Piotr Kotlarz
Instytut Mechaniki i Informatyki Stosowanej
Uniwersytet Kazimierza Wielkiego w Bydgoszczy

Zbigniew Kotulski
Instytut Podstawowych Problemów Technik
Polska Akademia Nauk, Warszawa
Instytutu Telekomunikacji
Politechnika Warszawska

Neuronowy układ „dec2bin", perspektywy implementacji w języku Erlang.

W tej pracy prezentowana jest koncepcje neuronowego dekodera wartości dziesiętnych na binarne. Praca powstała na podstawie mojej rozprawy doktorskiej obronionej IPPT PAN w Warszawie w drugiej połowie 2008 roku. Decyzja o publikacji oryginalnej koncepcji konstrukcji neuronowego dekodera wynika z faktu dalszego rozwijania implementacji neuronowych na potrzeby realizacji funkcji kryptograficznych. Realizacje te były prezentowane na wcześniejszych konferencjach KST. W zakresie dotychczasowych naszych (moich i mojego promotora, którego wymieniam jako współautora referatu) badań i eksperymentów narzędziem implementacyjnym były klasyczne języki programowania (C, C++, Visual Basic) co jak się okazało ograniczało potencjalne możliwości zastosowania proponowanego rozwiązania w praktyce. Niedawno zainteresowałem się środowiskiem programistycznym jakim jest Erlang. Po pobieżnym zainteresowaniu się dostępną (skromną) literaturą na ten temat wydaje się że implementacja sieci neuronowych realizujących funkcje kryptograficzne w Erlangu otwiera nowe możliwości wykorzystania proponowanego rozwiązania.

1. Wprowadzenie

Jako pierwszą realizację w Erlangu postanowiono zaimplementować blok neuronowy który jest kluczowy jeśli myślimy o neuronowych implementacjach np. algorytmów szyfrujących. Tym blokiem jest neuronowy dekoder wartości dziesiętnych na binarne. O zaletach i motywacjach implementacji neuronowych funkcji kryptograficznych napisane sporo zostało w pracach [2][3][4]. Sam blok dekodera występuje we wszystkich neuronowych implementacjach algorytmów szyfrujących. Więc on jako pierwszy został wytypowany do przetestowania w nowej implementacji, w języku Erlang. Jest to też o tyle ważny wybór że sam układ neuronowych został skonstruowany właśnie w sposób kaskadowy i funkcyjny co ma wiele wspólnego z filozofią programowania w Erlangu. Sama koncepcja układu dekodera jako moje oryginalne rozwiązanie warta jest opublikowania, w formie osobnej pracy naukowej, tym bardziej że dotąd publikowana nie była. Usprawni to również dalsze publikacje już z etapu implementacji dzięki możliwości powoływania się na prace wcześniejsze. Z całą pewnością podczas wystąpienia na konferencji KST'2010 będę też mógł przedstawić już pierwsze wyniki eksperymentów zrealizowanych w języku Erlang i porównać je z „klasycznymi" implementacjami.

2. Współczesne implementacje algorytmów szyfrujących

Pojawienie się komputerów, a co ważniejszej, ich rozpowszechnienie spowodowało duże zapotrzebowanie na oprogramowanie umożliwiające szyfrowanie przechowywanych i przesyłanych informacji. Z punktu widzenia bezpieczeństwa danego szyfru duże znaczenie ma nie tylko jakość samego algorytmu, ale również środowisko w jakim ten algorytm pracuje. Zatem implementacja algorytmu kryptograficznego ma kluczowe znaczenie dla bezpieczeństwa. W praktyce większość ataków na systemy szyfrowania wykorzystuje różnego rodzaju luki i niedociągnięcia w implementacjach np. atak na szyfr AES w implementacji OPENSSL [5]. Często jest tak, że w toku badań i dyskusji środowiska kryptologów powstają dwie wersje szyfru. Jedna na potrzeby implementacji sprzętowej, a druga programowej. Przykładem może tu być rodzina szyfrów strumieniowych opartych na generatorze FCSR[6],[7]. W pracy [8] zaproponowano konstrukcję FCSR dla konstrukcji sprzętowych, natomiast w pracy [9] proponowane są dwie wersje szyfru X-FCSR-128 i X-FCSR-256 dostosowane do implementacji programowych. Bezpośrednią inspiracją do podjęcia prac na konstrukcją neuronowego układu szyfrującego była praca dotycząca dwurundowego algorytmu AES w strukturach programowalnych [10]. Układy rekonfigurowane posiadają strukturę komórkową. Analogie do tej budowy można zaobserwować również w neuronowych układach szyfrujących.

3. Neuronowy dekoder wartości dziesiętnych na dwójkowe

Istotnym elementem neuronowego układu szyfrującego jest omawiany dalej dekoder. Jest on istotnym elementem neuronowej implementacji nieliniowego przekształcenia jakim jest S-blok [1]. Zawartość poszczególnych komórek S-bloku stanowią wartości zmiennoprzecinkowe, jest to rozwiązanie wygodniejsze z punkt widzenia realizacji neuronowej. Charakter działania układu neuronowego jest taki, że zawsze będzie na wyjściu udzielał odpowiedzi w postaci wartości przybliżonych. Z eksperymentów i doświadczeń przeprowadzonych w toku badań okazało się, że działanie sieci neuronowej, operującej na wartościach zmiennoprzecinkowych, użytej do realizacji S-bloku jest zasadne pod względem wydajności całego układu. Oczywiście odpowiedź S-bloku jest zawsze w postaci binarnej. Wobec tego koniecznym stało się opracowanie układu neuronowego, który będzie realizował zamianę wartości zmiennoprzecinkowej na binarną. Zakładając, że cały projektowany układ ma być realizowany za pomocą sieci neuronowej, zaistniała więc konieczność konstrukcji neuronowego dekodera. Układ taki na potrzeby przejrzystości dalszego wywodu nazwany został: „dec2bin”. Realizacja neuronowa pozwoli również na łatwość wprowadzania zmian w działaniu układu w zależności od wprowadzonych modyfikacji w innych częściach sieci

realizującej S-blok. Projektowany układ „dec2bin" posiada jedno wejście, na które podawana będzie odpowiedź z wcześniejszego modułu [1] implementacji s-bloku, w postaci dziesiętnej. „Dec2bin" na wyjściu ma udzielić odpowiedzi w postaci 4 bitowej, binarnej. Tablica prawdy dla projektowanego układu, przedstawiona jest w tabeli 1,

Tabela 1 Tablica prawdy, kodowanie 16 wartości dziesiętnych do postaci binarnej.

x1	y1	y2	y3	y4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Ważnym założeniem jest fakt, aby konstrukcja układu kodującego była uniwersalna i mogła być w przyszłości rozbudowywana. W tym celu przyjęta została zasada budowy modułowej, dzięki czemu możliwa jest łatwa modyfikacja układu np. ze względu na wielkość zbioru możliwych sygnałów wejściowych, a co za tym idzie długość binarnego sygnału na wyjściu.

Pierwszym krokiem w rozważaniach jest sprowadzenie problemu kodowania do realizacji zadania klasyfikacji, typowego dla sieci neuronowych. Jeśli zamiast jednej tablicy prawdy skonstruować cztery osobne tabelice zestawiając odpowiednio $x_1 : y_1, x_1 : y_2, x_1 : y_3, x_1 : y_4$, to możemy zaobserwować, że funkcje, które opisują te tabele realizują podział sygnału wejściowego na dwie klasy. Tak więc zadanie konstrukcji neuronowego układu kodującego polegać będzie na konstrukcji czterech odrębnych sieci, które będą realizowały poszczególne funkcje. Ich charakterystyki zapisane zostały w tabeli 2 oraz na rysunku 1.

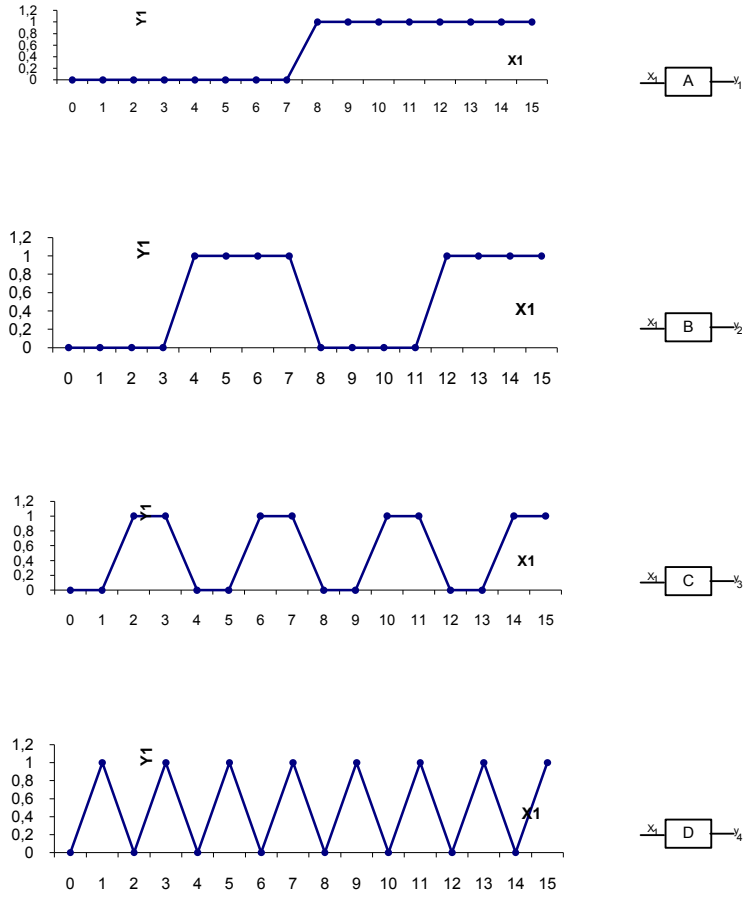
Tabela 2. Tablice prawdy dla układu „dec2bin".

A	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	y1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

B	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	y2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

C	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	y3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

D	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	y4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

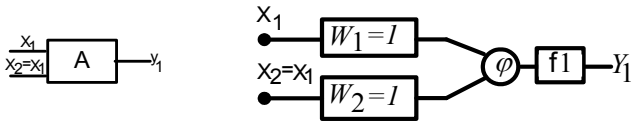


Rys. 2 Bloki elementarne sieci „dec2bin”.

Zidentyfikowane więc zostały cztery bloki elementarne, kolejnym krokiem jest ich implementacja neuronowa.

Realizacja bloku A

Ten problem jest stosunkowo prostym do realizacji neuronowej, nie wymaga nawet przeprowadzania procesu uczenia i wystarczy pojedynczy neuron, opisany na rysunku 3.

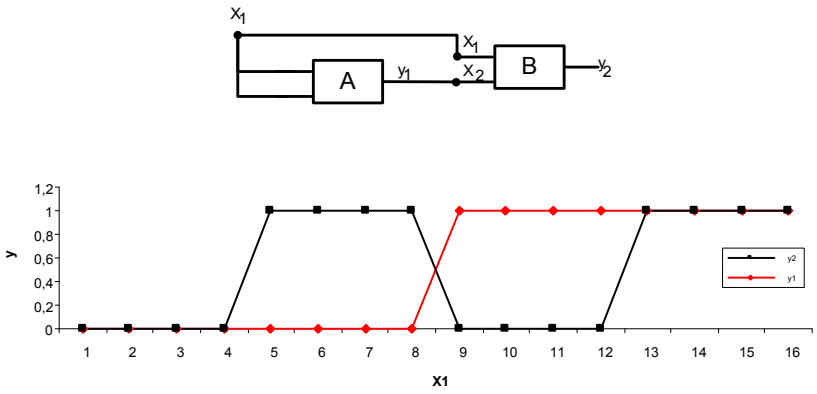


$$f = \begin{cases} 0, & \varphi \le p_1 \\ 1, & \varphi > p_1 \end{cases}, \quad \varphi = \sum w_i x_i, \quad p_1 = 7$$

Rys. 3 Sieć realizująca blok A.

Realizacja bloku B

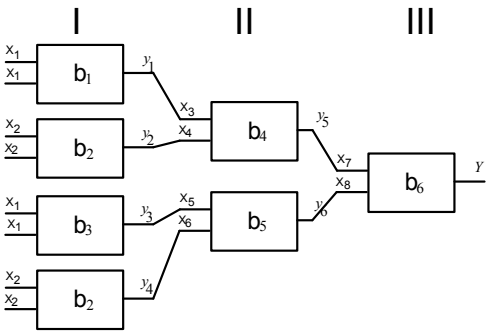
W tym przypadku zadanie jest nieco bardziej złożone, jeden neuron już nie wystarczy. Punktem wyjścia do rozważań jest zweryfikowanie, jakie informacje będą podawane na wejście projektowanego układu. Na pewno na wejście sieci „*B*” podawana będzie wartość zmiennoprzecinkowa (ta sama, która jednocześnie znajdzie się na wejściu układu „*A*”). Ponadto do dyspozycji mamy też odpowiedź układu „*A*”. Układ dokonuje podziału wartości zmiennoprzecinkowej na dwie grupy. Sieć, która jest obecnie przedmiotem rozważań ma za zadanie również dzielenie sygnału wejściowego na grupy. Jak widać na rysunku 4 odpowiedź układu „*B*” jest zależna od wartości podawanej na wyjściu bloku „*A*”.



Rys. 4 Zależność pomiędzy układami B i A.

Ta zależność właśnie zostanie wykorzystana do budowy układu *B*. Problem można więc rozpatrywać w taki sposób, że mamy podział sygnału wejściowego na dwie grupy (moduł „*A*”) oraz w każdej grupie podział na dwie podgrupy (moduł „*B*”). W celu realizacji modułu „*B*” skonstruowana została trzywarstwowa siec neuronowa. Poniżej (tabela 3) przedstawiona jest ogólna

koncepcja tej sieci. Tabela 3 zawiera tablice prawdy oddające zasadę działania funkcji realizowanej przez układ „ B ”. Schemat budowy modułu B pokazany jest na rysunku 5.



Rys. 5 Budowa modułu B.

Tabela 3 Tablice prawdy dla modułu B.

Warstwa I

x1	x2	y1	y2	y3	y4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	1	0	0	0
5	0	1	0	0	0
6	0	1	0	0	0
7	0	1	0	0	0
8	1	1	1	0	1
9	1	1	1	0	1
10	1	1	1	0	1
11	1	1	1	0	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1

Warstwa II

x3	x4	x5	x6	Y5	Y6
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
1	0	0	0	1	0
1	0	0	0	1	0
1	0	0	0	1	0
1	0	0	0	1	0
1	1	0	1	0	0
1	1	0	1	0	0
1	1	0	1	0	0
1	1	0	1	0	0
1	1	1	1	0	1
1	1	1	1	0	1
1	1	1	1	0	1
1	1	1	1	0	1

Warstwa III

x7	x8	Y
0	0	0
0	0	0
0	0	0
0	0	0
1	0	1
1	0	1
1	0	1
1	0	1
0	0	0
0	0	0
0	0	0
0	0	0
0	1	1
0	1	1
0	1	1
0	1	1

B

x1	x2	Y
0	0	0
1	0	0
2	0	0
3	0	0
4	0	1
5	0	1
6	0	1
7	0	1
8	1	0
9	1	0
10	1	0
11	1	0
12	1	1
13	1	1
14	1	1
15	1	1

Tabela 3 zawiera tablice prawdy dla poszczególnych warstw składających się na sieć neuronową modułu B . Na poszczególne warstwy składają się moduły elementarne, rysunek 5, których zasada działania zostanie teraz przestawiona. Moduł b_2 to neuron, którego zadaniem jest przekazanie na wyjście tego samego sygnału jaki podawany jest na jego wejście. Moduł $b_1 i b_3$ to pojedyncze neurony realizujące funkcję progową, odpowiednio z progiem $p = 3$ oraz $p = 1$. Elementy b_1, b_2, b_3 nie wymagają procesu uczenia, ich wagi przyjmują wartości 0 i 1. Elementy b_4, b_5, b_6 wymagają już przeprowadzenia procesu uczenia. Do realizacji poszczególnych elementów wystarczy również jeden neuron posiadający dwa wejścia i jedno wyjście. W tabeli 4 przedstawione zostały tablice

prawdy, jakie mają realizować poszczególne bloki b_4, b_5, b_6. Jednocześnie są one reprezentacją zbiorów uczących, z tym, że dla polepszenia jakości procesu uczenia w zbiorach uczących wartości 0 zostały zastąpione wartościami -1.

Tabela 4 Tablice prawdy dla modułów b4, b5, b5.

b4

x1	x2	Y
0	0	0
0	1	0
1	0	1
1	1	0

b5

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

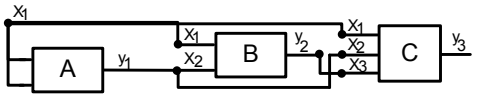
b6

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Tak więc układ składający się z elementów b1 do b6 realizuje funkcję zapisaną jako tabela prawdy układu B.

Realizacja bloku C.

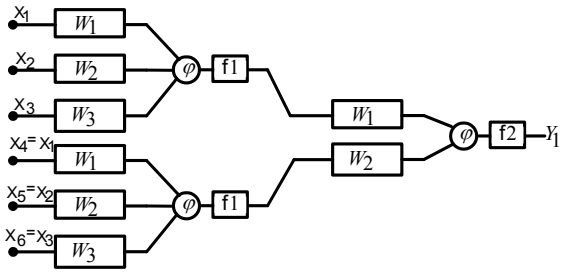
Realizacja kolejnego bloku wymaga użycia sieci neuronowej składającej się z dwóch warstw. Proces uczenia sieci przeprowadzany jest dla całej sieci od razu bez potrzeby „rozbijania" na podbloki. Na tym etapie realizacji kodera „dec2bin" na wejściu bloku „C" dysponujemy trzema różnymi wartościami. Dostępna jest wartość zmiennoprzecinkowa, wynik działania bloku A i bloku B, co obrazuje rysunek 6.



C	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	X2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	X3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	y3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1

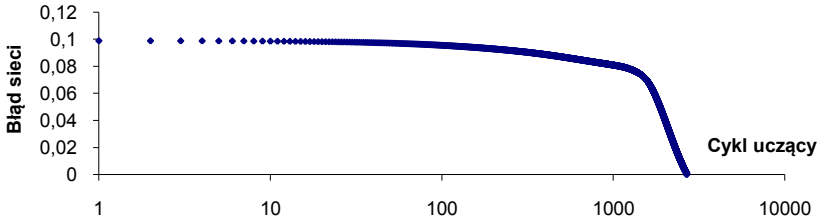
Rys. 6 Schemat działania modułu C.

Przy wykorzystaniu trzech sygnałów wejściowych (x1, x2, x3) realizacja tabeli prawdy pokazanej na rysunku 6 możliwa jest za pomocą sieci neuronowej, składającej się z trzech neuronów i używając powyższe tablice prawdy jako zbioru uczącego. Szczegółowa konstrukcja sieci pokazana jest na rysunku 7. Rysunek 8 przedstawia wykres odzwierciedlający proces uczenia.



$$f_1 = \varphi,\ f_2 = \begin{cases} 0, & \varphi \geq p_1 \\ 1, & \varphi < p_1 \end{cases},\ \text{gdzie: } \varphi = \sum w_i x_i,\ y_j = f_k(\varphi),\ p_1 = 0{,}5,$$

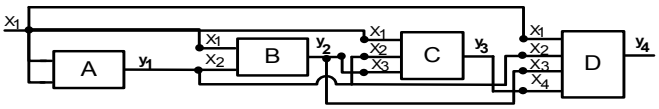
Rys. 7 Sieć neuronowa realizująca moduł C.



Rys. 8 Przebieg procesu uczenia sieci neuronowej realizującej moduł C.

Realizacja bloku D

Ostatni blok neuronowego kodera wartości dziesiętnych do postaci binarnej zrealizowany został zgodnie z taką samą ideą jak blok C. Różnicą jest tu fakt, że w pierwszej warstwie sieci wykorzystane zostały dwa neurony posiadające cztery wejścia. W celu wykorzystania czterech sygnałów pochodzących z wyjść układów A, B i C oraz wartości dziesiętnej, która poddawana jest kodowaniu.



Rys. 9 Schemat układu „dec2bin”

Tabela 6 Tablica prawdy dla modułu D

D	X1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	X2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	X3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	X4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	Y4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Tworząc sieć z przygotowanych bloków elementarnych A, B, C i D uzyskujemy sieć neuronową, rysunek 6, która realizuje kodowanie wartości zmiennoprzecinkowych do postaci binarnej czterobitowej. Układ „dec2bin" realizuje więc funkcję, której tablica prawdy przedstawiona została w tabeli 6.

Podsumowanie

Wyżej opisana koncepcja, realizacja neuronowego dekodera wartości dziesiętnych na binarne jest istotnym elementem budowy neuronowych układów szyfrujących. Układ dec2bin jak i neuronowa realizacja algorytmów DES nie mogą konkurować z klasycznymi implementacjami pod względem wydajności. Mają jednak inne zalety o których była mowa w już cytowanych naszych wcześniejszych pracach. Obiecujący natomiast wydaje się być kierunek związany z implementacjami w języku Erlang, szczególnie pod względem wydajności układu.

[1] NBS FIPS PUB 46, "Data Encryption Standard", 1977

[2] P. Kotlarz, Z. Kotulski, "On application of neural networks for s-boxes design.", Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg 2005, ISBN: 3-540-26219-9

[3] P. Kotlarz, Z. Kotulski, "Neural network as a programmable block cipher – experimental results", Advances in Information Processing and Protection, Springer-Verlag ISBN: 978-0-387-73136-0, 2007

[4] P. Kotlarz, „Znaczenie właściwości funkcji boolowskich w kryptografii.", Przegląd Telekomunikacyjny, ISSN 1230-3496 str 609-616, wrzesień 2007

[5] D. J. Bernstein, "Cache-timing attacks on AES", ENIGMA 2005

[6] A. Klapper, M. Goresky, "2-adic Shift Register. Fast Software Encryption", Second International Workshop. Lecture Notes in Computer Science, vol. 950, Springer Verlag, N. Y., pp.174-178, 1994

[7] A. Klapper, J. Xu, "Algebraic Feedback Shift Registers", (submitted to Elsevier Preprint), 2003

[8] F. Arnault, T.P. Berger, and C. Lauradoux. "Update on F-FCSR stream cipher.", ECRYPT Network of Excellence in Cryptology, Call for stream Cipher Primitives Phase 2 2006

[9] F. Arnault, Thierry P. Berger, C. Lauradoux, "X-FCSR - A New Software Oriented Stream Cipher Based Upon FCSRs", INDOCRYPT 2007

[10] V. Fischer, "Realization of the round 2 AES candidates using Altera FPGA", April 2000