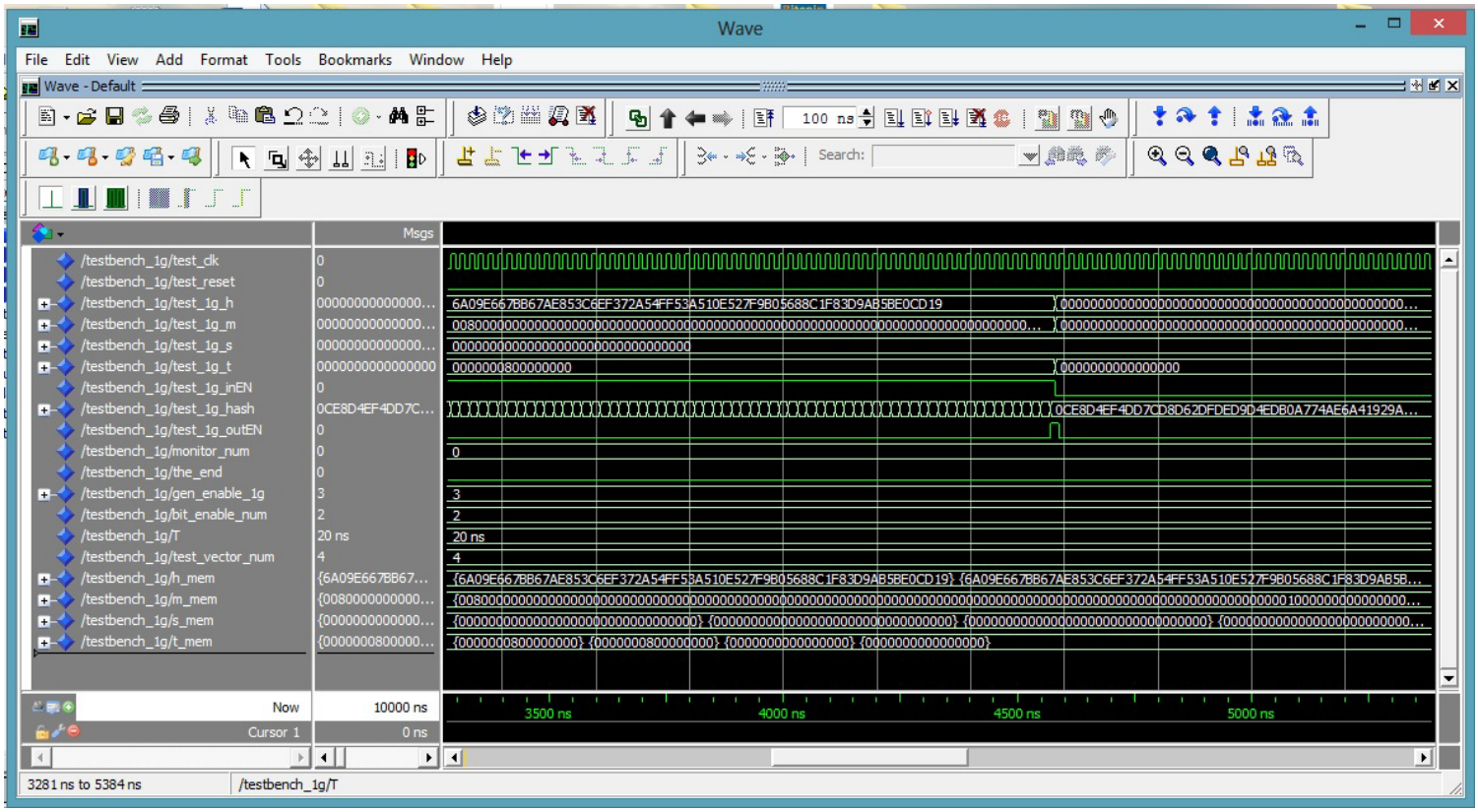
Slika 5.4.1. – Sadržaj lounch.bat fajla



Slika 5.4.2. – Sadržaj lounch.do fajla

Sam prikaz simulacije i grafički prikaz svakog signala može se videti u vremenskom dijagramu signala, slika 5.4.3. Na dijagramu se takođe mogu videti izlazne vrednosti realizovane BLAKE heš funkcije pa stoga sam dizajn možemo debagovati i vizuelno, ali takve metode oduzimaju puno vremena i nisu pouzdane pa je ipak sigurnije vršiti automatsko poređenje dobijenih heš vrednosti iz DUT-a sa predviđenim vrednostima u samom test okruženju. U slučaju greške u dizajnu *scorebord* će prekinuti simulaciju i u .log fajl će ispisati tip greške.

Verifikacija je urađena na prethodno opisani način i potvrđen je ispravan rad sve tri realizacije BLAKE-256 algoritma za heširanje.



Slika 5.4.3. – Vremenski dijagram signala u ModelSim-u

6. PERFORMANSE, ZAUZEĆE RESURSA, PROTOČNOST I VREME PROPAGACIJE SIGNALA NA ALTERA I XILINX FPGA PLATFORMAMA

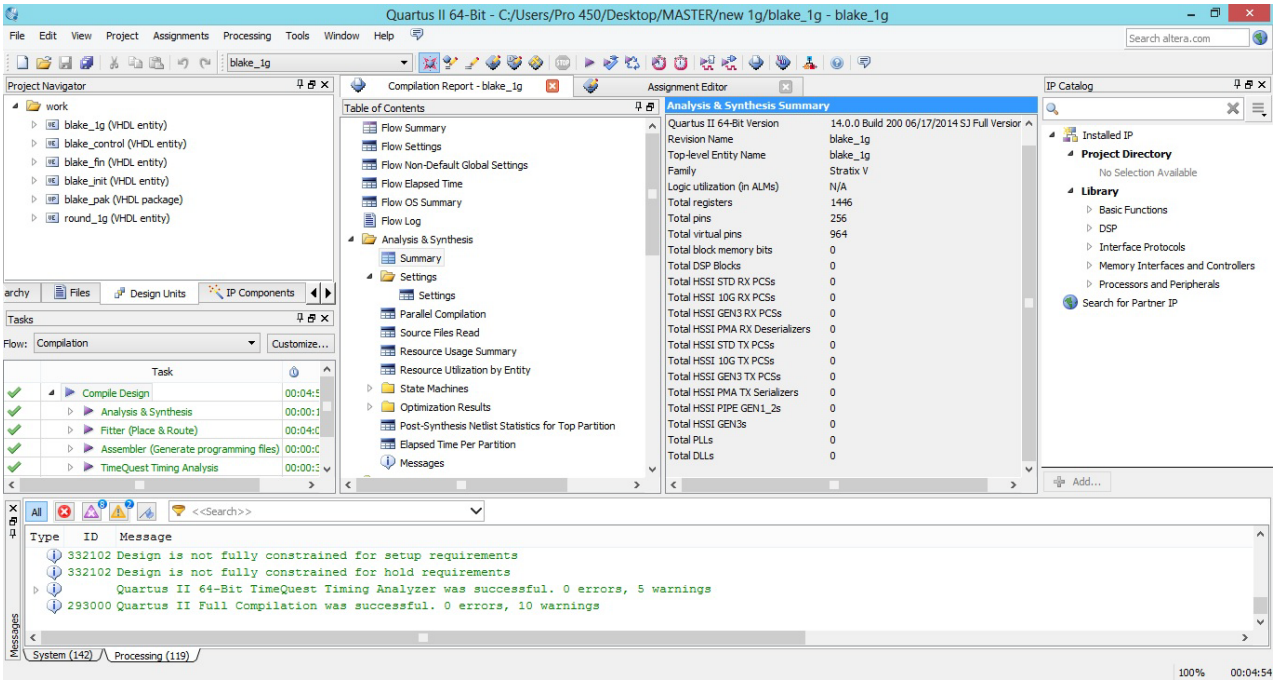
U ovom poglavlju biće prikazani rezultati analize i sinteze za Altera i Xilinx čipove. Biće tabelarno prikarane performanse u vidu maksimalne frekvencije i zauzeća resursa za svaku od IP implementacija jezgra koje su opisane u ovoj tezi. Na kraju poglavlja dato je tumačenje dobijenih rezultata kao i upoređivanje postignutih performansi sa predviđenim.

6.1. ALTERA I XILINX

Altera i Xilinx važe za apsolutne lidere među proizvođačima FPGA reprogramibilnih čipova u svetu. Stoga je u ovom radu fokus na testiranju performansi IP BLAKE jezgara upravo na njihov serijama čipova. Od Alterinih serija koristićemo Stratix III, i Cyclone II i III, dok ćemo od Xilinxovih serija koristiti Virtex-7 i Artix-7 serije čipova.

6.2. ALTERA QUARTUS II I REZULTATI SINTEZE

U naredne tri tabele prikazani su rezultati dobijeni iz fajlova koje je generisao Altera Quartus II alat posle procesa analize i sinteze (slika 6.2.2.1.). Svaka od tabela odgovara po jednoj od implementaciji. „Minimalna“, „Pipeline“ i „Potpuno kombinaciona“ implementacija odgovaraju tabelama Tabela 6.2.2.1, Tabela 6.2.2.2 i Tabela 6.2.2.3, respektivno.



Slika 6.2.1. – Quartus II alat

Čip	f_{max} (MHz)	Logika	Protok (Mb/s)
Cyclone II EP2C35F672C7	29.32	LUT (3104 - 9.3%) Regs (1062 - 3.4%)	134
Cyclone III EP3C80F780C7	39.97	LUT (3104 - 3.8%) Regs (1062 - 1.3%)	137
Stratix III EP3SL110F1152C3	76.88	LUT (3104 - 2.4%) Regs (1065 - 1.2%)	351

Tabela 6.2.1. – Rezultati analize i sinteze za „Minimalnu“ implementaciju BLAKE IP jezgra (Altera)

Čip	f_{max} (MHz)	Logika	Protok (Mb/s)
Cyclone II EP2C35F672C7	n/a	n/a	n/a
Cyclone III EP3C80F780C7	41.23	LUT (35883 - 44.2%) Regs (22589 - 27.8%)	21110
Stratix III EP3SL110F1152C3	96.00	LUT (25425 - 29.6%) Regs (22625 - 26.3%)	49152

Tabela 6.2.2. – Rezultati analize i sinteze za „Pipeline“ implementaciju BLAKE IP jezgra (Altera)

Čip	f_{max} (MHz)	Logika	Protok (Mb/s)
Cyclone II EP2C35F672C7	n/a	n/a	n/a
Cyclone III EP3C80F780C7	n/a	n/a	n/a
Stratix III EP3SL110F1152C3	3.44	LUT (29783 - 34.6%) Regs (613 - 0.7%)	1761

Tabela 6.2.3. – Rezultati analize i sinteze za „Potpuno kombinacionu“ implementaciju BLAKE IP jezgra (Altera)

6.3. XILINX ISE I REZULTATI SINTEZE

U naredne tri tabele prikazani su rezultati dobijeni iz fajlova koje je generisao Xilinx ISE alat posle procesa analize i sinteze. Svaka od tabela odgovara po jednoj od implementacija. „Minimalna“, „Pipeline“ i „Potpuno kombinaciona“ implementacija odgovaraju tabelama Tabela 6.3.2.1, Tabela 6.3.2.2. i Tabela 6.3.2.3, respektivno.

Čip	f_{max} (MHz)	Logika	Protok (Mb/s)
Artix 7 xc7a100t-3-csg324	91.98	LUT (3854 - 6%) Regs (1883 - 1%)	420
Virtex 7 xc7vx330t-2-ffg1157	112.34	LUT (3853 - 1%) Regs (1883 - 0,2%)	514

Tabela 6.3.1. – Rezultati analize i sinteze za „Minimalnu“ implementaciju BLAKE IP jezgra (Xilinx)

Čip	f_{max} (MHz)	Logika	Protok (Mb/s)
Artix 7 xc7a100t-3-csg324	120.32	LUT (35592 - 56%) Regs (29635 - 23%)	61605
Virtex 7 xc7vx330t-2-ffg1157	149.11	LUT (35592 - 17%) Regs (29636 - 7%)	76344

Tabela 6.3.2. – rezultati sinteze za „Pipeline“ implementaciju BLAKE IP jezgra (Xilinx)

Čip	f_{max} (MHz)	Logika	Protok (Mb/s)
Artix 7 xc7a100t-3-csg324	4.84	LUT (35893 - 56%) Regs (1147 - 0,5%)	2478.1
Virtex 7 xc7vx330t-2-ffg1157	5.89	LUT (35896 - 17%) Regs (1171 - 0,2%)	3014.1

Tabela 6.3.3. – Rezultati analize i sinteze za „Potpuno kombinacionu“ implementaciju BLAKE IP jezgra (Xilinx)

6.4. TUMAČENJE REZULTATA

Na osnovu dobijenih rezultata prikazanih u tabelama potpoglavlja 6.2 i 6.3, i izračunatog protoka heširanja poruka izvedeni su sledeći zaključci.

Od serije čipa prvenstveno zavisi maksimalna frekvencija dizajna koja zavisi i od veličine čipa, ali i tipa familije. Skuplje familije (Stratix kod Altere, Virtex kod Xilinxa) omogućavaju bolje performanse u odnosu na jeftinije familije (Cyclone kod Altere, Artix kod Xilinxa (u ranijim generacijama to je bila Spartan familija)). Naravno u okviru svake familije postoji spektar čipova od onih sa najnižim performansama do onih sa najvišim performansama što takođe utiče na krajnje perfromanse dizajna.

Dobijeni rezultati su u skladu sa očekivanjima koja su iznesena ranije u tekstu. „Minimalna“ implementacija postiže minimalne hardverske resurse, ali i minimalni protok heširanja poruke jer heširanje jednog bloka traje više taktova (112 taktova), a nije omogućeno paralelno procesiranje (heširanje) više poruka. „Pipeline“ implementacija zahteva najveće hardverske resurse, ali i postiže najveći protok heširanja zahvaljujući pajplajn strukturi - u tabelama 6.2.2 i 6.3.2 je prikazan protok kad je pajplajn maksimalno iskorišćen. „Potpuno kombinaciona“ varijanta troši minimalnu količinu registara, ali isto tako je radna frekvencija minimalna zbog ogromne kombinacione logike tj. kašnjenja kroz nju. Međutim, pošto se jedan blok poruke procesira u jednom taktu ostvaruje se i dalje visok protok heširanja. Napomenimo da su postignuti protoci heširanja veoma visoki, a da pri tome potrošnja hardverskih resursa nije enormna. Upotrebom jačih čipova bi mogli da se postignu i viši protoci heširanja.

7. ZAKLJUČAK

Rad je prikazao specifikaciju BLAKE-256 algoritma za heširanje, njegove moguće implementacije, kao i performanse na različitim FPGA platformama. U okviru rada su razmatrane tri različite implementacije, a izbor implementacije treba da zavisi od željenih performansi – minimalni resursi, ali i minimalan protok heširanja, maksimalni resursi, ali i maksimalan protok heširanja ili balans između zahtevanih resursa i protoka heširanja.

Iako nije izabran za novi SHA-3 standard, BLAKE i dalje predstavlja veoma jak algoritam za heširanje sa puno potencijala.

Blakecoin, nova virtuelna kripto valuta, koja omogućava plaćanje i transakcije preko Interneta širom sveta bez ograničenja i provizije, je prva na svetu kripto valuta koja koristi BLAKE-256 algoritam za heširanje kao njenu PoW (*Proof to Work*) funkciju.

Svojstva BLAKE-256 algoritma se mnogo bolje koriste u radu sa kripto valutama nego kod Keccak (pobednik konkursa za SHA-3 standard) ili drugih heš algoritama. Ne zahteva mnogo resursa i omogućava rad, bez gubitka performansi, u pozadini za vreme drugih operacija na računaru. Takođe je energetski efikasniji za 5 do 15 procenata u odnosu na druge heš algoritme.

BLAKE je takođe u međuvremenu dobio i svog naslednika. BLAKE2 algoritam predstavlja poboljšan BLAKE algoritam koji koristi manje resursa, brži je od MD5 algoritma, koristi minimalni *padding* itd.

LITERATURA

[1] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.-W. Phan, "SHA-3 proposal BLAKE ", *version 1.4*, 2011.

[2] Jian Guo1, Pierre Karpman, Ivica Nikolic, LeiWang1, ShuangWu, "Analysis of BLAKE2", *Nanyang Technological University, Singapore, École normale supérieure de Rennes, France*, 2013.

[3] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, Christian Winnerlein, " BLAKE2: simpler, smaller, fast as MD5" , 2013.

[4] Roger Woods, John McAllister, Gaye Lightbody, Ying Yi " FPGA-based Implementation of Signal Processing Systems", *John Wiley & Sons*, 2008.

[5] Bruce Schneier, " Primenjena kriptografija" *Mikro knjiga*, 2007.

[6] Mostafa Taha, Virginia Tech, " The birth of a new hashing standard: SHA-3", *EE Timess Journal*, 2010.

[7] *BLAKE Software implementations* [Online]. Available: https://131002.net/blake/

[8] Timothy Stapko, "Cryptography for embedded systems", *EE Timess Journal*, 2010.

[9] *CAST ALTERA SHA-256 Secure Hash Function Megafunction* [Online]. Available: http://www.cast-inc.com/ip-cores/encryption/sha-256/cast_sha256-a.pdf

[10] *Blakecoin (BLC) alternative cryptocoin – First coin to use Blake-256 Hashing Function* [Online]. Available: http://blog.gryfencryp.to/2014/03/24/blakecoin-first-blake-256-coin

[11] Vaibhav Doshi, Richa Arya, Rajesh Kumar Yadav, " FPGA Based Area And Throughput Implementation of JH And BLAKE Hash Function ", *International Journal of Computer Trends and Technology*, 2012.

[12] Jean-Luc Beuchat Eiji Okamoto, Teppei Yamazaki, "Compact Implementations of BLAKE-32 and BLAKE-64 on FPGA", *Proc. of IEEE Graduate School of Systems and Information Engineering University of Tsukuba, Tsukuba, Ibaraki, Japan*, 2010.

[13] *Introduction to Blakecoin* [Online]. Available: http://www.blakecoin.org