Architektury akceleratorów kryptograficznych opartych na układach logicznych FPGA.

Marcin Rogawski
rogawskim@prokom.pl

PROKOM Software S.A.
Ul. Grójecka 127
Warszawa Polska

Wprowadzenie

Rozwój telekomunikacji i cyfrowego przetwarzania danych, rosnącą rola informacji i oszczędności czasu, we współczesnym świecie, spowodowała, że konieczność ochrony przechowywanych oraz przekazywanych ogólnodostępnymi kanałami wiadomości, stała się indywidualna i naturalna potrzeba każdego człowieka. Jednocześnie szybki rozwój zaawansowanych technik kryptoanalitycznych sprawia, że przed projektami kolejnych algorytmów szyfrowania pojawiają się coraz większe wymagania. Dzisiaj nie wystarcza już tylko udowadniane bezpieczeństwo szyfru. Musi on dodatkowo spełniać warunki dotyczące takich zagadnień jak efektywność jego implementacji programowej i sprzętowej, szybkość realizowanej operacji szyfrowania i deszyfrowania, a także adaptacyjność w nietypowych środowiskach. Celem niniejszej publikacji jest zwrócenie uwagi czytelnika na korelacje faktów dotyczących bezpieczeństwa algorytmu, sposobu implementacji i szybkości działania jego realizacji w akceleratorach kryptograficznych opartych na układach programowalnych FPGA.

Spis treści:

Wprowadzenie do akceleratorów kryptograficznych

W systemach cyfrowych bardzo skomplikowane operacje, dotyczące takich zagadnień jak obliczenia na liczbach zmienno-przecinkowych, realizacja zaawansowanych systemów multimedialnych i wiele innych operacji wymagają dużej liczby zasobów komputera (czasu procesora, pamięci RAM).
Zadania tego typu traktowane są wiec w szczególny sposób i realizowane są poprzez dedykowane układy scalone nazywane **koprocesorami** albo **akceleratorami sprzętowymi** danych operacji.

Szyfrowanie to szczególny rodzaj działania matematycznego, które potrzebuje do poprawnej realizacji nie tylko dużej ilości zasobów komputerowych, ale także wymaga operacji na coraz to większych liczbach.

Wzrastające wymagania na bezpieczeństwo transmisji oraz przechowywania danych powodują, że powstają coraz bardziej zaawansowane algorytmy szyfrowania. Realizacje programowe są różnej klasy – nigdy jednak najlepsze z nich nie są w stanie równać się ze specjalizowanymi układami scalonymi.

Układy scalone, które maja możliwość wielokrotnego re-programowania, o których będzie mowa w dalszej części artykułu noszą nazwę FPGA (ang. *field programable gate array*).

Większość akceleratorów kryptograficznych jest zorganizowana w taki sposób, aby wszystkie operacje, szyfrowania i deszyfrowania oraz operacje dostępu do pamięci głównej bądź dyskowej były realizowane przez moduły sprzętowe. W związku z tym faktem możemy wyróżnić w nich podstawowe elementy:

- bloki wspierające operacje I/O,
- bloki pamięci wewnętrznej RAM,
- krypto kanały, konfigurujące akcelerator,
- kontrolery, realizujące funkcje procesora,
- jednostki wykonujące operacje kryptograficzna.

W artykule będziemy mówić o architekturach jednostek wykonujących szyfrowania/deszyfrowania ale także obliczania wartości funkcji skrótu.

1. Analiza funkcji rundy algorytmu blokowego pod katem implementacji sprzętowej.

Teoria Shanonna dała podstawy pod budowę współczesnych szyfrów blokowych. Postulowała ona kilkukrotne naprzemienne stosowanie warstwy liniowej (dyfuzji) i nieliniowej (konfuzji). Pojedyncze zestawienie warstwy podstawiającej i rozpraszającej wraz z operacja dodania podklucza stanowi rundę szyfru. Kilkukrotnie powtórzona wraz z oddzielnym algorytmem generacji podkluczy stanowi cały szyfr. Analiza funkcji rundy algorytmu blokowego pod katem implementacji sprzętowej sprowadza się do przeprowadzenia badania ile potrzeba niezbędnych zasobów do realizacji rundy szyfrowania bądź deszyfrowania. Następnie w zależności od tego jak w stosunku do ilości dostępnych zasobów wygląda zestawienie efektywności poszczególnych elementów następuje wybór strategii dalszego projektowania.

Projektowanie akceleratora kryptograficznego w układach programowalnych realizującego określony algorytm blokowy polega wiec na analizie:

- zasobów przeznaczonych na realizacje projektu:
  - wybór układu programowalnego (charakterystyka podstawowych parametrów – ilość wejść i wyjść oraz ich rodzaj, ilość komórek logicznych, matryce komórek pamięci),
- budowy algorytmu, a w szczególności:
  - ilości i rodzaju operacji składowych,
  - ilości i wymiarze operacji nieliniowych,
  - rodzaju i wymiarze operacji liniowych,
  - sposobie dodania podklucza rundy,

Po zaimplementowaniu całości rundy, bądź tez tylko jej elementów składowych, narzędzia projektowe wspomagające wytwarzanie układów cyfrowych (np. MaxPLUS II), pozwalają na przeprowadzenie różnego rodzaju testów.

Najistotniejsze są wyniki na analizatorze czasowym, który pozwala zidentyfikować element, który ma najdłuższy czas opóźnienia sygnału, wyniki raportów z kompilacji, które mówią o tym ile poszczególne elementy rundy szyfru potrzebują zasobów logicznych oraz wyniki symulacyjne, weryfikujące poprawność funkcjonalna.

Kolejnym krokiem jest ocena osiągniętych wyników oraz konfrontacja ich z kryterium optymalizacji projektu.

Jeżeli celem nadrzędnym jest duża przepustowość projektu – należy wówczas zastanowić się nad wyborem odpowiedniej architektury (przedstawione są w dalszej części publikacji), możliwościami realizacji jej w wybranym układzie programowalnym oraz sposobami optymalizacji poszczególnych elementów składowych rundy. Kryteria, jakim należy się posłużyć dotyczyć powinny takich zagadnień jak skrócenie czasu propagacji sygnału cyfrowego poprzez dany element oraz wyrównanie czasu trwania poszczególnych operacji składowych.

Jeżeli celem nadrzędnym jest duża efektywność projektu – należy wybrać architekturę iteracyjną oraz należy przeprowadzić analizę raportów kompilacji. Pozwoli ona na wskazanie elementów wykorzystujących największa ilość zasobów i wskaże nam elementy, od których należy zacząć optymalizacje pod względem efektywności. Należy tutaj zwrócić uwagę na funkcje elementu optymalizowanego. Jeżeli ma ona charakter automatu to należy się zastanowić czy przeprowadzana optymalizacja efektywnościowa nie przyniesie szkody w postaci zwiększenia się opóźnień propagacji sygnałów sterujących, kluczowych z punktu widzenia poprawnego działania projektu.

2. Analiza funkcji generacji podkluczy.

Współczesne szyfry blokowe opierają swoje bezpieczeństwo o dwa algorytmy. Pierwszym jest algorytm mieszania danych (ang. *data mixing*), drugim, nie mniej istotnym, algorytm generacji podkluczy rund (ang. *key schedule*). Potencjalne uchybienie bezpieczeństwa (ang. *security flow*), jakim jest możliwość przeprowadzenia skutecznego ataku z kluczami zależnymi (ang. *related keys attack*), skłania projektantów algorytmów blokowych do konstruowania coraz bardziej skomplikowanych algorytmów generacji podkluczy. Komplikacja tego elementu szyfru nie zawsze idzie w parze ze wzrostem bezpieczeństwa [FURI03], natomiast niemal zawsze ma kluczowy wpływ na szybkość przetwarzania układu scalonego realizującego funkcje kryptograficzna [ROGA03]. Permutowany wybór bitów podkluczy rund prawdopodobnie odejdzie raz na zawsze w momencie, kiedy całkowicie ustąpi standard DES [FIPS46].

Algorytmy generacji podkluczy rund ze względu na strukturę możemy wiec podzielić na dwie grupy:

- dwustopniowe z kluczami przejściowymi (np. Serpent, Camellia, Hierocrypt, Blowfish),
- jednostopniowe (np. Rijndael,).

Implementacja i optymalizacja realizacji obu typów algorytmów, oprócz założeń dotyczących efektywności oraz przepustowości rozwiązania powinna dodatkowo dotyczyć takich zagadnień jak elastyczność implementacji. Pod tym pojęciem rozumiane są: możliwość szybkiej zmiany klucza sesyjnego oraz równomierność procesu szyfrowania i deszyfrowania (długość trwania). Niestety wpływ na tą cechę implementacji ma przede wszystkim konstrukcja szyfru.

Po za tym niezwykle istotnymi są cechy samego algorytmu generacji podkluczy. Z natury rzeczy jego konstrukcja powinna być bardziej efektywna i szybsza niż algorytm mieszania danych. W wielu przypadkach tak nie jest [TOSH01], co stanowi o kiepskich efektach zrealizowanych układów scalonych opartych o dany algorytm blokowy.

Zalecana wiec strategia realizacji projektu opiera się o założenie, ze należy w taki sposób projektować układ scalony, aby najdłużej trwająca operacja była funkcja rundy. Maksymalna częstotliwość taktowania zegara będzie wiec zależna od długości okresu jaki potrzebuje sygnał cyfrowy na pokonanie układu logicznego funkcji rundy. Szybkość działania akceleratora kryptograficznego staje się cechą ściśle związaną z naturalnymi cechami algorytmu blokowego.

Jednostopniowy algorytm generacji podkluczy jest bardzo często bardzo szybkim, jednakże mało elastycznym rozwiązaniem problemu konstrukcji tego typu algorytmów. W algorytmie Rijndael szybkość szyfrowania i deszyfrowania przy częstych zmianach podklucza może różnić się nawet o połowę. Spowodowane to jest faktem, ze klucz sesyjny szyfrowania i deszyfrowania nie jest ten sam. Deszyfrując realizujemy operacje odwrotnego porządku kolejności rund i kolejności podkluczy. Niestety ostatni podklucz możemy wyliczyć gdy mamy poprzedni. W taki sposób dochodzimy do wniosku, ze musimy zrealizować operacje generacji podkluczy dwukrotnie przy każdym deszyfrowaniu. Z klucza głównego wyznaczamy klucz deszyfrowania, następnie wyznaczamy w odwrotnej kolejności podklucze rund.

Na wybór strategii implementacji generacji podkluczy w algorytmie blokowym, maja wiec wpływ cechy przede wszystkim związane z sama jego konstrukcja.

Poniższe zestawienie ma na celu powiązać cechy budowy szyfru blokowego i możliwości jego implementacji. Istnieją trzy podstawowe sposoby realizacji układu cyfrowego realizującego funkcje szyfrowania algorytmem blokowym. Są to strategie szyfrowania/deszyfrowania:

1.	**Z wcześniejszym wyznaczeniem podkluczy.**
Rozróżniane są dwa tryby pracy tego typu implementacji. Tryb ustawienia układu oraz tryb szyfrowania/deszyfrowania. Podczas pierwszego okresu realizowane są wszystkie operacje służące wyznaczeniu wszystkich podkluczy rund, a następnie realizowany jest ciąg operacji szfrujących/deszyfrujących. Istnieją dwa sposoby realizacji tej strategii. Pierwszy polega na wyznaczeniu wszystkich podkluczy poza układem scalonym, następnie umieszczenie ich w rejestrach wewnętrznych służących po przechowywania kluczy rund i rozpoczęcie operacji szyfrowania/deszyfrowania. Drugi natomiast polega na realizacji w ramach układu bloku funkcjonalnego realizującego operacje generacji podkluczy, którego zadaniem jest wyznaczenie wszystkich podkluczy rund oraz zapamiętanie ich w wewnętrznych rejestrach i używanie w czasie procesu szyfrowania/deszyfrowania. Strategia tego typu jest zalecana dla algorytmów, które maja nierównomierny charakter działania poszczególnych elementów szyfru blokowego (np. dużo dłuższa długość trwania algorytmu generacji podkluczy od algorytmu szyfrowania). Także właściwym jest dobranie tej strategii realizacji szyfru blokowego dla projektów, które posiadają wspólne elementy (taka sama runda szyfrowania i wyznaczania pokluczy np. Camellia). Nie ma wówczas redundancji elementów. Po za tym strategia tego typu zalecana jest do rozwiązań w których przetwarzana jest duża ilość danych w stosunku do częstości zmiany klucza głównego – sesyjnego.	

2.	**Z równoczesnym wyznaczaniem podkluczy.**
Strategia zalecana w szczególności dla algorytmów, które posiadają zbliżony czas realizacji rundy szyfru i rundy generacji podkluczy. Polega na jednoczesnym wyznaczaniu rundy szyfrowania i generacji podkluczy do następnej rundy. Działanie układu scalonego realizującego funkcje szyfrowania danym algorytmem rozpoczyna się od wyznaczenia podklucza pierwszej rundy szyfrowania, a następnie już jednocześne wykonywane są runda szyfrowania oraz runda podklucza dla następnej rundy szyfrowania. Zmiana klucza sesyjnego nie ma zbyt wielkiego wpływu na zachwianie procesem szyfrowania czy deszyfrowania (np. Rijndael).	

3.	**Z częściowym wyznaczeniem podkluczy.**
W czasie okresu, kiedy układ przygotowywany jest do pracy realizowane są operacje, które pozwalają na wyznaczenie części niezbędnych danych do generacji podkluczy. Rozwiązanie zalecane dla szyfrów, których algorytm generacji podkluczy jest dwustopniowy (podklucze przejściowe i podklucze rund np. Serpent) oraz dla takich, w których podstawowa runda szyfru i runda podklucza są takie same (np. Camellia).	

3. Analiza metody implementacji funkcji nieliniowych zawartych w algorytmie.

Znaczenia funkcji nieliniowych dla algorytmu blokowego nie można przecenić, stąd też właściwe jej zaprojektowanie jest jedna z podstaw uzyskania dobrego, odpornego na ataki szyfru blokowego. Eli Biham i Adi Shamir oparli kryptoanalizę różnicową o założenie, że prawdopodobieństwo różnicy wyjściowej dwóch wartości, można wyznaczyć z prawdopodobieństwem jeden, dla zadanej różnicy wartości wejściowych, dla wszystkich operacji, za wyjątkiem tych, które są nieliniowe. Teoretyczna miara odporności jest znalezienie charakterystyki (przejścia różnicowego przez algorytm, bądź jego cześć) o zadanym prawdopodobieństwie. Prawdopodobieństwo to jest wyznaczane poprzez przemnożenie prawdopodobieństwa każdego elementu nieliniowego występującego w wybranej charakterystyce. Praktyczna miara odporności szyfru blokowego jest ilość par tekst jawny – zaszyfrowany, jakie trzeba zgromadzić, żeby moc przeprowadzić skuteczny atak. Ilość takich par jest równa odwrotności prawdopodobieństwa wybranej charakterystyki.

Algorytm pozbawiony operacji nieliniowych nie posiada więc żadnych charakterystyk, które maja prawdopodobieństwo mniejsze od jednego. W związku z czym do praktycznego ataku z wybranym tekstem jawnym potrzeba jednej pary zestawienia tekst jawny – zaszyfrowany, spełniających charakterystykę.

Najpopularniejszym sposobem zapewniania odpowiednich własności nieliniowych jest używanie skrzynek podstawieniowych.

Wymiary skrzynek podstawieniowych stosowanych w szyfrach blokowych:

- 6x4 np. DES,
- 4x4 np. Serpent,
- 8x8 np. Rijndael,
- 8x256 np. CAST256,
- 7x7 np. Camellia,
- 9x9 np. Camellia
- 9x32 np. Mars.

Sposób realizacji skrzynki podstawieniowej jest zależny od: rodzaju układu programowalnego, w jakim chcemy zrealizować algorytm (chodzi tu głównie o rodzaj zasobów, jakie układ posiada) oraz od wielkości skrzynki podstawieniowej.

Pierwszy sposób polega na realizacji skrzynki podstawieniowej w postaci funkcji kombinacyjnej. Można ja na przykład uzyskać wyznaczając za pomocą tablic Karnaugha. Tego typu optymalizacja nie jest optymalna dla rozwiązań w specjalizowanych układach programowalnych, których podstawowym elementem jest komórka składająca się ze stałej ilości wejść i wyjść (np. ALTERA podstawowa komórka ma 4 wejścia jedno wyjscie). Narzędziem które w wydajny sposób umożliwia technologiczne dopasowanie funkcji boolowskiej do układów programowalnych jest dekompozycja funkcjonalna. Jej algorytmy są rozwijane na Politechnice Warszawskiej w Zakładzie Podstaw Telekomunikacji i produktem badań jest oprogramowanie Demain, pozwalające na optymalizacje funkcji kombinacyjnej pod względem efektywności, jak i pod względem ilości poziomów tablic look-up 4x1, co dalej przekłada się na wielkość opóźnienia sygnału cyfrowego na tym elemencie.

Drugi sposób polega na realizacji skrzynek podstawieniowych za pomocą pamięci typu ROM. Wejściowa wartość traktowana jest jako adres tej pamięci natomiast wartość wyjściowa jest zawartością komórki o zadanym adresie. Możliwa jest realizacja tego typu pamięci w wersji synchronicznej (w większości układów programowalnych) oraz asynchronicznej (np. FLEX - ALTERA). Cecha charakterystyczna tego rozwiązania jest stale i zazwyczaj mniejsze niż w przypadku realizacji kombinacyjnej opóźnienie, na tym elemencie logicznym. Najnowsze generacje układów programowalnych posiadają coraz większa ilość dostępnych komórek pamięci, które mogą być użyte do tego rodzaju rozwiązań.

	Sposób realizacji			
Rodzaj skrzynki	f. komb.	DEMAIN	ROM async.	ROM sync.
DES 6x4 (1)	78 LC /13,7ns	23 LC /7,8ns	128 bit /9ns	128 bit /9ns
AES 8x8	350 LC /31ns	253 LC /15,1ns	2048bit / 8ns	2048bit / 8ns
Camellia 8x8 (1)	352 LC /32ns	251 LC /16,2ns	2048bit / 8ns	2048bit / 8ns
Misty 7x7	**36 LC / 13,5ns**	**74 LC / 14,3ns**	896 bit/9ns	896 bit/8ns
Misty 9x9	**71 LC / 15,0ns**	**84 LC / 17,1ns**	4608 bit/9ns	4608 bit/8ns
Serpent 4x4 (1)	4 LC / 9,5ns	4LC / 8,5ns	64 bit / 12,6ns	64 mem bit / 9,9ns
CAST256 8x256	-	- *1)	- *2)	65536 bit /10ns *3)
Mars 9x32	-	- *1)	- *2)	131072 bit /10ns *3)

Tabela 3.1. Zależności parametrów efektywnościowo-szybkościowych od wielkości skrzynki podstawieniowej w układach FLEX10KE.

*1) niekomercyjna wersja oprogramowania DEMAIN posiada możliwość przeprowadzenia dekompozycji funkcjonalnej na funkcjach o wymiarze: 12 wejść, 8 wyjść.

*2) możliwość realizacji skrzynek podstawieniowych w formie asynchronicznej pamięci ROM jest dostępna w układach, które nie posiadają wystarczającej ilości zasobów, żeby zrealizować konfiguracje 9x32 i 8 x256.

*3) skrzynkie podstawieniowe zrealizowane w układach typu Stratix.

4. Rodzaje architektur akcelaratorów kryptograficznych opartych na szyfrach blokowych.

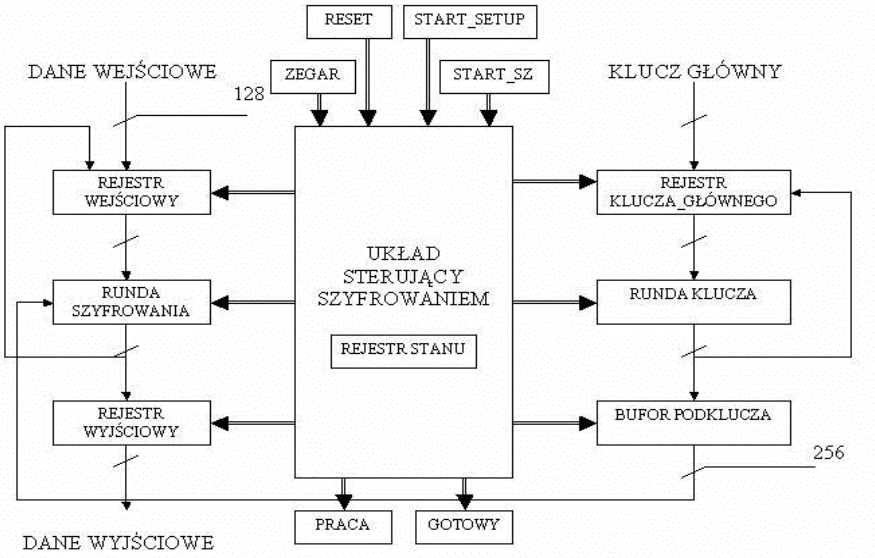
4.1. Architektura Iteracyjna.

Architektura Iteracyjna jest najnaturalniejszym dla szyfrów blokowych sposobem realizacji. Symetryczna kryptografia preferuje symetrie, nie tylko z powodu faktu używania tego samego klucza do szyfrowania i deszyfrowania, ale także z punktu widzenia budowy szyfru (rundy w algorytmie są takie same) i rundy (wszystkie bity wchodzącego do rundy bloku są traktowane tak samo). Z tego tez powodu wypływa niezwykle istotny fakt – nie ma potrzeby implementacji całości szyfru bowiem wystarczy iteracyjne powtórzenie pewnego fragmentu kodu w przypadku rozwiązania programowego lub także iteracyjne przetworzenie sygnału cyfrowego przez dedykowany układ scalony. Z iteracyjnego charakteru tego typu implementacji bierze się nazwa architektury.

Głównymi cechami tego typu rozwiązań są przede wszystkim duża oszczędność zasobów środowiska, w którym realizowany jest algorytm. Poprawność realizacji funkcji szyfrowania i deszyfrowania dla rozwiązań programowych nie wymaga dużej ilości komórek pamięci typu RAM, a dla rozwiązań sprzętowych nie dużej ilości komórek pamięci typu ROM. Omawiana cecha architektoniczna nosi nazwę efektywności implementacji i jest zdecydowanie najlepsza w porównaniu z pozostałymi architekturami.

Kalkulacja kosztów, istoty bezpieczeństwa, możliwości przepustowości medium transmisyjnego, do którego przyłączony jest akcelerator kryptograficzny oraz praktyczna wiedza na temat ilości informacji przepływającej przez układ przekłada się na największa popularność tej architektury.

Na całkowity czas procesu szyfrowania bądź deszyfrowania składa się wiec czas propagacji sygnału cyfrowego poprzez logikę zawarta pomiędzy rejestrami, oraz czas zapisu informacji do rejestru. Szybkość przetwarzania tego typu architektury zależy wiec nie tylko od specyfiki algorytmu blokowego, ale także od ilości operacji zapisu informacji przejściowych (np. wartość szyfrogramu po 3 rundach).



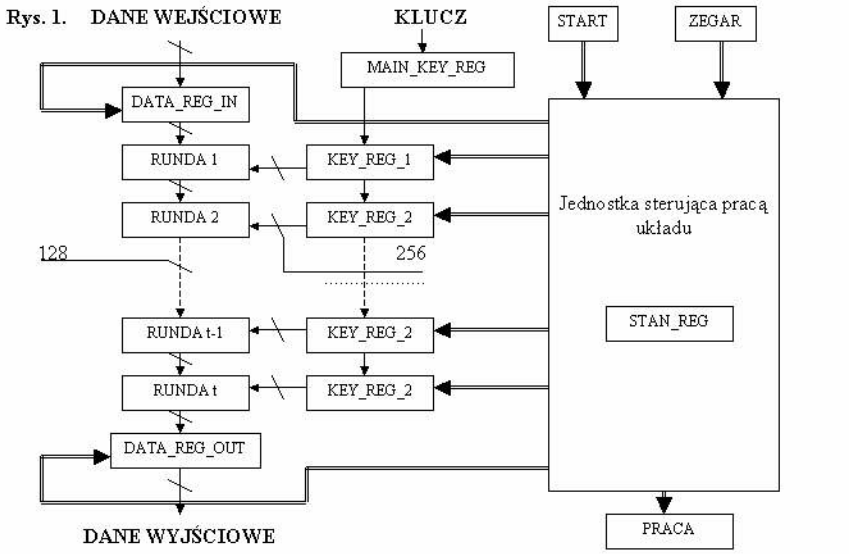
Rys. 4.1. Schemat blokowy architektury Iteracyjnej.

4.2. Architektura Kombinacyjna.

Innym sposobem realizacji algorytmu kryptograficznego to implementacja postaci funkcji kombinacyjnej danych i klucza. W tego typu rozwiązaniach najistotniejszym faktem jest możliwość realizacji całego procesu szyfrowania/ deszyfrowania w ciągu jednego cyklu zegara. Pomijane sa zapisy pośrednich wartości, a wszystkie rundy są fizycznie zaimplementowane w układzie scalonym. Prowadzi to oczywiście do bardzo nieefektywnego wykorzystania dostępnych zasobów, ale także wpływa na szybkość przetwarzania dedykowanego rozwiązania. Czas potrzebny sygnałowi cyfrowemu do przejścia „ścieżki logicznej" skraca się. Na czas trwania procesu szyfrowania jednego bloku danych w architekturze iteracyjnej potrzebny jest czas przejścia sygnału cyfrowego przez wszystkie rundy oraz czas zapisu wartości pośrednich do rejestrów wewnętrznych

Dodatkowa zaleta architektury kombinacyjnej jest najmniej skomplikowany układ sterujący. Szybkość przetwarzania układu scalonego zrealizowanego na bazie tej architektury jest wiec doskonałym wskaźnikiem mówiącym o możliwościach implementacyjnych danego algorytmu.

Największa wada oprócz niskiej efektywności jest fakt, ze architektura tego typu jest idealna tylko dla szyfrów opartych na DES. Generacja podkluczy jest w nich bardzo prymitywna, a co za tym idzie czas wyznaczania ich jest pomijalnie mały w stosunku do czasu potrzebnego na realizacje rundy szyfrowania/deszyfrowania. Można powiedzieć, ze dzięki temu wszystkie podklucze rundy są wyznaczone na czas i wynik operacji kryptograficznej jest poprawny.



Rys. 4.2.: Schamat blokowy Architektury Kombinacyjna

Problem ten może być przezwyciężony poprzez zastosowanie wcześniejszej fazy ustawienia podkluczy do wszystkich rund szyfrowania, albo tez wręcz realizacje algorytmu generacji podkluczy poza układem scalonym. Podczas fazy przygotowania modułu sprzętowego wprowadza się podklucze rund, a po zakończeniu tego procesu można rozpocząć właściwe szyfrowanie/deszyfrowanie w ramach architektury kombinacyjnej.

4.3. Architektura Potokowa.

Procesory o zredukowanym zbiorze instrukcji noszą nazwę RISC (ang. *Reduced Instruction Set Computer*). Wszystkie rozkazy w tego typu procesora charakteryzują się tym, że ich maja jednolita budowę. Cecha ta pozwala na łatwe i szybkie ich dekodowanie. Inna bardzo istotna cecha jest podział cyklu rozkazowego na 5 etapów: pobranie, dekodowanie, pobranie argumentów, wykonanie, zapisanie wyniku. Każdy z nich jest realizowany przez specjalizowana jednostkę, a czas wykonania każdego z etapów jest bardzo podobny. Architektura potokowa w procesorach RISC pozwala wiec na wykonywanie kilku rozkazów w ciągu jednego cyklu zegara, dokładniej zakończenie jednego rozkazu jest realizowane w ciągu jednego cyklu. Problemy, jakie się pojawiają w tego typu rozwiązaniach to zaburzanie potoku przez instrukcje skoków warunkowych.

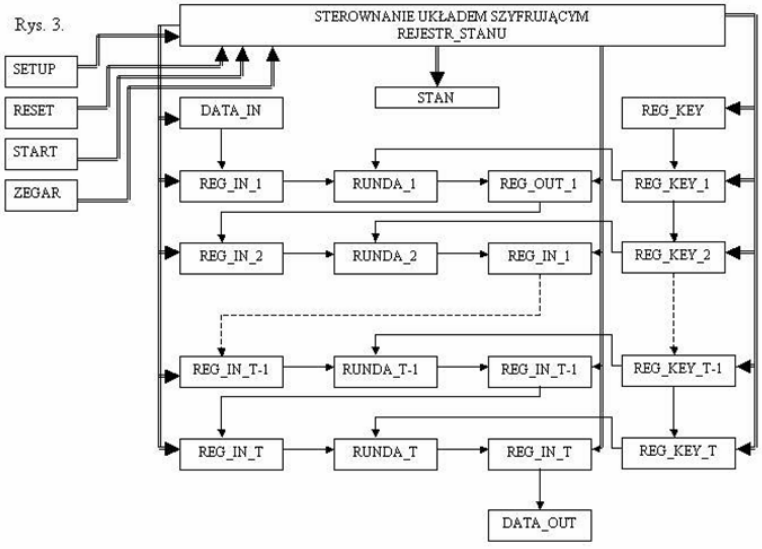
Algorytmy kryptograficzne w większości przypadków składają się z kilku takich samych rund (ewentualnie pierwsza lub ostatnia są nieco inne). Czas propagacji sygnału cyfrowego dla różnych danych dla pojedynczej rundy jest ilość przybliżeniu stały. Można wiec algorytm podzielić na kilka bloków funkcjonalnych, które będą realizowały operacje szyfrowania dana runda. W ten sposób realizowana jest architektura potokowa dla akceleratorów kryptograficznych.

Najważniejszymi zaletami jest ogromna szybkość przetwarzania – rzędu kilku giga-bitów. Tak samo jak ilość przypadku architektury kombinacyjnej ilość każdym cyklu realizowane jest szyfrowanie całego bloku danych. Istotna różnica pomiędzy tymi dwoma rozwiązaniami jest fakt, ze ilość przypadku rozwiązania potokowego cykl zegara jest ilość kilka razy krótszy dzięki czemu cały proces szyfrowania/deszyfrowania jest krótszy i oczywiście przetwarzanie jest dużo większe.

Inna bardzo duża zaleta tego typu rozwiązania jest fakt, ze większość algorytmów kryptograficznych można zrealizować ilość formie potokowej.

Bardzo dużym ograniczeniem jest niezbędna bardzo duża ilość zasobów do realizacji poprawnego procesu szyfrowania i deszyfrowania.

Najpoważniejsza wada jest jednak brak możliwości realizacji innych trybów pracy szyfrów blokowych niż tryb ECB (ang. *Electronic Code Book*).



Rys. 4.3. Schemat blokowy architektury Potokowej

4.4. Architektury Hybrydowe.

W każdej dziedzinie życia często najbardziej przydatnymi rozwiązaniami okazują się być hybrydy. Wiele algorytmów kryptograficznych, takich jak np. Serpent, DES, 3DES, składa się z wielu rund szyfrowania, a jednocześnie implementacja każdej z nich jest na tyle efektywna, że bardziej opłacalnym rozwiązaniem może się okazać realizacja iteracyjno-kombinacyjna.

Najlepszym sposobem ([BOCZ00]) realizacji algorytmu Serpent w środowiskach o ograniczonej liczbie zasobów logicznych okazała się architektura iteracyjno – kombinacyjna. W czasie jednego cyklu zegara realizowanych jest aż 8 rund szyfrowania/deszyfrowania. Rozwiązanie to jest nie wiele bardziej zasobo-chłonne, natomiast szybkość przetwarzania jest tam sześciokrotnie większa od rozwiązania iteracyjnego.

Do podobnych wniosków można dojść także w przypadku wielu innych algorytmów kryptograficznych ([WORC01]).

5. Strategia wyboru odpowiedniej architektury dla postawionych wymagań funkcjonalnych.

Wybór architektury modułu kryptograficznego w akceleratorach sprzętowych okazuje się być ściśle zależny od następujących czynników:

- bezpieczeństwo,
- szybkość,
- efektywność,
- zastosowanie.

Z każdym z tych czynników oczywiście związany jest konkretny układ programowalny, a co za tym idzie jego cena, co jest niestety najważniejszym czynnikiem decydującym.

	Architektura akceleratora kryptograficznego			
	Iteracyjna	**Kombinacyjna**	**potokowa**	**hybrydowa**
Bezpieczeństwo	Pełna możliwość realizacji dowolnych trybów pracy algorytmu blokowych.	Pełna możliwość realizacji dowolnych trybów pracy algorytmu blokowych. Konieczność przechowywania podkluczy rund w wewnętrznej pamięci RAM.	Nie możliwa realizacja popularnych trybów pracy algorytmów blokowych. Rozwiązaniem może być zastosowanie hybryd popularnych trybów pracy, których działanie będzie uzależnione od ilości stopni potoku.	Zależne od wybranego rozwiązania.
Szybkość	Najmniejsza	Pośrednia	Największa	Zależne od wybranego rozwiązania.
Efektywność	Do realizacji poprawnej funkcji szyfrowania lub deszyfrowania potrzebna jest najmniejsza liczba niezbędnych zasobów w stosunku do pozostałych rozwiązań.	Bardzo duża liczba zasobów potrzebna do realizacji funkcji danego algorytmu za pomocą architektury kombinacyjnej.	Bardzo duża liczba zasobów potrzebna do realizacji funkcji danego algorytmu za pomocą architektury iteracyjnej.	Zależne od wybranego rozwiązania.
Zastosowanie	Sieci o maksymalnej przepustowości poniżej 100Mb/s.	Sieci o maksymalnej przepustowości poniżej 100Mb/s.	Sieci Gigabit-Ethernet	Sieci o maksymalnej przepustowości poniżej 100Mb/s

Tabela 6.1: Zestawienie zależności architektury akceleratora kryptograficznego od najistotniejszych cech.

6. Podsumowanie.

O sukcesie w sprzętowej realizacji modułu kryptograficznego decyduje wiele czynników. Możemy je generalnie podzielić na trzy grupy:

- matematyczne,
- technologiczne.
- koncepcyjne,

Pierwsze wynikają z doświadczenia projektowego, z wiedzy z dziedziny algebra liniowa oraz z możliwości samego algorytmu kryptograficznego.

Kolejny czynnik, noszący nazwę technologicznego, wynika z faktu najbardziej prozaicznego - ceny układów reprogramowalnych. Oczywiście najłatwiejsze rozwiązanie to zakupić najdroższy układ trzeciej generacji, który oprócz standardowych komórek logicznych posiada wbudowane matryce pamięci oraz zintegrowane procesory sygnałowe DSP.

Realizacja kryptograficznego akceleratora jest za pewne najłatwiejsza mając do dyspozycji: doświadczenie, wiedze matematyczna oraz najnowsze technologie. Nie mając tego wszystkiego projektant nie jest jednak skazany na porażkę.

Zagadnienie wyboru architektury dla modułu kryptograficznego jest często najważniejszym i decydującym zarówno pod względem efektywnościowym jak i przepustowościowym.

Przedstawione w artykule architektury są wszystkimi najistotniejszymi i najbardziej popularnymi sposobami implementacji algorytmów kryptograficznych. Żadna nie jest doskonała, każda posiada istotne wady, jak i kluczowe zalety.

7. Bibliografia.

[BORA03] Bora Piotr – „Metody sprzętowej implementacji algorytmów kryptograficznych" – 2003r.

[CZER04] Czerwiński Remigiusz – „Analiza Implementacji algorytmu MISTY w strukturach programowalnych"

[LUBA03] Łuba Tadeusz – „Synteza Układów Cyfrowych"- 2003r.

[LUBA00] Łuba Tadeusz – Wykłady dotyczące dekompozycji funkcjonalnej (www.zpt.pw.waw.pl)

[NTMI01]. Nippon Telegrach and Telephone Corporation, Mitshubishi Electric Corporation – „Specyfication of Camellia –a 128 bit block cipher" – 2001r.

[BIKN98] Biham Eli, Knudsen Lars - Serpent

[FIPS197] Deamen Joan, Rijmean Vincent – "Advanced Encryption Standard" – 2001r.

[TOSH01] TOSHIBA Corp. – "Specyfication of Block Cipher:Hierocrypt-3" – 2001r.

[LAMA] Lai X., Massey J.L. - "The IDEA Block Cipher",

[RIVE00] Rivest R. – "RC6 Block Cipher" – 2000r.

[WORC01] Elbrit AJ, Yip W., Chatwynd B., Paar C. – „An FPGA Implementation and Performance Evaluation" of the AES Block Cipher Candidate Algorithm Finalists

[SCHN] B.Schneier – "Kryptografia dla praktykow"

[BISH91] Biham E., Shamir A. – "Differential Cryptanalysis of DES-like systems" – 1991r.

[FIPS46] DES – fips 46,

[MISZ03] Michał Misztal – "Projektowanie szyfrów blokowych" – 2002r. (cykl wykładów),

[FURI03] S. Furuya, V. Rijmen – "Observation on Hierocrypt-3/L1 key scheduling algorithm" – 2003r.

[ROGA03] Rogawski Marcin – „Analysis of the Implementation Hierocrypt algorithm (and its comparison to Camellia algorithm) using ALTERA devices".