Slovenská technická univerzita v Bratislave
Fakulta elektrotechniky a informatiky
Katedra informatiky a výpočtovej techniky

Implementácia blokového šifrátora pomocou PLD

Tímový projekt

team č. 12:

Bc. Martin Prokša
Bc. Viliam Otepka
Bc. Ivan Varga
Bc. Martin Zeman

vedúci projektu: doc. Ing. Ladislav Hudec, CS
šk. rok: 2002/2003

I Dokumentácia k projektu

Obsah

Obsah I-1

I.1 Úvod I-2

I.1.1 Rozsah a ciele I-2

I.1.2 Prehľad dokumentu I-2

I.2 Analýza problému I-4

I.2.1 Princípy šifry IDEA I-4

I.2.2 IDEA šifrovanie I-4

I.2.3 IDEA dešifrovanie I-6

I.2.4 IDEA testovanie I-7

I.3 Špecifikácia riešenia I-10

I.3.1 Špecifikácia požiadaviek interaktívnej prezentácie I-10

I.3.2 Špecifikácia požiadaviek šifrátora IDEA I-10

I.4 Návrh riešenia interaktívnej prezentácie I-11

I.4.1 Návrh interaktívnej prezentácie I-11

I.4.2 Výber implementačného prostredia interaktívnej prezentácie I-11

I.4.3 Návrh grafickej časti prezentácie šifry IDEA I-12

I.5 Implementácia I-14

I.5.1 Implementácia grafického rozhrania prezentácie I-14

I.5.2 Implementácia algoritmu prezentácie I-15

I.6 Návrh a Implementácia konečného produktu I-16

I.6.1 BIST I-16

I.6.2 Šifrátor IDEA I-17

I.6.3 Návrh jedného kola šifry I-17

I.6.4 Návrh výstupnej transformácie I-18

I.6.5 Návrh úpravy násobenia pre šifru IDEA I-19
I.6.6 XOR I-20
I.6.7 Správa kľúčov I-21
I.6.8 Návrh generovania šifrovacích kľúčov I-21
I.6.9 Návrh generovania dešifrovacích kľúčov I-22
I.7 Testovanie I-28
I.7.1 Softvérová metóda I-28
I.7.2 Hardvérové testovanie I-29
I.8 Používateľská príručka I-30
I.8.1 Rozhranie šifrátora I-30
I.8.2 Režimy činnosti I-32
I.8.3 Porovnanie priepustnosti I-35
I.8.4 Používateľská príručka programu IdeaTest I-36
I.9 Literatúra I-41

I.1 Úvod

Tento projekt sa zaoberá problematikou šifrovania, prezentácie šifrovacieho algoritmu ako aj samotný návrh hardvérovej implementácie šifry. V tejto časti dokumentu sa budeme venovať iba problematike šifrovania a jej prezentácie.

I.1.1 Rozsah a ciele

Vzhľadom na to, že daná problematika je obšírna je práca na tímovom projekte je rozdelená do dvoch častí. Úlohou zimného semestra je vytvoriť grafickú prezentáciu šifrovania pomocou šifry IDEA [4] a vysvetlenie tejto šifry. Výstupy z tejto časti budú uverejnené na tímovej web stránke. Cieľom vytvárania prezentácie je oboznámiť sa s princípmi šifry IDEA, ako aj priblížiť túto šifru ostatným záujemcom ktorý nemajú žiadne vedomosti z oblasti šifrovania.

V letnom semestri bude naša práca pozostávať z navrhnutia, implementácia a simulácie návrhu šifry IDEA do PLD programovateľného obvodu.

Cieľom tejto časti dokumentu je analyzovať, špecifikovať a navrhnúť interaktívnu prezentáciu šifry IDEA. Budeme sa venovať analýze a návrhu implementácie z pohľadu hardvérovo nezávislých funkcií obvodu, ako napríklad zhodnotenie rôznych prístupov k implementácií matematicko-logických operácií.

I.1.2 Prehľad dokumentu

Tento dokument sa stane súčasťou dokumentácie k tímovému projektu. Dokumenty, ktoré budú tvoriť ucelený celok budú označené rímskymi číslicami.

V tomto dokumente sa nachádzajú špecifikácie požiadaviek na

interaktívnu prezentáciu šifry IDEA. Dokument analyzuje problém šifrovania pomocou šifry IDEA a návrh riešenia interaktívnej prezentácie šifry IDEA.

I.2 Analýza problému

I.2.1 Princípy šifry IDEA

IDEA (International Data Encryption Algorithm) je názov patentovaného a univerzálne aplikovateľného blokového šifrovacieho algoritmu. Používa rovnaký kľúč (tajný kľúč) na šifrovanie aj dešifrovanie. S kľúčom veľkosti 128 bitov, IDEA je ďaleko bezpečnejšia ako široko známy DES (Data Encryption Standard). Mnohí považujú tento algoritmus za najbezpečnejší blokový šifrovací algoritmus týchto dní. Algoritmus bol vytvorený v rámci projektu v spolupráci so Švajčiarskym federálnym technologickým inštitútom a spoločnosťou Ascom [4]. Hlavným cieľom tohoto projektu bolo vytvoriť silný šifrovací algoritmus, ktorý by mal nahradiť DES algoritmus.

Výhody algoritmu:

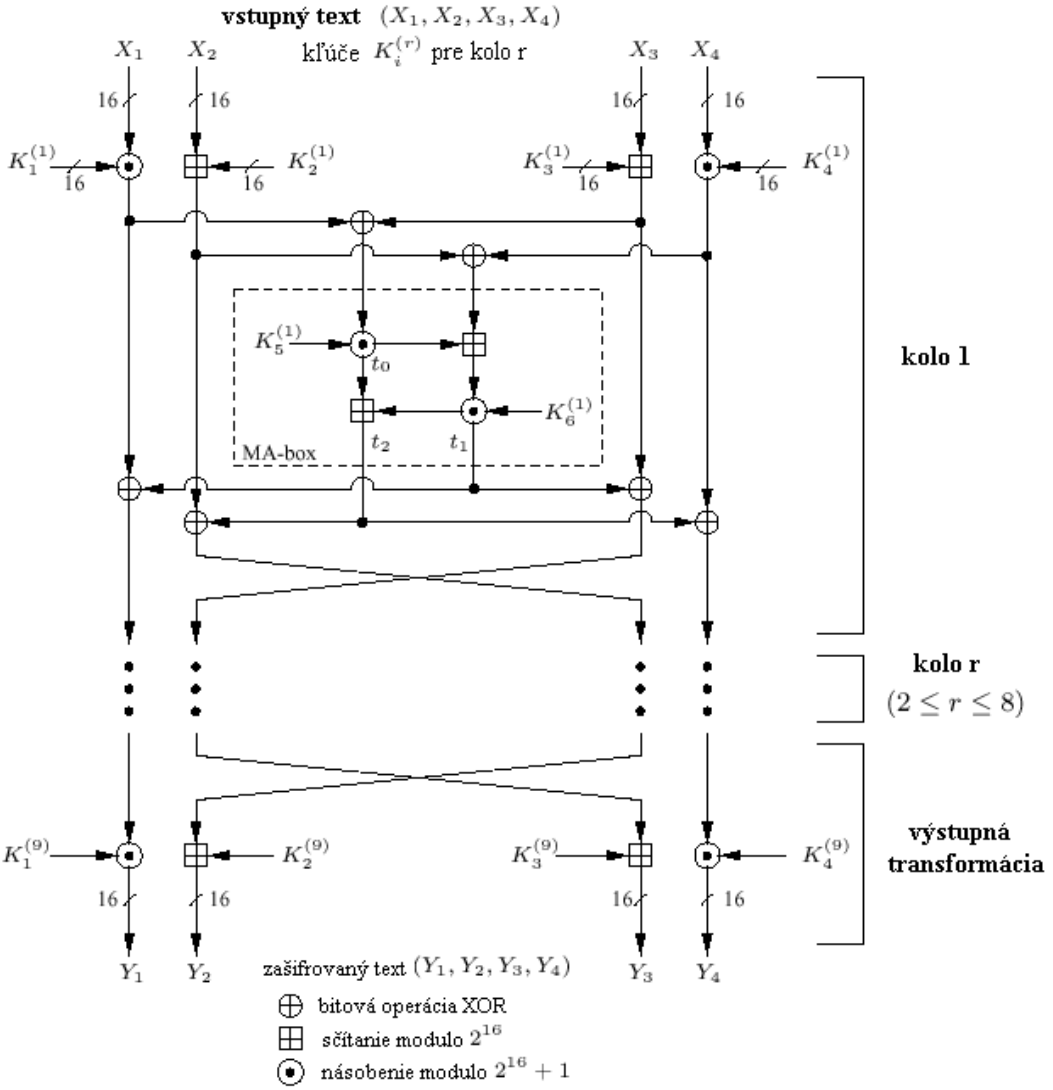
- poskytuje vysoký stupeň ochrany
- je úplne špecifikovaný a jednoducho pochopiteľný
- prístupný pre všetkých
- použiteľný v širokom rozsahu aplikácií
- môže byť ekonomicky implementovateľný do elektronických komponentov (VLSI čip)
- môže byť využitý efektívne
- je silný, malý a rýchly

I.2.2 IDEA šifrovanie

IDEA šifruje 64 bitový vstupný text na 64 bitový zašifrovaný blok, použitím 128 bitového kľúča. Proces šifrovania pozostáva z ôsmich výpočtovo identických kôl a výstupnej transformácie [1]. Výstup z nejakého kola je

vstupom pre nasledujúce kolo. Výstup z ôsmeho kola je vstupom výstupnej transformácie, a jej výstupom je už požadovaný zašifrovaný blok.

Na obrázku č. I.1 sú zobrazené bloky šifry IDEA. Na začiatku je rozkreslené prvé kolo šifrovania. Vstupy X1,X2,X3,X4 sú nezašifrovaný vstupný text rozdelený do štyroch 16 bitových slov. Na konci vystupuje zašifrovaný blok rozdelený do štyroch 16 bitových slov Y1,Y2,Y3,Y4 .



obr č.I.1 Bloky šifry IDEA

Ako je zrejmé z obrázku, do každého kola vstupuje sada šiestich kľúčov veľkosti 16 bitov. Pre osem kôl teda potrebujeme 48 kľúčov. Výstupná transformácia potrebuje naviac štyri kľúče. Spolu je teda potrebné vygenerovať 52 kľúčov.

Tieto šifrovacie kľúče sa generujú zo vstupného 128 bitového kľúča nasledovným spôsobom:

1. 128 bitový vstupný kľúč je rozdelený na osem 16 bitových kľúčov – prvých osem kľúčov
2. 128 bitový kľúč je cyklicky bitovo posunutý vľavo o 25 bitov a znovu rozdelený na osem ďalších 16 bitových kľúčov
3. druhý krok sa opakuje dovtedy, pokiaľ nie je vygenerovaných všetkých 52 kľúčov

Ako je zrejmé z obrázka, algoritmus využíva tri základné operácie:

- bitová operácia XOR – dva 16 bit vstupy, jeden 16 bit výstup
- sčítanie modulo 2^{16} - dva 16 bit vstupy, jeden 16 bit výstup
- násobenie modulo $2^{16} + 1$ (nula je interpretovaná ako 2^{16}) - dva 16 bit vstupy, jeden 16 bit výstup – algoritmus je možné nájsť v [2]

I.2.3 IDEA dešifrovanie

Dešifrovanie využíva rovnaký algoritmus ako šifrovanie, pričom vstup je zašifrovaný 64 bitový blok a výstup je dešifrovaný text. Rozdiel je v tom, že sa použije 52 dešifrovacích kľúčov, ktoré sú odvodené od 52 šifrovacích kľúčov. Transformácia šifrovacích kľúčov na dešifrovacie určuje nasledovná tabuľka:

kolo r	$K'^{(r)}_1$	$K'^{(r)}_2$	$K'^{(r)}_3$	$K'^{(r)}_4$	$K'^{(r)}_5$	$K'^{(r)}_6$
$r = 1$	$(K^{(10-r)}_1)^{-1}$	$-K^{(10-r)}_2$	$-K^{(10-r)}_3$	$(K^{(10-r)}_4)^{-1}$	$K^{(9-r)}_5$	$K^{(9-r)}_6$
$2 \leq r \leq 8$	$(K^{(10-r)}_1)^{-1}$	$-K^{(10-r)}_3$	$-K^{(10-r)}_2$	$(K^{(10-r)}_4)^{-1}$	$K^{(9-r)}_5$	$K^{(9-r)}_6$
$r = 9$	$(K^{(10-r)}_1)^{-1}$	$-K^{(10-r)}_2$	$-K^{(10-r)}_3$	$(K^{(10-r)}_4)^{-1}$	—	—

obr č.I.2 Transformácia šifrovacích kľúčov na dešifrovacie

Popis indexov k obrázku č.I.2:

r – číslo kola

K – šifrovacie kľúče

K^I – dešifrovacie kľúče

$-K_i$ – inverzné sčítanie (modulo 2^{16}) kľúča K_i

K_i^{-1} – inverzné násobenie (modulo $2^{16} + 1$) kľúča K_i

Inverzné násobenie sa dá odvodiť z rozšíreného Euklidovho algoritmu (algoritmus 2.107 v [1]).

I.2.4 IDEA testovanie

Na testovanie implementovaného algoritmu použijeme testovacie vektory získané z [1]. Ďalším spôsobom testovania môže byť testovanie s náhodne generovaným kľúčom a vstupom. Vstup sa zašifruje a následne spätne dešifruje a porovná sa či nastala zhoda. Ak áno, algoritmus je pre tieto vstupy funkčný. Ak nie, algoritmus je zle implementovaný.

Počítame s obidvoma spôsobmi testovania, pričom pri náhodne generovaných vstupoch chceme testovať algoritmus opakovane, teda pre viacero náhodne generovaných vstupov (zhruba stotisíc opakovaní).

Testovacie vektory pre šifrovanie:

128-bit $K = (1,2,3,4,5,6,7,8)$							64-bit $M = (0,1,2,3)$			
r	$K_1^{(r)}$	$K_2^{(r)}$	$K_3^{(r)}$	$K_4^{(r)}$	$K_5^{(r)}$	$K_6^{(r)}$	X_1	X_2	X_3	X_4
1	0001	0002	0003	0004	0005	0006	00f0	00f5	010a	0105
2	0007	0008	0400	0600	0800	0a00	222f	21b5	f45e	e959
3	0c00	0e00	1000	0200	0010	0014	0f86	39be	8ee8	1173
4	0018	001c	0020	0004	0008	000c	57df	ac58	c65b	ba4d
5	2800	3000	3800	4000	0800	1000	8e81	ba9c	f77f	3a4a
6	1800	2000	0070	0080	0010	0020	6942	9409	e21b	1c64
7	0030	0040	0050	0060	0000	2000	99d0	c7f6	5331	620e
8	4000	6000	8000	a000	c000	e001	0a24	0098	ec6b	4925
9	0080	00c0	0100	0140	—	—	11fb	ed2b	0198	6de5

obr. č.I.3 IDEA testovacie vektory pre šifrovanie

Všetky údaje v tabuľke sú v *hexadecimálnom* (číselná sústava zo základom 16) tvare veľkosti 16 bitov (štyri znaky *hexa*). Vstupom je 128 bitový kľúč K = (1,2,3,4,5,6,7,8) a vstupný 64 bitový text M = (0,1,2,3).

Sú zobrazené všetky šifrovacie kľúče a výstup pre každé kolo algoritmu a takisto po výstupnej transformácii (r=9), čo je už zašifrovaný blok.

Testovacie vektory pre dešifrovanie:

$K = (1,2,3,4,5,6,7,8)$							$C = (11fb,ed2b,0198,6de5)$			
r	$K'^{(r)}_1$	$K'^{(r)}_2$	$K'^{(r)}_3$	$K'^{(r)}_4$	$K'^{(r)}_5$	$K'^{(r)}_6$	X_1	X_2	X_3	X_4
1	fe01	ff40	ff00	659a	c000	e001	d98d	d331	27f6	82b8
2	fffd	8000	a000	cccc	0000	2000	bc4d	e26b	9449	a576
3	a556	ffb0	ffc0	52ab	0010	0020	0aa4	f7ef	da9c	24e3
4	554b	ff90	e000	fe01	0800	1000	ca46	fe5b	dc58	116d
5	332d	c800	d000	fffd	0008	000c	748f	8f08	39da	45cc
6	4aab	ffe0	ffe4	c001	0010	0014	3266	045e	2fb5	b02e
7	aa96	f000	f200	ff81	0800	0a00	0690	050a	00fd	1dfa
8	4925	fc00	fff8	552b	0005	0006	0000	0005	0003	000c
9	0001	fffe	fffd	c001	—	—	0000	0001	0002	0003

obr. č.I.3 IDEA testovacie vektory pre dešifrovanie

Aj v tejto tabuľke sú všetky údaje v *hexadecimálnom* tvare veľkosti 16 bitov (štyri znaky *hexa*). Vstupom je 128 bitový kľúč K = (1,2,3,4,5,6,7,8) a vstupný 64 bitový zašifrovaný blok C = (11fb,ed2b,0198,6de5).

Sú zobrazené všetky dešifrovacie kľúče a výstup pre každé kolo algoritmu a takisto po výstupnej transformácii (r=9), čo je vlastne dešifrovaný text.

I.3 Špecifikácia riešenia

I.3.1 Špecifikácia požiadaviek interaktívnej prezentácie

Požiadavky na interaktívnu prezentáciu sme konzultovali s vedúcim tímového projektu doc. Ing. Ladislavom Hudecom.

Interaktívna prezentácia má slúžiť na to, aby laik oboznámený so základmi logických systémov pochopil systém šifrovania pomocou šifry IDEA. Ide o interaktívnu prezentáciu, teda používateľ si má možnosť zadať svoje dáta a šifrovací kľúč. Dáta sa zadávajú buď ako nešifrované a program sa použije na ich zašifrovanie, alebo zadá šifrované dáta a tieto budú dešifrované.

Pri prezentovaní konkrétnych hodnôt sa vždy jedná o dáta vypočítané z vstupu od používateľa.

Celý postup šifrovania spolu s vysvetlením matematicko-logických funkcií bude umiestnený na tímovej web stránke spolu s interaktívnou prezentáciou.

I.3.2 Špecifikácia požiadaviek šifrátora IDEA

Šifrovací algoritmus IDEA treba opísať pomocou jazyka VHDL, aby bolo možné syntetizovať ho do štruktúry PLD obvodu. Pri návrhu bude zohľadnená veľkosť najväčšieho PLD obvodu podporovaného vývojovým prostriedkom Web Pack Xilinx [7].

Hlavným kritérium návrhu je optimalizácia na najväčšiu možnú rýchlosť. Obvod by mal byť navrhnutý tak, aby dosahoval maximálnu dátovú priepustnosť šifrovania. Na overenie funkčnosti návrhu je potrebné spraviť funkčnú simuláciu v simulátore jazyka VHLD a časovú simuláciu výstupného kódu z programového prostredia Xilinx Foundation Technology [7].

I.4 Návrh riešenia interaktívnej prezentácie

I.4.1 Návrh interaktívnej prezentácie

Prezentácia bude obsahovať tri vrstvy. Na najvyššej vrstve budú zobrazené vstupné dáta, šifrovací kľúč a výstupné dáta, všetko v *hexadecimálnom* tvare. Šifrovací kľúč má veľkosť 128 bitov. Vstupné alebo výstupné dáta majú veľkosť 64 bitov. Označením objektu znázorňujúceho šifrátor sa vyvolá zobrazenie nižšej úrovne.

Na zobrazení druhej vrstvy sa nachádzajú bloky zobrazujúce jednotlivé kolá šifrátora, spolu s vstupmi a výstupmi. Pre sprehľadnenie nebudú hodnoty vstupov a výstupov zobrazené okamžite, ale až po zaslaní požiadavky od používateľa. Po vybraní konkrétneho objektu bloku sa blok šifrátora zobrazí v zobrazení najnižšej úrovne.

Na najnižšej, poslednej úrovni, sú zobrazené všetky matematicko-logické funkcie a ich prepojenie. Opäť nie je vhodné zobrazovať všetky hodnoty na obrazovke implicitne, ale až po požiadavke od používateľa. Pri logických funkciách budú hodnoty zobrazované v binárnej číselnej sústave aby si užívateľ ľahšie mohol daný výsledok prepočítať sám.

I.4.2 Výber implementačného prostredia interaktívnej prezentácie

Interaktívna prezentácia šifry IDEA sa má nachádzať na tímovej web stránka. Z tohto pohľadu sa treba pozerať aj na problém výberu implementačného prostredia. Dajú sa rozdeliť na prostriedky, ktoré sa spúšťajú na web serveri a prostriedky spúšťané na klientskom počítači.

Na výber sme mali programovacie prostriedky: *PHP, Java aplet, Java*

script a Flash. Rozhodli sme sa pre prostriedok *Flash* od firmy *Macromedia* [8]. Pre tento prostriedok sme sa rozhodli preto, lebo sa najviac hodí na tvorbu interaktívnych prezentácií a je možné spúšťať prezentáciu aj z iného média ako je Internet alebo pracovať s prezentáciou bez pripojenia na Internet. Nevýhodou tohto riešenia je potreba pomocného modulu do prehliadača.

I.4.3 Návrh grafickej časti prezentácie šifry IDEA

Popri prezentácii samotného algoritmu práce šifry IDEA je dôležitý aj slovný opis činnosti a vysvetlenie prípadných odborných pojmov. Rovnako sa snažíme o vytvorenie prehľadného a intuitívneho prostredia. Za týmto účelom môžu byť jednotlivé úrovne odlíšené použitím rozdielnych farieb a hlavné okno prezentácie bude rozdelené na tri časti.

- V hornej časti sa pre jednoduchšiu orientáciu bude nachádzať okno informujúce o prezentačnej vrstve v ktorej sa práve nachádzame, môžu sa tu nachádzať aj odkazy na zvyšné vrstvy.
- V hlavnom okne bude graficky znázornená činnosť šifry na zvolenej úrovni. Všetky funkčné bloky diagramu musia byť pomenované rovnako ako vstupné a výstupné polia hodnôt.
- V pravej časti prezentácie bude okno v ktorom bude zobrazované doplňujúce informácie o obsahu hlavného okna a vysvetlenie použitých pojmov z oblasti šifrovania. Tieto údaje sa budú zobrazovať po *kliknutí* na aktívnu časť, alebo odkaz v hlavnom okne.

Pri prechode medzi jednotlivými úrovňami je potrebné zachovanie hodnôt vstupu, výstupu a prípadných hodnôt medzi jednotlivými operáciami, ktoré môžu byť zobrazené formou bublinkovej nápovedy *(text zobrazený v okienku tesne nad ukazovateľom myši)* pri pohybe myšou cez dátový prepoj, alebo implicitne v informačnom okne. Tu treba pri impementácii algoritmu šifry počítať s potrebou udržiavať všetky hodnoty dočasných výpočtov, ktoré chceme mať prístupné v prezentácii.

Navrhované časti prezentácie:

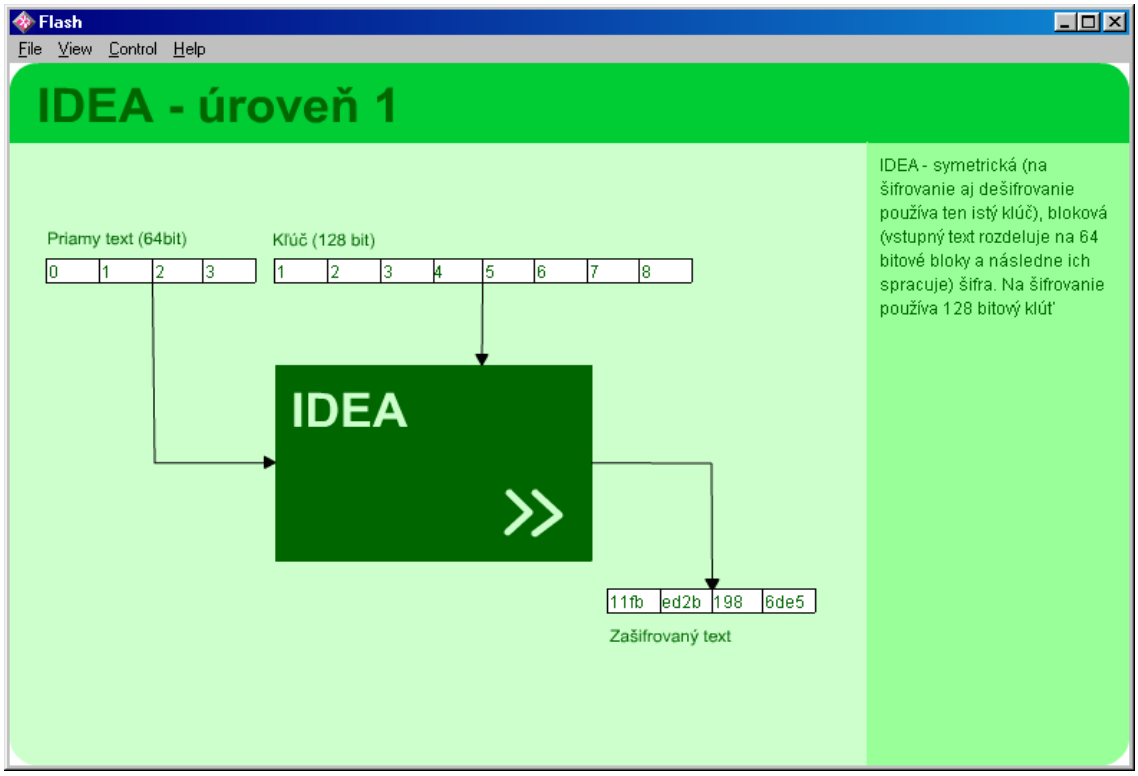
- Najvyššia úroveň zobrazuje šifru ako čiernu skrinku zo vstupnými údajmi priamy text šifrovací kľúč a výstupným zašifrovaným textom
- Úroveň jednotlivých kôl šifrovania so znázornenými dátovými tokmi medzi jednotlivými kolami. Zobrazených bude 8 kôl a blok spracovania šifrovacieho kľúča.
- Úroveň jedného kola so znázornenými bitovými operáciami. Toto zobrazenie bude spoločné pre prvých sedem kôl.
- Úroveň posledného kola na bitovej úrovne do ktorého vstupujú iba štyri vygenerované kľúče na rozdiel od predchádzajúcich kôl do ktorých vstupuje šesť kľúčov.
- Spracovanie šifrovacieho kľúča na bitovej úrovni.

I.5 Implementácia

I.5.1 Implementácia grafického rozhrania prezentácie

Na tvorbu prezentácie sme sa rozhodli využiť nástroj *Flash* 5 firmy Macromedia, ktorá umožňuje tvorbu interaktívnych webových stránok a jednoduchých animácií [5] [6] [8]. Na prezeranie výsledného produktu je potrebný prehliadač www stránok s inštalovaným modulom prehrávača *Flash* animácií, ktorý je volne stiahnuteľný na Internete.

V súčasnej dobe sú rozpracované prvé dve úrovne prezentácie šifry. Obrázok č.I.4 zobrazuje najvyššiu úroveň. Okno je rozdelené na tri časti ktoré zobrazujú aktuálnu úroveň, náhľad na šifru z najvyššej úrovne a časť zobrazujúcu spresnený popis zvolených častí v hlavnom okne.



Obr. č.I.4 Najvyššia úroveň prezentácie šifry IDEA.

I.5.2 Implementácia algoritmu prezentácie

Pri implementácii sme vychádzali z opisu šifrovacieho algoritmu v jazyku C, ktorý je možné nájsť v [3]. Celý algoritmus bol implementovaný v jazyku *„Action Script“* [5], ktorý je súčasťou zvoleného implementačného prostredia *Macromedia Flash*.

Pri vykonaní algoritmu sa budú uchovávať nasledovné údaje:

- Používateľom zadaný vstupný text a kľúč
- Všetky vygenerované šifrovacie aj dešifrovacie kľúče
- Vstupy a výstupy každého kola algoritmu
- V rámci každého kola algoritmu všetky vstupy a výstupy každého bloku (XOR, sčítanie násobenie)
- Všetky údaje budú v základnom tvare a pri vizualizácii sa podľa potreby budú transformovať do príslušnej sústavy (binárna, šestnástková, desiatková)

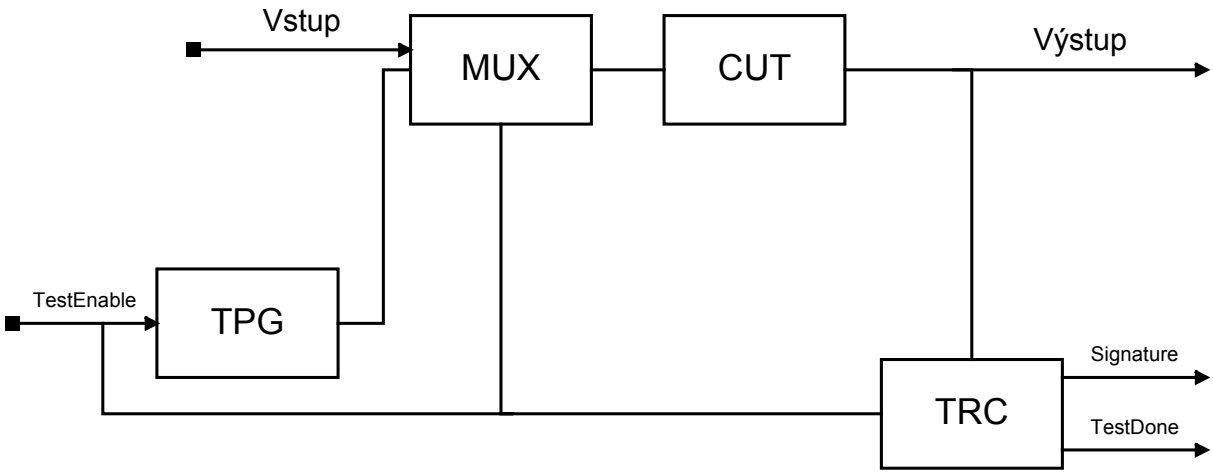
Vizualizačná časť bude zabezpečovať zobrazenie iba tých údajov, o ktoré bude mať používateľ záujem. Cieľom je nezahltiť používateľa nepotrebnými údajmi, ktoré by pôsobili neprehľadne.

I.6 Návrh a Implementácia konečného produktu

Zadanie projektu znie na optimalizáciu algoritmu na rýchlosť(priepustnosť). Tomu sme prispôsobili celý návrh implementácie šifry IDEA do programovateľného obvodu. Na rozdiel od procesorov je pri šifrovaní prerušenie zriedkavé a nastane iba pri zmene kľúča. Tým pádom sme mohli navrhnúť prúdový prostriedok a nemuseli sme sa moc zaujímať koľko bude mať prúdov, aj keď viac stupňov prúdu vkladá väčšie oneskorenie. Keď ale vďaka väčšiemu počtu prúdov dosiahneme vyššiu frekvenciu oneskorenie sa veľmi nezvýši.

I.6.1 BIST

Prípravu na testovanie výslednej hardvérovej implementácie sme zabezpečili pomocou vstavaného generátora testov. Generátor testov využíva na generovanie vzoriek LFSR 2. Ako generujúci polynóm som použil primitívny polynóm $x^{16}+x^3+x^2+x+1$. Na obrázku I.5 je generátor pod skratkou TPG. CUT "circuit under test" je samotný testovaný obvod, teda šifrátor IDEA. MUX prepína vstup do testovaného obvodu, buď na generátor testov(v režime testovania), alebo na vstupy od používateľa(v režime práce).

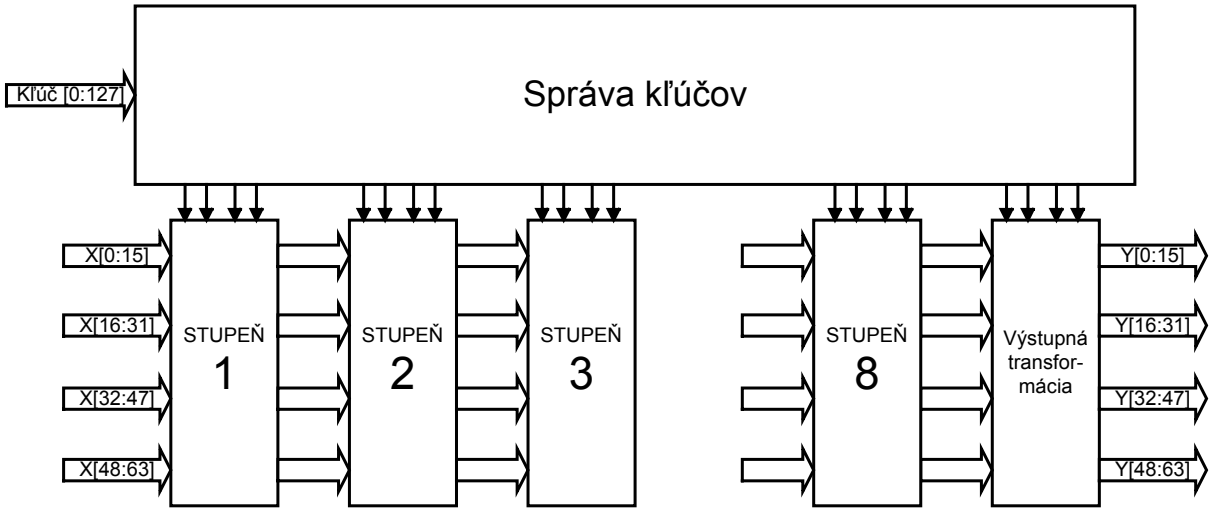


Obrázok č. I.5

Na testovanie bezporuchovosti výstupu šifrátora v režime testovania slúži obvod TRC, ktorý generuje kontrolnú sumu pomocou algoritmu LFSR 2 s primitívnym polynómom $x^{16}+x^3+x^2+x+1$. Na konci testovania, po 100 vzorkách porovná výslednú kontrolnú sumu s vopred zadanou správnou hodnotou.

I.6.2 Šifrátor IDEA

Celý šifrátor sme rozdelili na správu kľúčov a deväť stupňov samotného šifrátora. Do správy kľúčov vstupuje 128 bitové slovo a do stupňov šifrátora štyri 16 bitové slová. Do každého stupňa ešte vstupuje šesť podkľúčov zo správy kľúčov.



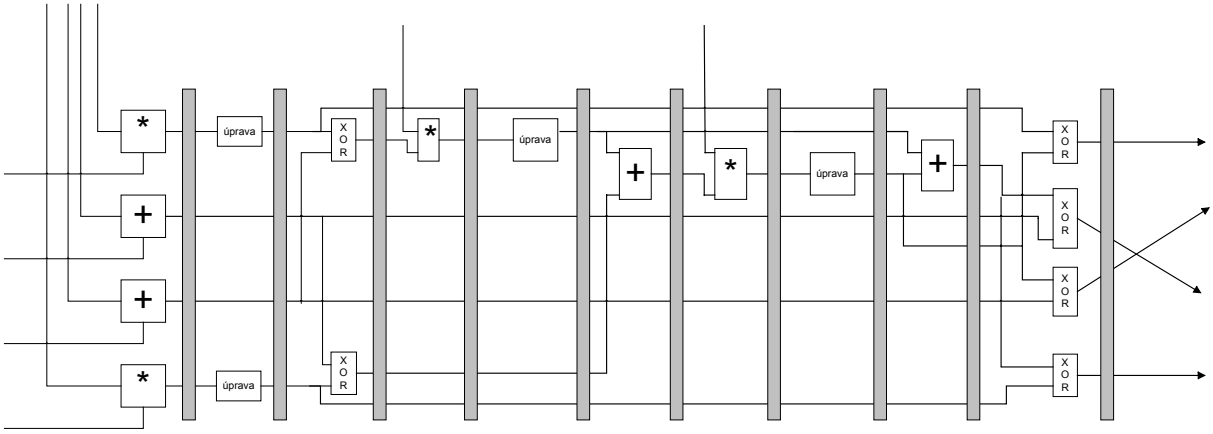
Obrázok č. I.6

Prvých osem stupňov je rovnakých, takže stačilo ich navrhnúť iba raz. Posledný stupeň sa volá výstupná transformácia.

I.6.3 Návrh jedného kola šifry

Jedno kolo je prúdová súčiastka na obrázku I.7. Stupeň sa skladá z 11

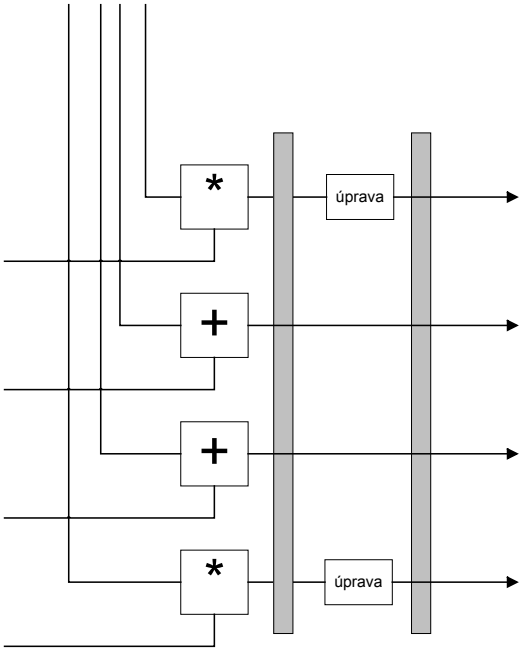
častí oddelených záchytnými registrami. Všetky časti stupňa, okrem úpravy násobenia sú priamo podporované zvoleným programovateľným obvodom a teda už sa nedajú rozdeliť.



Obrázok č. I.7

I.6.4 Návrh výstupnej transformácie

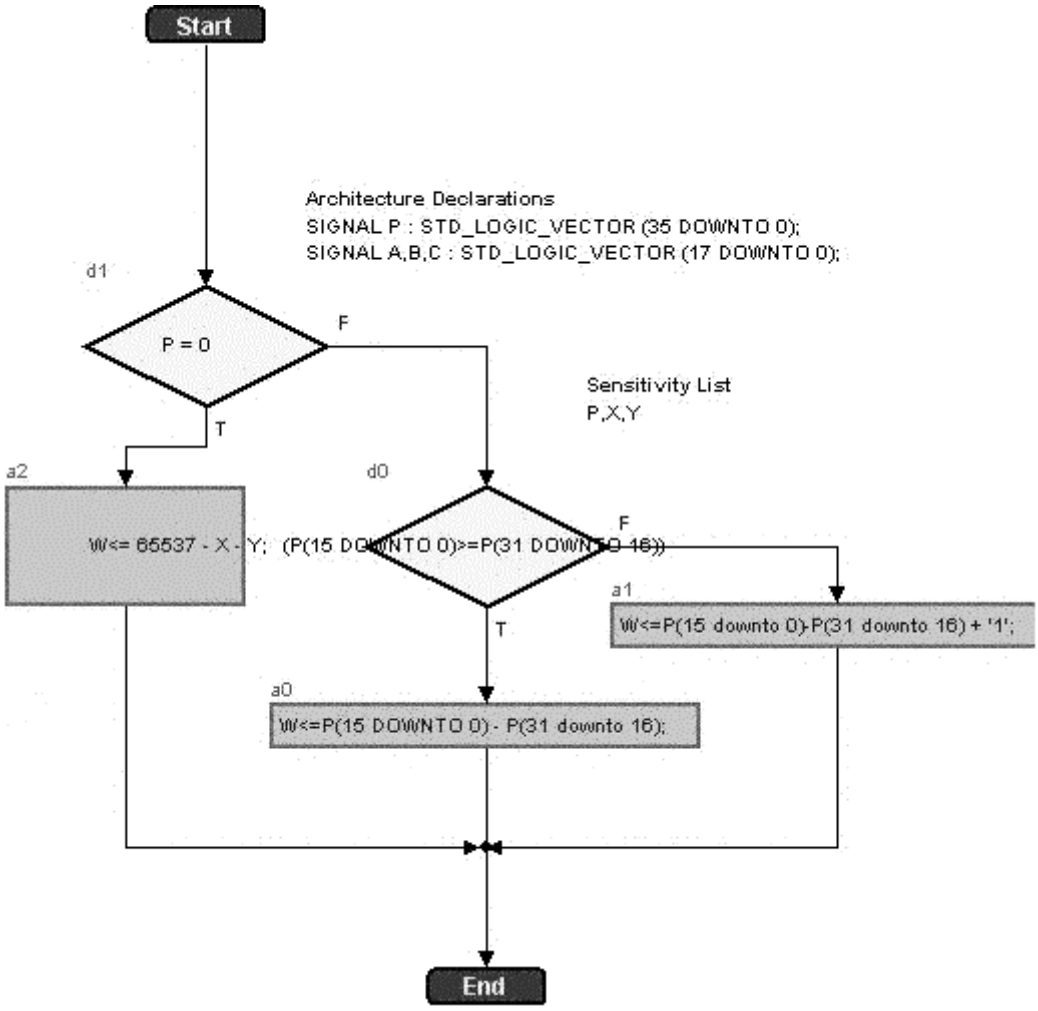
Posledný stupeň v šifre IDEA nieje plnohodnotný stupeň, ale iba prvá časť celé stupňa ako je vidieť na obrázku I.8.



Obrázok č. I.8

I.6.5 Návrh úpravy násobenia pre šifru IDEA

Pri šifre IDEA je potrebné používať trochu iné násobenie, ako je bežné. Rozdiel je pri násobení nulou. Násobenie nulou nedáva ako výsledok nulu. Na obrázku I.9 je vývojový diagram VHDL pre úpravu násobenie v šifre IDEA. Vstupom do tejto jednotky je 36 bitové slovo (P) a výstupom je 16 bitové slovo (W).



Obrázok č. I.9 Úprava násobenia v šifre IDEA

I.6.6 XOR

Z optimalizačných príčin sme súčiastku XOR zapísali pomocou (not(X) and Y) or (X and not(Y)). Tento spôsob vytvorenia bol rýchlejší do chvíle, kým sme neprerobili jedno kolo na viac prúdových stupňov, obrázok I.7, a tým pádom sa najpomalšou súčiastkou stalo násobenie.

I.6.7 Správa kľúčov

Správa kľúčov vykonáva funkciu distribúcie a výpočtu podkľúčov zo zadaného 128 bitového slova. Ďalej správa kľúčov vypočítava dešifrovacie kľúče pomocou algoritmu inverzného násobenia a inverzného delenia. Vstup SEL rozhoduje či sa budú distribuovať šifrovacie, alebo dešifrovacie kľúče. Dôležitým výstupom je KEYDONE, ktorý indikuje pripravenosť šifrátora na vkladanie dát. Po reset-e obvodu sa KEYDONE nastaví na nulu a pri pripravenosti na šifrovanie sa preklopí na jednotku. V prípade výpočtu šifrovacích kľúčov trvá tento výpočet iba tri takty, ale pri dešifrovacích kľúčoch je dĺžka trvania závislá na zadanom kľúči.

I.6.8 Návrh generovania šifrovacích kľúčov

Pri šifrovaní algoritmom IDEA vstupuje do každého kola sada šiestich kľúčov veľkosti 16 bitov. Pre osem kôl teda potrebujeme 48 kľúčov. Výstupná transformácia potrebuje ešte naviac štyri kľúče. Dohromady teda potrebujeme vygenerovať 52 šifrovacích kľúčov.

Ako už bolo spomínané, tieto šifrovacie kľúče sa generujú zo vstupného 128 bitového kľúča nasledovným spôsobom:

1. 128 bitový vstupný kľúč je rozdelený na osem 16 bitových kľúčov - prvých osem kľúčov

2. 128 bitový kľúč je cyklicky bitovo posunutý vľavo o 25 bitov a znovu rozdelený na osem ďalších 16 bitových kľúčov

3. druhý krok sa opakuje dovtedy, pokiaľ nie je vygenerovaných všetkých 52 kľúčov

Tento algoritmus predpokladá cyklický bitový posun vľavo pokiaľ nie sú všetky kľúče vygenerované. Po bližšom skúmaní tohoto postupu dôjdeme k záveru, že celý postup sa dá optimalizovať takým spôsobom, že každý z 52

šifrovacích kľúčov sa dá priamo odvodiť zo vstupného 128 bitového kľúča výberom určitej postupnosti bitov. Takto sa docieli veľmi rýchle vygenerovanie šifrovacích kľúčov, kde oneskorenie spôsobuje len oneskorenie na vodičoch.

Generovanie 52 šifrovacích kľúčov zo vstupného 128 bitového kľúča priamo určuje nasledovná tabuľka. Táto tabuľka v podstate zobrazuje závislosť bitov kľúčov od vstupných 128 bitov.

kolo	sk1	sk2	sk3	sk4	sk5	sk6
1	0–15	16–31	32–47	48–63	64–79	80–95
2	96–111	112–127	25–40	41–56	57–72	73–88
3	89–104	105–120	121–8	9–24	50–65	66–81
4	82–97	98–113	114–1	2–17	18–33	34–49
5	75–90	91–106	107–122	123–10	11–26	27–42
6	43–58	59–74	100–115	116–3	4–19	20–35
7	36–51	52–67	68–83	84–99	125–12	13–28
8	29–44	45–60	61–76	77–92	93–108	109–124
OT	22–37	38–53	54–69	70–85	—	—

Obr. č. I.10 : Tabuľka generovania šifrovacích kľúčov

- sk1-6 : daná šestica šifrovacích kľúčov pre dané kolo algoritmu

Všetky dešifrovacie kľúče v našej implementácii sú generované týmto spôsobom.

I.6.9 Návrh generovania dešifrovacích kľúčov

Pre pripomenutie uvádzame, že dešifrovacie kľúče sa generujú zo šifrovacích kľúčov na základe nasledovnej tabuľky:

Dešifrovacie kľúče						
kolo r	kľúč1	kľúč2	kľúč3	kľúč4	kľúč5	kľúč6
1	K49*	K50#	K51#	K52*	K47	K48
2	K43*	K45#	K44#	K46*	K41	K42
3	K37*	K39#	K38#	K40*	K35	K36
4	K31*	K33#	K32#	K34*	K29	K30
5	K25*	K27#	K26#	K28*	K23	K24
6	K19*	K21#	K20#	K22*	K17	K18
7	K13*	K15#	K14#	K16*	K11	K12
8	K7*	K9#	K8#	K10*	K5	K6
9 = VT	K1*	K2#	K3#	K4*	-	-

Obr. č. I.11 : Tabuľka generovania dešifrovacích kľúčov

Kxx* = MULINV - inverzné násobenie kľúča Kxx modulo $(2^{16} + 1)$

Kxx# = ADDINV - inverzné sčítanie kľúča Kxx modulo (2^{16})

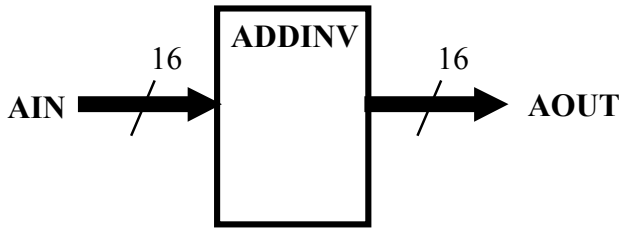
VT – výstupná transformácia

Z tabuľky vyplýva, že treba navrhnúť dva základné komponenty:

- inverzné sčítanie
- inverzné násobenie

I.6.9.1 Inverzné sčítanie

Inverzné sčítanie predstavuje jednoduchý komponent s jedným 16 bitovým vstupom a jedným 16 bitovým výstupom.



Obr. č. I.12: Komponent inverzného sčítania

Tento komponent vykonáva jednoduchú funkciu:

```
AOUT <= 65536 - AIN
```

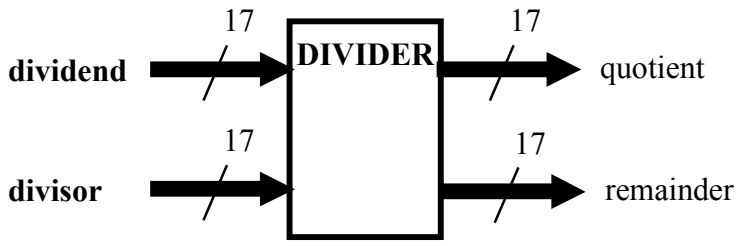
I.6.9.2 Inverzné násobenie

Funkcia inverzného násobenia vráti takú hodnotu, že platí:

(X * MULINV(X)) modulo 65537 = 1

Pri implementácii treba najprv navrhnúť deličku, nakoľko delenie je pre daný algoritmus nevyhnutné. My sme sa po bližšom skúmaní možných algoritmov delenia rozhodli použiť už implementovanú deličku od Rolanda Höllera, ktorá bola implementovaná v rámci projektu volného IP jadra procesora 8051.

Delička je implementovaná ako kombinačný logický obvod. Je plne syntetizovateľná a parametrizovateľná. Pre naše potreby sme definovali 17 bitové vstupy a výstupy.



Obr.č. I.13: Delička

Funkcia inverzného násobenia sa dá odvodiť z rozšíreného Euklidovho algoritmu. Celý hardvérový návrh vychádza z optimalizovaného algoritmu implementovaného vo vyššom programovacom jazyku (ActionScript).

V tomto algoritme sme identifikovali základné štyri fázy:

Algoritmus vo vyššom programovacom jazyku FÁZY

```
function ideamulinv(k)
 {
 var a,b,y,y1,y2,q,r;

 if(b==0)
  { return 0; }

 y2=0; y1=1;
 b=k;
 a=65537;

 while (b>0)
  {
        q =  a/b ;//celociselne delenie
        r =a%b;   //zvysok po deleni

        y=y2-q*y1;

        a=b; b=r; y2=y1; y1=y;
  }

 if (y2>=0) return y2&0x0000ffff;
 else return (65537+y2)&0x0000ffff;
 }
```

inicializácia

delenie

násobenie a zápis medzi výsledkov

výstupná úprava

Na základe týchto štyroch fáz sme navrhli konečný stavový automat so

štyrmi stavmi zodpovedajúcimi daným fázam.

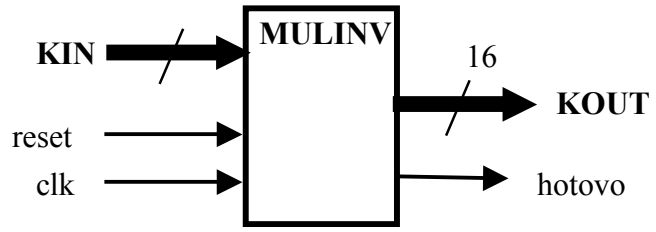
Fázy	**Stavy**
1. inicializácia	init
2. delenie	delenie
3. násobenie a zápis medzi výsledkov	násobenie
4. výstupná úprava	hotovostav

Vykonávanie algoritmu je dané na základe prechodu medzi jednotlivými stavmi konečného automatu:

STAV	Podmienka	Nasledujúci STAV
INIT	Ak k=0 inak	HOTOVOSTAV DELENIE
DELENIE		NASOBENIE
NASOBENIE	pokiaľ b>0 inak	DELENIE HOTOVOSTAV
HOTOVOSTAV		HOTOVOSTAV

Celý obvod je synchronizovaný na nábežnú hranu synchronizačného vstupu CLK. Naviac je definovaný ešte riadiaci vstup RESET, ktorý pri aktívnej hodnote nastaví obvod do stavu INIT. Pretože daný algoritmus trvá pre rôzne vstupy rôzny počet cyklov, treba definovať riadiaci výstup HOTOVO, ktorý má aktívnu hodnotu ak sú už platné výsledky na výstupe.

Celý komponent inverzného násobenia potom vyzerá nasledovne:



Obr.č. I.14: Komponent inverzného násobenia

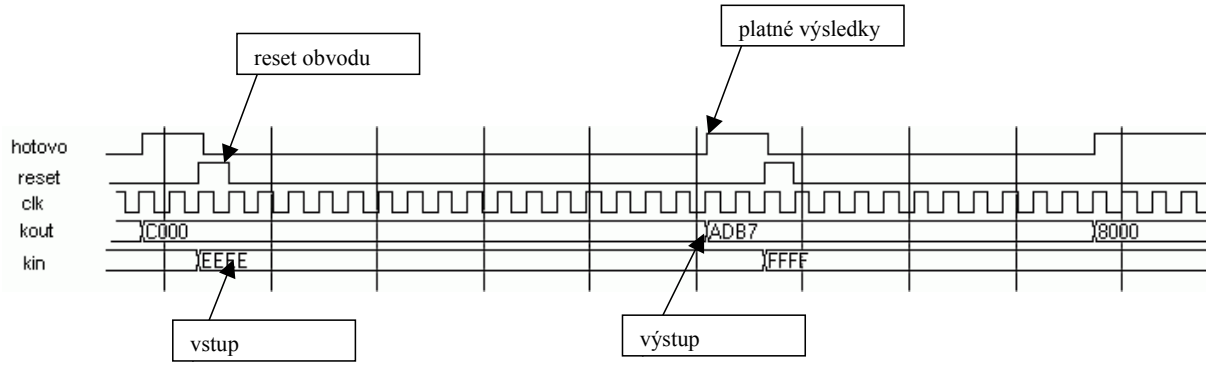
KIN – 16 bitový vstup

KOUT – 16 bitový výstup

reset – riadiaci vstup, spôsobí reset obvodu

clk – synchronizačný vstup (hodiny)

hotovo – riadiaci výstup indikujúci platnosť výstupu KOUT



Obr. č. I.15: Časovanie komponentu inverzného násobenia

I.7 Testovanie

Testovanie obvodu je realizované dvomi spôsobmi.

- Softvérová metóda – vytvorením testovacieho VHDL kódu
- Hardvérová metóda – implementácia samotestovateľného modulu vo vnútri čipu.

I.7.1 Softvérová metóda

Jednou z možností ako otestovať funkčnosť obvodu je vytvorenie testovacieho programu ktorý simuluje zadávanie dát na vstupy obvodu a sleduje akým spôsobom sa menia výstupné signály. Testovanie spočíva v tom, že sa postupne zadávajú vstupné dáta určené na šifrovanie a aj dešifrovanie, pričom dáta by mali tvoriť reprezentačnú vzorku. Vzorka by mala pokrývať všetky poruchy aby sa dosiahlo úplné otestovanie obvodu. V našom prípade, kedy zložitosť obvodu je neúmerne vysoká a ani nepoznáme detailne jeho štruktúru sme ako testovaciu vzorku zvolili náhodne vygenerovanú postupnosť 100 čísiel určených na šifrovanie a následne na dešifrovanie. Pri každej sade vzoriek sme použili 2 náhodné kľúče. Na generovanie testovacieho kódu resp. funkcie sme vytvorili program zapísaný v jazyku C++ ktorý na základe akejsi šablóny vytvorí požadovaný zdrojový kód. Tento kód sa spojí s kódom šifrátora a následne sa spustí simulácia v Xilinx. Simulátor vygeneruje časový diagram na ktorom sú znázornené všetky potrebné priebehy signálov. Na základe týchto údajov sme vizuálne skontrolovali správnu činnosť obvodu. Náhodne sme si vybrali 25 vzoriek pri ktorých sme si pomocou WEB prezentácie vypočítali správne výsledky ktoré sme potom overili aj softvérovou simuláciou. Testovanie dopadlo úspešne.

I.7.2 Hardvérové testovanie

Nedá sa v pravom slova zmysle hovoriť o hardvérovom testovaní, pretože prakticky celé testovanie sa simuluje v prostredí Xilinx. V čipe je integrovaný samotestovací obvod BIST (Build-In Self Test) ktorý jednoduchým a rýchlim spôsobom otestuje funkčnosť obvodu. Nastavením riadiacich signálov na vstupe obvodu sa obvod prepne do samotestovacieho režimu, pričom úspešnosť resp. neúspešnosť testu indikuje nastavením výstupných riadiacich signálov, konkrétne TEST_DONE a EQUAL. Naše testovanie spočívalo v nastavení príslušných signálov potrebných na prepnutie obvodu do testovacieho režimu a následné odčítanie signálu EQUAL z časového diagramu. Test bol úspešný.

I.8 Používateľská príručka

I.8.1 Rozhranie šifrátora

Na obr. I.16 sú zobrazené všetky vstupno-výstupné zbernice a riadiace signály ktoré sú použité v implementácii. Popis signálov:

- X :
  - Šírka: 64 bitov
  - Typ: vstup

  Vstup dát ktoré sa budú šifrovať alebo dešifrovať.
- Y :
  - Šírka: 64 bitov
  - Typ: výstup

  Výstup šifrovaných prípadne dešifrovaných dát.
- INKEY :
  - Šírka: 128 bitov
  - Typ: vstup

  Vstup inkey očakáva vloženie šifrovacieho prípadne nešifrovacieho kľúča.
- CLK :
  - Typ: vstup

  Časovanie obvodu.
- RST :
  - Typ: vstup

  Inicializácia obvodu.
- SEL :
  - Typ: vstup

  Určenie režimu činnosti – šifrovanie alebo dešifrovanie.
- TEST_ENABLE :
  - Typ: vstup

  Prepnutie obvodu do samotestovateľného režimu.

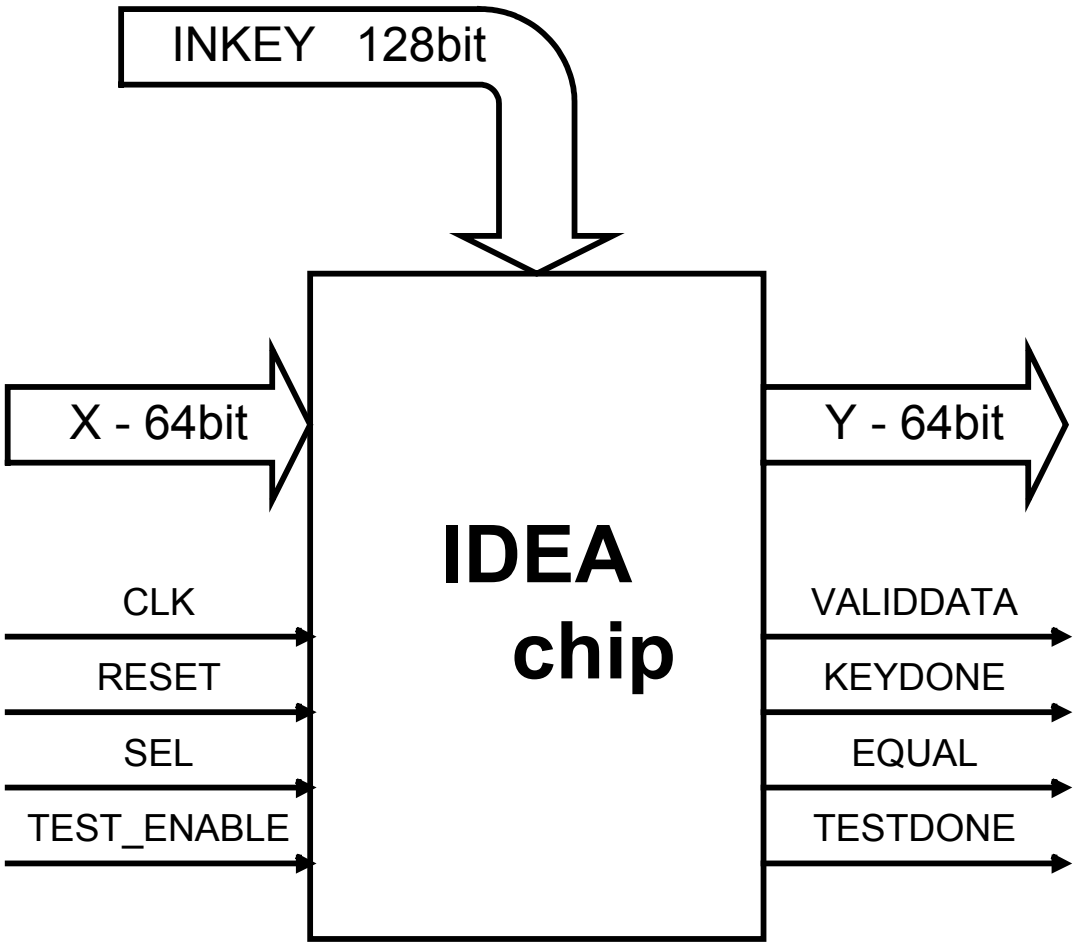
- VALIDDATA :
  - Typ: výstup

    Indikácia platnosti výstup.
- KEYDONE :
  - Typ: výstup

    Indikácia platnosti zadaného kľúča.
- EQUAL
  - Typ: výstup

    Indikácia úspešného testu.
- TESTDONE
  - Typ: výstup

    Indikácia ukončenia testu. Signál nenesie informáciu o úspešnom alebo neúspešnom teste ale iba o ukončení testu.



Obr. I.16: Rozhranie šifrátora

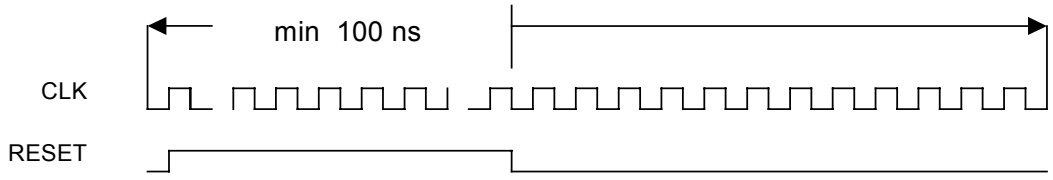
I.8.2 Režimy činnosti

Čip pracuje v 4 režimoch:

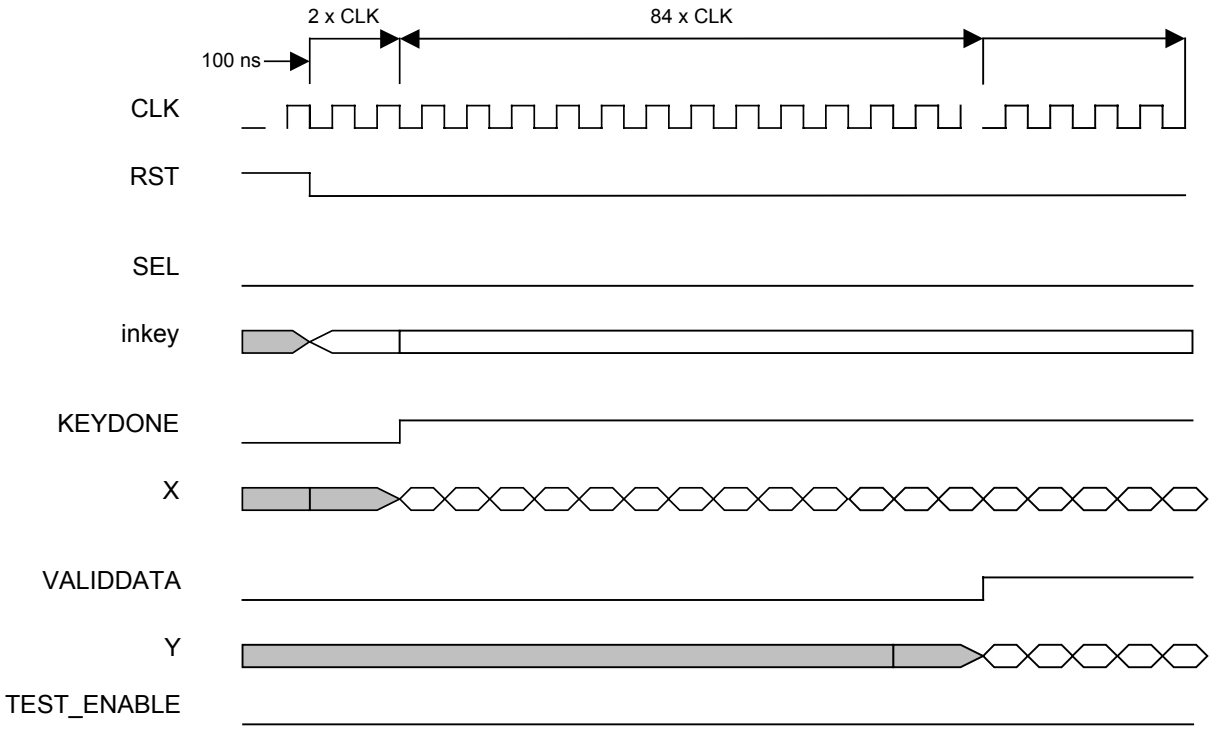
1. **inicializácia** – Režim sa aktivuje pomocou signálu RST ktorý je aktívny vo vysokej úrovni. Časový priebeh zobrazujúci inicializáciu obvodu je na obr. I.17. Obvod je nutné inicializovať vždy po pripojení napätia, po zmene kľúča alebo po zmene módu činnosti (výber šifrovania alebo dešifrovania). Počas inicializácie sú ignorované akékoľvek ostatné signály. Obvod môže generovať na výstupe Y náhodné hodnoty ktoré však nie sú platné. Signál RST musí byť aktívny po dobu minimálne 100 ns.
2. **Výpočet kľúča** – Po ukončení inicializácie automaticky obvod prechádza

do režimu výpočtu kľúča (obr. I.17 a I.18). Na základe signálu SEL, ktorým sa určuje či sa jedná o šifrovanie alebo dešifrovanie, je aj stanovená doba ktorú treba na výpočet kľúča. Na výpočet sady šifrovacích kľúčov treba vždy 2 takty. NA výpočet dešifrovacích kľúčov nie je presne stanovená doba kedy sú kľúče k dispozícii. V tomto prípade dokončenie výpočtu indikuje signál KEYDONE.

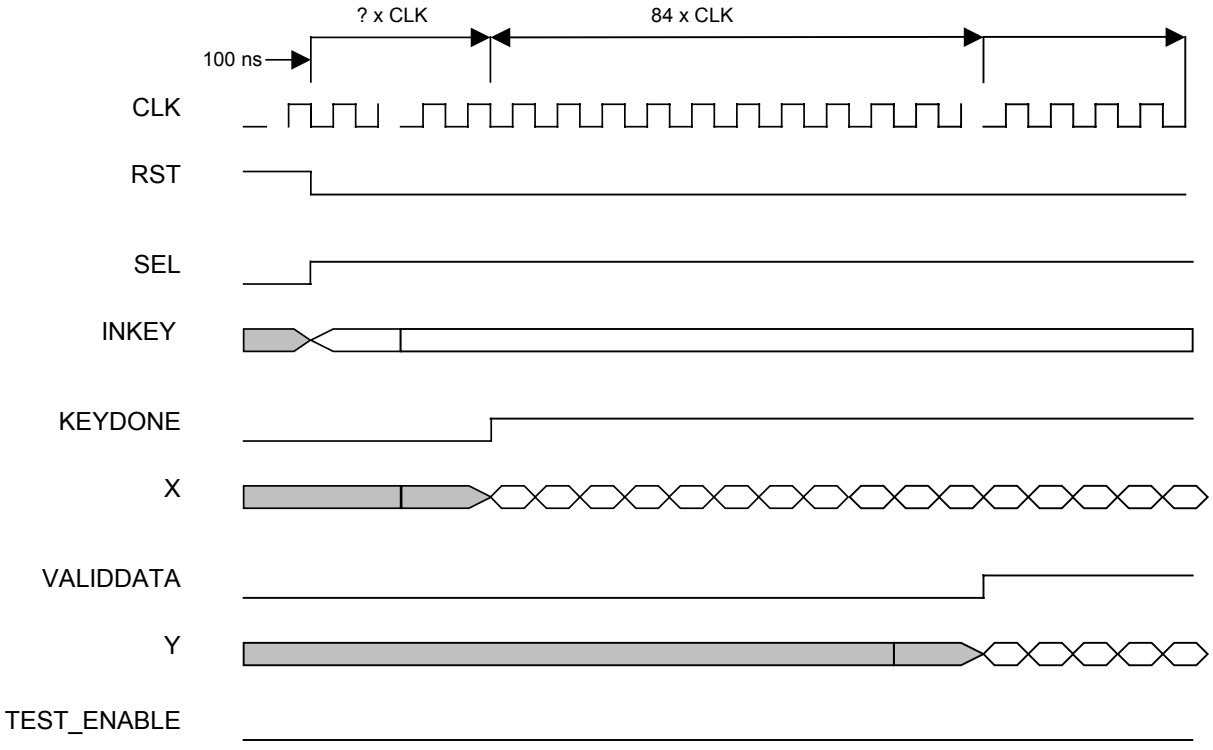
3. **Šifrovanie resp. dešifrovanie** – Po aktivácii signálu KEYDONE ktorý je generovaný po úspešnom spracovaní zadaného kľúča a ktorý je aktívny vo vysokej úrovni sa môžu začať vkladať dáta na šifrovanie prípadne dešifrovanie. Na obr. I.17 je znázornené šifrovanie a na obr. I.18 dešifrovanie. V každom hodinovom cykle je spracovaný blok dát o veľkosti 64 bitov. Očakávané dáta sú k dispozícii vždy po 84 taktoch od aktivácii signálu KEYDONE. Korektné údaje sú tiež signalizované signálom VALIDDATA ktorý je aktívny vo vysokej úrovni. Počas týchto 84 taktoch sú na výstupe Y generované dáta ktoré sú spôsobené vnútorným zapojením obvodu a nie sú platným výsledkom. Používateľ by ich mal ignorovať.
4. **Testovací režim** – Režim je aktivovaný signálom TEST_ENABLE ktorý je aktívny vo vysokej úrovni. V tomto režime obvod otestuje svoju vnútornú štruktúru ako aj výpočet kľúčov. Obr. I.19 znázorňuje časový priebeh režimu testovania.



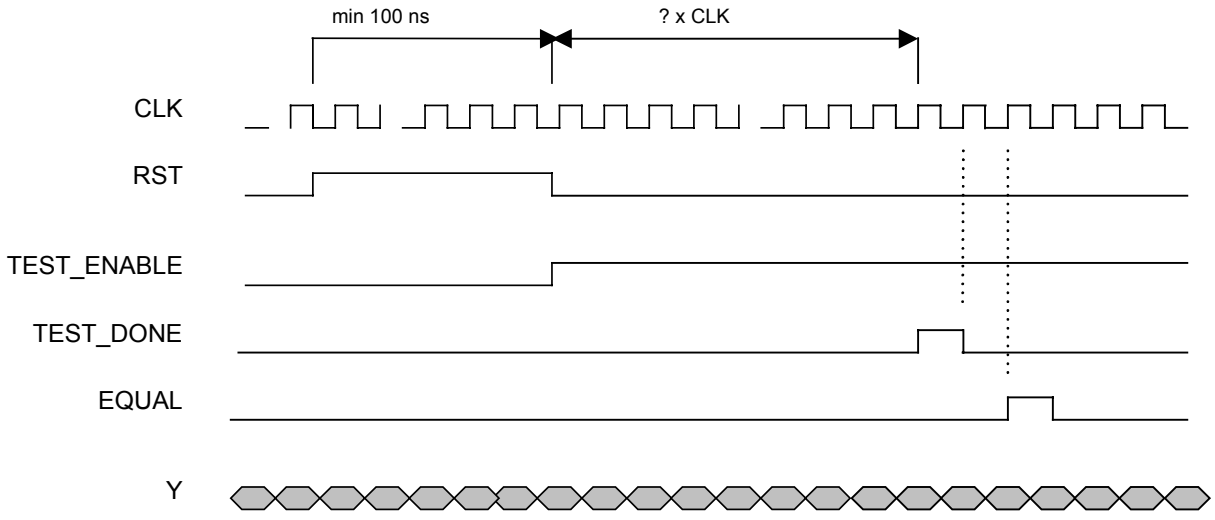
Obr. I.16 Inicializácia



Obr. I.17 Šifrovanie



Obr. I.18 Dešifrovanie



Obr. I.19 Testovací režim

I.8.3 Porovnanie priepustnosti

Celý čip bol optimalizovaný na priepustnosť ktorá sa udáva v bitoch za sekundu. Podarilo sa nám dosiahnuť rýchlosť približne 83 MHz. Pri spracovaní 64 bitov počas jedného taktu je priepustnosť obvodu:

P = 83MHz * 64 bitov = 5312 Mbps (Mega bit per second – Mega bitov za sekundu)

Zanedbávame inicializáciu a počiatočnú latenciu obvodu, keďže tieto fázy sa vyskytujú iba na začiatku práce obvodu a z pohľadu dlhodobejšej práce sú zanedbateľné.

V obr. I.20 sú uvedené rýchlosti zberníc používaných v bežných ako aj serverových počítačoch. Z tabuľky vyplýva, že rýchlosť šifrovania aj dešifrovania je postačujúca pre všetky bežne používané zbernice. Iba v jednom prípade je rýchlosť čipu nepostačujúca. Jedná sa o zbernicu PCI-X ktorá nie je zatiaľ komerčne nasadená a je iba vo vývoji.

Problém môže nastať iba v prípade, že by používateľ veľmi často menil kľúče, pretože po zmene kľúča je nutná inicializácia obvodu, následne výpočet nových kľúčov a 84 taktov na získanie prvých správnych výsledkov.

	šírka zbernice	**frekvencia [MHz]**	**priepustnosť [Mbps]**	
ISA	16 bit	8	128	√
EISA	32 bit	8	256	√
VESA	32 bit	33	1056	√
MCA	16 - 32 bit	10 - 16	160 - 512	√
PCI	32 bit	33	1056	√
PCI64	64 bit	33	2112	√
PCI-X	64 bit	133	8512	x
USB 1.1	serial	x	12	√
USB 2.0	serial	x	480	√
FireWire 400	serial	x	400	√
FireWire 800	serial	x	800	√
Ethernet 10Mbit	serial	x	10	√
Ethernet 100Mbit	serial	x	100	√
Ethernet 1Gbit	serial	x	1000	√
Token Ring	serial	x	16	√

Obr. I.20: Rýchlosti zberníc

I.8.4 Používateľská príručka programu IdeaTest

I.8.4.1 Účel programu

Pri návrhu a vytváraní opisu šifrátora Idea, vznikla potreba priebežne testovať jednotlivé verzie implementácie. Na otestovanie činnosti šifrátora, ktorého správanie je opísané pomocou jazyka VHDL, sa používa tzv. „testbench“. Tento testbench je rovnako ako celý opis správania šifrátora, zapísaný pomocou jazyka VHDL a jeho účelom je na zadané vstupy šifrátora posielať testovacie dátové vzorky pri zvolenom časovaní a prípadné porovnávanie výsledkov spracovania.

S postupom vývoja a zmien vytváraných v šifrátore bolo nutné rovnako meniť aj hodnoty a štruktúru testu ako aj v niektorých prípadoch zväčšiť počet vzoriek v teste aby bolo možné odskúšať šifrátor počas dlhšej periódy. V týchto podmienkach by bolo časovo náročné ručne prepisovať testovací súbor najmä v prípade potreby väčšieho množstva vzoriek. Z tohto dôvodu vznikla potreba automatizovať proces generovania testovacieho súboru s kódom VHDL podľa meniacich sa požiadaviek na časovanie, počet vzoriek a meniacu sa štruktúru testu. Výsledkom snahy o zjednodušenie tohto procesu je program IdeaTest, ktorý zjednodušuje generovanie testov pri uvedených podmienkach.

I.8.4.2 Činnosť programu

Ako sme uviedli vyššie cieľom programu bolo cieľom nielen zmenu hodnôt a počtu vzoriek ale aj umožniť vykonať zmeny v štruktúre testovacieho súboru bez nutnosti upraviť program na generovanie testov. Za týmto účelom program na svoju činnosť využíva vzorový súbor s testom. Tento súbor má pevne určené meno *ideatb_vzor* a musí byť pre správnu činnosť programu umiestnený v rovnakom adresári ak samotný program.

Súbor ideatb_vzor je, upravený súbor s testom pre šifrátor. Úprava súboru spočíva v pridaní špeciálnych sekvencií znakov v miestach na ktoré budú zapísané zadané hodnoty programu IdeaTest. Program rozoznáva nasledujúce špeciálne sekvencie znakov:

- ##clk1## - používaná ako čas nábežnej hrany
- ##clk0## - používaná ako čas dobežnej hrany
- ##zaccas## - používaná ako čas zadania prvej vzorky
- ##rep## - symbol opakovania od miesta značky po koniec riadku, pričom program ukončuje všetky riadky čiarkou a poslednú bodkočiarkou
- ##x1## až ##x4## - nahrádza 64 bitovou hodnotou vygenerovanej vsupnej

vzorky, rozdelenej na štyri 16 bitové bloky, zapísanej v binárnej sústave

- ##y1## až ##y4## - nahrádza 64 bitovou hodnotou vypočítanej zašifrovanej vzorky, rozdelenej na štyri 16 bitové bloky, zapísanej v binárnej sústave
- ##index## - používa sa v spojení so značkou opakovania a je nahrádzaný číslom iterácie opakovania
- ##inkey## - je nahradený 128 bitovým šifrovacím kľúčom zapísaným v binárnom tvare

Pri potrebe zmeny štruktúry testovacieho súboru je teda možné zmeniť vzorový súbor programu a tým dosiahnuť generovanie nových testovacích súborov bez nutnosti zmeny programu IdeaTest.

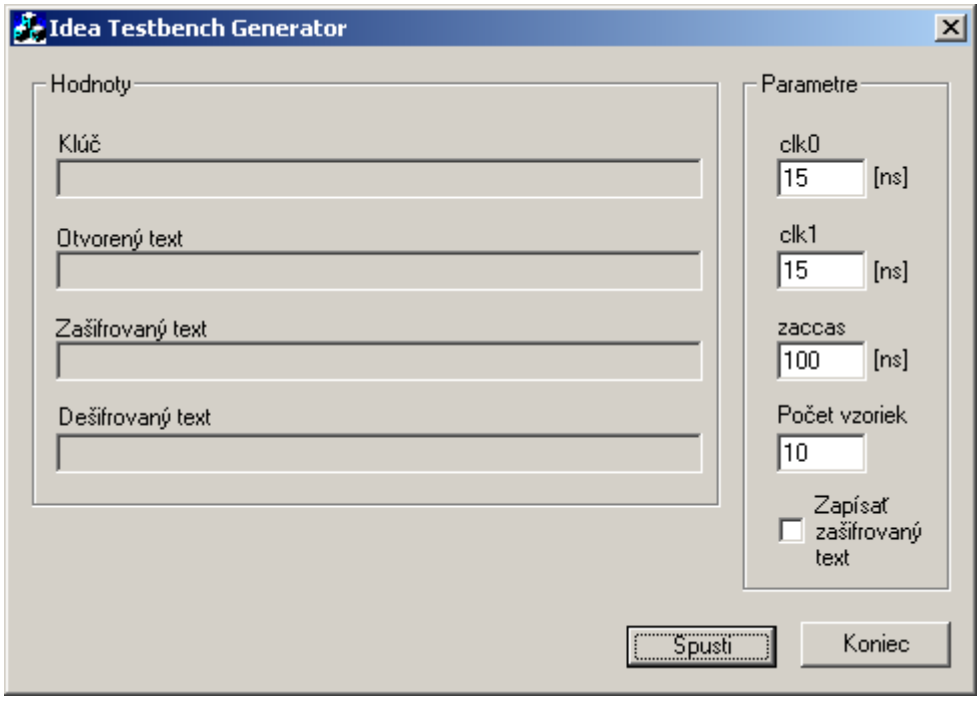
Pri výpočte zašifrovaných hodnôt je z dôvodu overenia správnosti šifrovania vzorky najskôr vygenerovaná hodnota testovacej vzorky vstupného testu zašifrovaná a následne dešifrovaná táto hodnota je porovnaná s pôvodnou vygenerovanou hodnotou a program pokračuje iba v prípade, že sú tieto hodnoty rovnaké.

I.8.4.3 Prostredie programu

Program je dialógová aplikácia ktorá je rozdelená na dve časti. V pravej časti – parametre program umožňuje zadávať vstupné hodnoty ktoré budú vkladané do testovacieho súboru. Konkrétne sú to hodnoty:

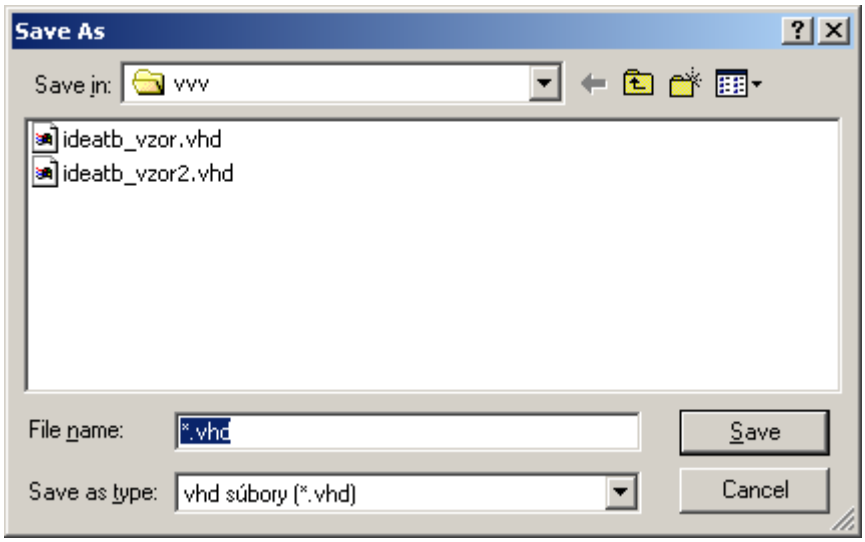
- clk0 – nahrádza hodnotou značku ##clk0##
- clk1 – nahrádza hodnotou značku ##clk1##
- zaccas – nahrádza hodnotou značku ##zaccas##
- pocet vzoriek – určuje počet vygenerovaných testovacích vzoriek
- zapísať zašifrovaný text – po povolení tejto voľby sú do súboru zapísané aj zašifrované vzorky pre jednoduché porovnávanie výsledkov zo šifrátora.

V ľavej časti – hodnoty, sa zobrazujú generované vzorky spolu so zodpovedajúcimi zašifrovanými a opätovne dešifrovanými hodnotami a hodnotou kľúča. (Obr. I.21)



Obr. I.21. Okno programu IdeaTest.

V dolnej časti okna sa nachádzajú tlačítka pre ukončenie programu a spustenie generovania testovacieho súboru. Po spustení generovania sa otvorí dialógové okno ktoré umožňuje zadať miesto a názov nového vygenerovaného testovacieho súboru, (obr. I.22). Po potvrdení výberu je súbor vygenerovaný a program zobrazí hlásenie o úspešnom vytvorení testovacieho súboru.



Obr. I.22. určenie miesta a názvu nového súboru.

I.9 Literatúra

[1] A. Menezes, P. van Oorschot, and S. Vanstone: Handbook of Applied Cryptography, CRC Press, 1996.

[2] A. Buldas, J. Poldre: A VLSI Implementation of RSA and IDEA encryption engine

[3] Fauzan Mirza: International Data Encryption Algorithm – Implementation summary

[4] Encryption solutions for secure communication, http://www.mediacrypt.com, (október 2002)

[5] Macromedia Inc. 2000. Macromedia Flash 5 Action Script.

[6] Macromedia Inc. 2000. Macromedia Flash 5 Užívateľská príručka.

[7] XILINX homepage, http://www.xilinx.com.

[8] Macromedia homepage, http://www.macromedia.com