DuD
Datenschutz und Datensicherheit

DES auf FPGAs

Hochgeschwindigkeits Architekturen für den Data Encryption Standard auf Rekonfigurierbarer Hardware

Jens-Peter Kaps, Christof Paar

Algorithmenunabhängige Sicherheitsprotokolle stellen eine große Herausforderung für Hard- und Software Implementationen dar.Auf der einen Seite sind sie in der Regel zu aufwendig für eine reine Hardware-Lösung, auf der anderen Seite ist eine Software Lösung in vielen Fällen zu langsam. Rekonfigurierbare FPGAs stellen eine vielversprechende Alternative da.

In diesem Artikel werden diverse Architekturoptionen für DES auf FPGAs gezeigt. Das wohl interessanteste Resultat ist, daß Datenraten von bis zu 402.7 Mbit/s für DES auf FPGAs erzielt werden können

[FOTO]

Dipl.-Ing., MSc., Jens-Peter Kaps

arbeitet bei GTE CyberTrust Solutions Inc.(Needham, Massachusetts / USA). Seine Arbeitsgebiete sind der Secure Electronic Transaction (SET) Standard, Public Key Infrastructure und Zertifizierungsstellen.

E-Mai: Jens-Peter.Kaps@cybertrust.gte.com

1 Einleitung

Moderne kryptographische Übertragungsprotokolle bieten eine Reihe von kryptographischen Algorithmen zur Auswahl an, d.h. sie sind Algorithmenunabhängig. Der hierfür erforderliche Wechsel zwischen verschiedenen Algorithmen kann sehr einfach in Software realisiert werden, aber ist schwierig in Hardware zu erreichen. Andererseits zeichnen sich Hardware-Implementationen durch eine wesentlich höhere Geschwindigkeit und bessere physikalische Sicherheit aus. Die Lösung dieses Problems sind programmierbare Hardware-Bausteine; sogenannte *Field Programmable Gate Arrays (FPGAs)*. Ein FPGA besteht im wesentlichen aus einem See von Logikzellen (Sea of Gates), deren Funktion durch Programmieren verändert werden kann (z.B. Flip Flop, logisches UND mit 6 Eingängen, usw.). Auch die Verbindungen dieser Logikzellen können durch Programmieren gesetzt werden. FPGAs haben die folgenden Vorteile:

- **Algorithmen Flexibilität**, dasselbe FPGA kann im laufenden Betrieb umprogrammiert werden, es kann somit mehrere Algorithmen unterstützen,
- **Skalierbare Sicherheit**, wird erreicht durch verschiedene Versionen desgleichen Algorithmus (z.B. DES und Tripple-DES),
- **Veränderbare Architektur Parameter**, z.B. variable Schlüssellängen oder unterschiedliche Operations-Modi können einfach realisiert werden,
- **Effizienz**, derselbe Baustein kann für symmetrische und asymmetrische Verschlüsselungsverfahren verwendet werden.

Der Daten Verschlüsselungs Standard oder *Data Encryption Standard (DES)* ist ein sehr weit verbreiteter Algorithmus. Er ist Teil vieler Datenübertragungs-Standards, wie z.B. IPSec, ATM Zellen Verschlüsselung, Secure Socket Layer (SSL) und auch dem neuen Transport Layer Security (TLS) Standard. Obwohl DES bald durch den Advanced Encryption Standard (AES) ersetzt werden wird, wird DES auch in den nächsten Jahren eine wichtige Rolle spielen.

Moderne kryptographische Datenübertragungsprotokolle werden nicht nur zum sicheren Übertragen einzelner Nachrichten eingesetzt, sondern mehr und mehr zum Verschlüsseln von allen Datenströmen, z.B. ATM (asynchronous transfer mode) Netzwerke für Telephon und Datenfernübertragung, Virtual Private Networks (VPN) für das abhörsichere Verbinden zweier Firmen-Netze durch das Internet usw.. Diese Anwendungen stellen hohe Anforderungen an die Datenübertragungsraten und somit an die Verschlüsselungsraten.

Dieser Beitrag zeigt eine kurze Übersicht über DES und beschreibt anschließend verschiedene Architekturen zur effiezienten Implementation von DES auf FPGAs gefolgt den Testergebnissen.

2 Vorhergegangene Arbeiten

Die frühesten Veröffentlichungen zum Thema: Implementation von DES in Hardware sind [6] und [11]. Die maximale Verschlüsselungsraten betrugen zwischen 4.72 Mbit/sec und 20 Mbit/sec. Referenz [8] beschreibt eine Implementation in 3 µm Technologie, die bereits 32 Mbit/sec erreichte.

Moderne Hardware Implementationen erreichen Geschwindigkeiten von bis zu

1 Gbit/sec [2] und 1.4 Gbit/sec [5]. Hierbei wurde GaAs Technologie verwendet.

Die erste Veröffentlichung, die sich mit einer Realisierung von DES auf FPGAs befaßte ist [9]. Hier wird die DES Architektur für einen Schlüssel optimiert, der nur durch umprogrammieren gewechselt werden kann. Es wurden Datenraten von bis zu 26 Mbit/sec erreicht. Eine neue, äußerst vielversprechende Technologie sind *Dynamically Programmable Gate Arrays* (DPGAs)[4]. Sie erlauben einen Wechsel des Algorithmus in nur einem Taktzyklus. Moderne FPGAs benötigen mehrere 10 Millisekunden aufgrund der beschränkten Bandbreite der externen Speicher Bausteine [15].

3 DES Algorithmus

DES ist ein sogenannter Blockchiffre, d.h. er verschlüsselt und entschlüsselt jeweils einen Datenblock, im Gegensatz zu Stromchiffren, die den Datenstrom Bit für Bit verschlüsseln. Bei DES sind die Datenblöcke 64 bit lang und der Schlüssel ist 56 bit lang. Die folgende Funktionsbeschreibung wird die internen Funktionen hervorheben, die für eine Hochgeschindigkeits-Implementation in einem FPGA besonders wichtig sind. Für eine vollständige Beschreibung sei der Leser auf die Bücher [12] und [13] verwiesen.

3.1 Die Hauptfunktionen

Der Klartext x wird am Eingang von DES einer Permutation, der sog. *initial permutation (IP)*, unterzogen. Das Resultat nennen wir x_0. Es wird in einen linken Teil L_0 und einen rechten Teil R_0 aufgeteilt. Dieses liegt am Eingang des sogenannten *Feistel Netzwerks* an. Das Feistel Netzwerk hat eine iterative Struktur, d.h. dieselbe Funktion wird mehrmals nacheinander durchlaufen, wobei der Ausgang eines Durchlaufes der Eingang des nächsten Durchlaufes ist. Die Abbildung 1 zeigt eine Runde (Iteration) des DES Feistel-Netzwerkes die 16 mal durchlaufen wird.

Das Zeichen $\oplus$ steht für ein logisches Exklusiv-Oder von zwei Bit-Strings. Das f deutet die f-Funktion an.

Die f-Funktion ist in Abbildung 2 dargestellt. Sie expandiert die 32 Eingangs-Bits zu 48 Bit (durch duplizieren einzelner Bits) und wendet ein Exklusiv-Oder auf sie und die 48 Bits von der Schlüssel-Transformation an. Das Ergebnis wird

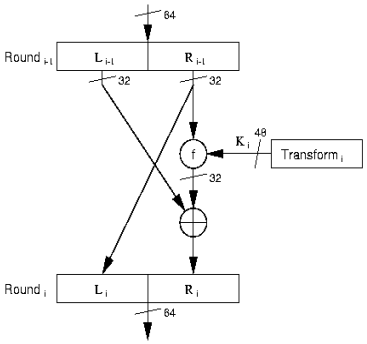

Abb. 1: Eine Runde von DES

durch 8 S-Boxen geleitet. Die S-Boxen sind Tabellen, die einem 6 Bit-Eingangswert einen 4 Bit-Ausgangswert zuordnen. Die 32 Bit am Ausgang der S-Boxen werden permutiert.

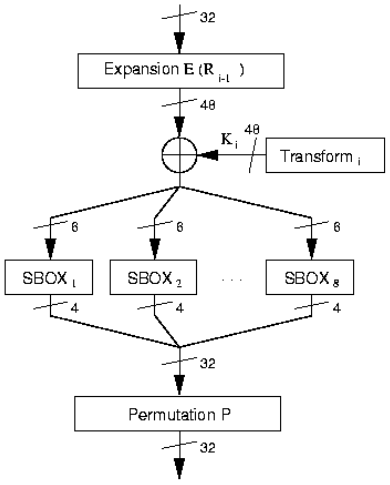

Abb. 2: Die f-Funktion

3.2 Der Schlüsselgenerator

Jede Runde von DES braucht einen eigenen Teil-Schlüssel. Diese werden von dem 64 Bit langem Schlüssel K abgeleitet. K enthält 8 Paritäts-Bits, die effektive Länge ist somit 56 Bit.

Die Teilschlüssel-Generierung ist ebenso ein iterativer Prozeß und benötigt 16 Iterationen. Abbildung 3 zeigt eine Runde der Schlüsselgenerierung. Der Schlüssel K passiert eine Permutationsstufe *P1* und wird dann in zwei Hälften zu je 28 Bit gespalten: C_0 und D_0. Je nach Iteration werden C und D um ein oder zwei Positionen rotiert. Das Ergebnis ist das Eingangssignal der nächste Runde, und wird anschließend durch eine Permutationsstufe *P2* geleitet wodurch der Teilschlüssel entsteht.

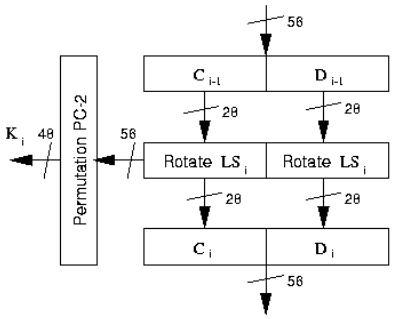

Abb. 3: Eine Runde der Schlüssel Generierung

4 Architekturen

DES basiert auf einer iterativen Struktur. Das Feistel-Netzwerk als auch die Schlüsselgenerierung haben eine elementare Funktion, die 16 mal durchlaufen wird. Aus dieser Grundstruktur läßt sich das in Abbildung 4 dargestellte Blockdiagram herleiten.

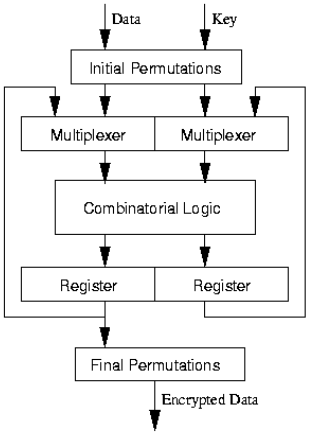

Abb. 4: DES Block Diagram

Sowohl das Feistel Netzwerk als auch die Teilschlüssel-Generierung sind in dem Block *Combinatorial Logic* (CLU, combinatorial logic unit) zusammen gefaßt. Um den Ausgang der CLU wieder an den Eingang zu leiten, werden *Register* und *Multiplexer* benötigt. Die Multiplexer schalten den Eingang der CLU zwischen dem Ausgang der Eingangspermutationen (*Initial Permutations*) und dem Ausgang der CLU um. Die *Register* speichern das

Ergebnis am Ausgang der CLU und leiten es dem *Multiplexer* wieder zu. Das Ausgangssignal des Daten Registers gelangt auch durch die abschließende Permutation (*Final Permutation*). Eine Kontroll-Logik signalisiert nach außen, ob der Ausgang der *Final Permutation* gültig ist (das ist nach 16 Durchläufen der CLU der Fall).

4.1 Loop Unrolling

Unter *Loop Unrolling* wird das *Ausrollen* mehrerer Runden verstanden, d.h. während eines Taktzyklus werden zwei Runden von DES berechnet. Dieses kann durch aneinanderhängen von zwei CLUs erreicht werden (siehe Abbildung 5).

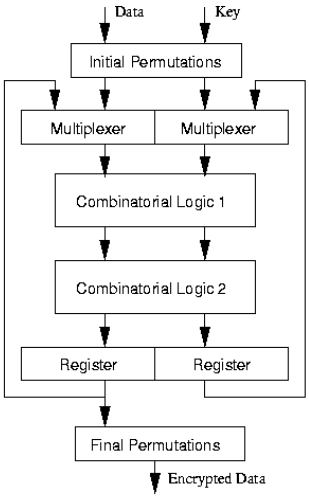

Abb. 5: Loop Unrolling

Für eine Iteration von DES gilt das Zeitmodell: $T_{mux} + T_{cl} + T_{reg}$, wobei T_{mux} für die Signallaufzeit durch den Multiplexer, T_{cl} durch die CLU und T_{reg} durch die Register steht. Es wird angenommen, daß die Signal Verzögerungen, die durch Verdrahtung entstehen, Teil der jeweiligen Blöcke sind. Für DES mit 16 Runden ergibt sich nun die folgende Gleichung:

$$16 * T_{mux} + 16 * T_{cl} + 16 * T_{reg}$$

Bei einfachem *Loop Unrolling* (zwei aufeinanderfolgende Runden) werden bei jeder Iteration 2 CLUs ausgeführt. Dadurch werden für DES nur noch 8 Taktzyklen benötigt und die Gleichung sieht so aus:

$$8 * T_{mux} + 16 * T_{cl} + 8 * T_{reg}$$

$$4 * T_{mux} + 16 * T_{cl} + 4 * T_{reg}$$

Dieses Prinzip kann auch auf vier ausgerollte Runden angewendet werden:

Die Zeitverzögerungen, die durch die CLUs eingeführt werden, können nicht verringert werden, aber die Anzahl der Multiplexer und Register Durchläufe können verringert werden. Moderne Schaltungssynthese Programme können solche ausgerollten Schaltungen durch *boundary optimization* besser optimieren.

4.2 Pipelining

Das *Pipelining* ist eine weitere Architekturoption, mit der Geschwindigkeitsverbesserungen erziehlt werden können. Hierbei werden zwei oder mehr Datenblöcke gleichzeitig verarbeitet. Abbildung 6 zeigt eine zweistufige Pipeline.

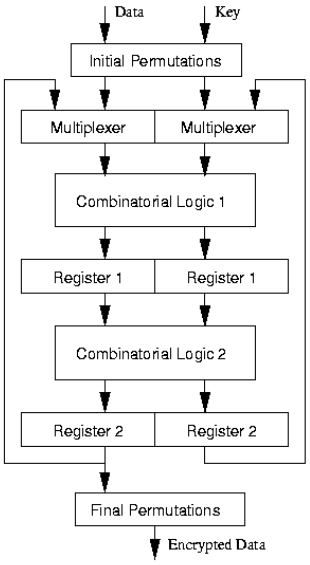

Abb. 6: Pipeline

Ein Datenblock x_1 und der dazu gehörige Schlüssel k_1 gehen durch die *Initial Permutations* und den Multiplexer, die CLU1 berechnet das Ergebnis der ersten Runde, welches in das *Register 1* abgelegt wird. Beim nächsten Taktzyklus gelangt das Ergebnis an den Eingang der CLU2, welche das Ergebnis der zweiten Runde berechnet. Dieses gelangt in die *Register 2*. Zur selben Zeit wird ein neues Paar x_2 und k_2 geladen, deren erste Runde berechnet und in das *Register 1* abgelegt. Jetzt ist die Pipeline mit zwei Daten-Schlüssel-Paaren gefüllt.

Bei jedem weiteren Taktzyklus werden die nächsten Runden beider Paare berechnet. Sobald das erste Paar 16 Runden absolviert hat, kann ein neues Paar in die Pipeline geladen werden.

Der Vorteil dieser Architektur ist, daß zwei oder mehr Daten-Schlüssel-Paare gleichzeitig berechnet werden können. Im Gegensatz zu einer zweifachen Implementation von DES steigt der Resourcen-Bedarf hier um weniger als Faktor 2, da die Eingangs- und Ausgangspermutationen sowie die Multiplexer nur einmal vorhanden sind. Zudem wird nur eine Kontroll-Logik benötigt.

Die maximale Taktfrequenz sollte ähnlich der für ein Design ohne Pipeline sein, da die Signale pro Taktzyklus dieselbe Anzahl von Logik-Bausteinen durchlaufen. Die Pipeline kann zudem einfach auf vier oder mehr Stufen ausgebaut werden.

> Es soll hierbei hervorgehoben werden, daß dieses Design, im Gegensatz zu vielen kommerziellen DES Implementationen pro Datenblock einen Schlüssel laden kann.

4.3 Kombination

Natürlich ist es möglich, die Architekturoptionen *Loop Unrolling* und *Pipelining* miteinander zu verbinden. So wurde hier eine zwei stufige Pipeline mit einfachem Loop Unrolling implementiert.

4.4 Vergleich und Designs

Für DES wurden vier Operationsmodi standardisiert [13, Seite 83]: *electronic codebook mode* (ECB), *cipher block chaining mode* (CBC), *cipher feedback mode* (CFB) und *output feedback mode* (OFB). Die Modi CBC, CFB und OFB benötigen das Ergebnis der vorhergegangenen Verschlüsselung um den neuen Datenblock zu verschlüsseln. Dies ist kein Problem für ein Loop Unrolled Design, mit einem Pipeline Design kann man dieses aber nicht realisieren, da an zwei oder mehreren Datenblöcken gleichzeitig gearbeitet wird. Für den ECB Mode oder den ATM-counter Mode kann jedoch ein Pipeline Design verwendet werden.

Um diese Architektur Optionen zu untersuchen wurden die in Tablle 1 gezeigten Implementationen vorgenommen.

Name	Beschreibung
DES_ED16	Standard DES (16 Iterationen)
DES_ED8	DES mit 2 ausgerollten Runden (8 Iterationen)
DES_ED4	DES mit 4 ausgerollten Runden (4 Iterationen)
DES_ED16x2	DES mit 2 stufiger Pipeline
DES_ED16x4	DES mit 4 stufiger Pipeline
DES_ED8x2	DES mit 2 stufiger Pipeline, jede mit 2 ausgerollten Runden

Tabelle 1: Implementierte DES Architekturen

5 DES Design

Um DES optimal an ein FPGA anzupassen wurde der Algorithmus in Funktionsblöcke zerlegt.

5.1 Funktionsblöcke

Wie oben gezeigt wurde, besteht DES aus Permutationen, Registern, Multiplexern und kombinatorischer Logik. Die kombinatorische Logik beinhaltet das Feistel-Netzwerk und Teilschlüssel-Generierung. Das Feistel-Netwerk ist aus einem 32-bit XOR und der *f*-Funktion aufgebaut, die ihrerseits aus einer Expansion, einem 48-bit XOR, acht S-Boxen und einer Permutation besteht.
Die Schlüsselgenerierung besteht aus Schieberegistern und einer Permutation. Die Schieberegister müssen ihren Inhalt um ein oder zwei Positionen nach links oder nach rechts rotieren, je nach Iteration und Modus (Verschlüsselung, Entschlüsselung).

Somit ergeben sich die folgenden Funktionsblöcke:

- Permutationen und Expansionen
- Register
- Multiplexer
- Standard Logik Funktionen (XOR)
- S-Boxen
- Schieberegister

5.2 Logik Resourcen

Die Funktionsblöcke, die im vorhergehenden Abschnitt identifiziert wurden, müssen nun optimal in einem FPGA implementiert werden.

- **Permutationen und Expansionen** Permutationen vertauschen die Bits eines Bit-Strings. Expansionen vertauschen auch die Bits und verdoppeln einzelne Bits. Beides kann durch „Verdrahten“ realisiert werden.
- **Register** Register können entweder durch Logikelemente oder durch Speicherelemente realisiert werden. Die meisten modernen FPGAs verfügen über RAM/ROM Elemente.
- **Multiplexer** Multiplexer können auch sehr einfach mit kombinatorischer Logik realisiert werden.
- **S-Boxen** Die S-Boxen sind Tabellen der Größe 6 x 4 und beinhalten somit 64 Bit vier-Bit Werte. Die Implementation der S-Boxen hat einen großen Einfluß auf die Geschwindigkeit des Designs [7]. Es ist am effizientesten, die S-Boxen mit ROM Elementen zu realisiern.
- **Schieberegister** Die Schieberegister, die in der Schlüssel Generierung verwendet werden, können sehr einfach durch Multiplexer und „Verdrahtung“ implementiert werden.

5 Resultate

Es wurden die in Tabelle 1 gezeigten Designs implementiert und auf ihre Effektivität hin untersucht. Das Design *DES*_ED16 wurde für viele Vergleiche als Referenz hergenommen. Die Einheit **CLB** steht für *Combinatorial Logic Block* und ist ein Maß für den Resourcen-Verbrauch auf dem FPGA. Die Abkürzung **CLU** steht weiterhin für *Combinatorial Logic Unit*, d.h. eine Runde des Feistel Netzwerkes.

5.1 Loop Unrolling

Die Designs *DES_ED8* und *DES_ED4* wurden implementiert. Das Design *DES_ED8* hat zwei ausgerollte Runden (CLUs) und braucht somit acht Taktzyklen für eine Verschlüsselung, das Design *DES_ED4* hat vier CLUs und verschlüsselt einen Datenblock in vier Taktzyklen. Tabelle 2 zeigt die Ergebnisse.

Design	Chip	CLBs used	CLBs per CLU	Min CLK in ns	Data Rate per CLU in Mbit/s	Data Rate in Mbit/s	Data Rate rel
DES_ED16	XC4008E-3-PG223	262	262	40.4	99.1	**99.1**	1.00
DES_ED8	XC4013E-3-PG223	443	222	54.0	74.1	**148.2**	1.50
DES_ED4	XC4028EX-3-PG299	722	241	86.7	46.1	**184.5**	1.86

Tabelle 2: Ergebnisse von Loop Unrolling

Mit dem einfachen *DES_ED16* Design wurden schon Datenraten von 99.1 Mbit/sec erreicht. Das Design *DES_ED8* ist mit 148.2 Mbit/sec 50% schneller als des Referenzdesign und benötigt 69% mehr Resourcen. Das Design *DES_ED4* ist mit 184.5 Mbit/sec nur 25% schneller als *DES_ED8* und verbraucht 63% mehr Resourcen. Ein nicht linearer Zuwachs der Datenrate kann beobachtet werden. Ein weiteres Ausrollen zu 8 Runden würde somit voraussichtlich keinen besonderen Geschwindigkeitszuwachs ergeben.

5.2 Pipelining

Zwei Pipeline Designs wurden implementiert: *DES_ED16x2* mit einer zweistufigen Pipeline und *DES_ED16x4* mit einer vierstufigen Pipeline. Die Ver- und Entschlüsselung braucht in beiden Fällen 16 Taktzyklen.. Tabelle 3 enthält die Ergebnisse.

Design	Chip	CLBs used	CLBs per CLU	Min CLK in ns	Data Rate per CLU in Mbit/s	Data Rate in Mbit/s	Data Rate rel
DES_ED16	XC4008E-3-PG223	262	262	40.4	99.1	**99.1**	1.00
DES_ED16x2	XC4013E-3-PG223	433	217	43.5	91.9	**183.8**	1.86
DES_ED16x4	XC4028EX-3-PG299	741	185	39.7	100.7	**402.7**	4.06

Tabelle 3: Ergebnisse von Pipelining

Das Design *DES_ED16x2* ist mit 183.8 Mbit/sec fast doppelt so schnell wie das Refenrenzdesign und das Design *DES_ED16x4* ist mit 402.7 Mbit/sec fast vier mal so schnell. Somit ist der Geschwindigkeitszuwachs annähernd konstant. Der Bedarf an Logik Resources per CLU sinkt sogar. Der Grund dafür ist, daß die Multiplexer und die Kontrollogik nur einmal implementiert werden.

5.3 Gemischtes Design

Ein Design, das sowohl Loop Unrolling als auch Pipelining verwendet, ist schon in seiner einfachsten From so groß, wie die größten oben gezeigten Designs. Deshalb wurde nur das Design *DES_ED8x2* implementiert. Es enthält eine zweistufige Pipeline mit je zwei ausgerollten Runden pro Stufe. Tabelle 4 zeigt die Ergebnisse.

Ein Vergleich dieser Ergebnisse ist schwierig. An der minimalen Taktperiode kann man sehen, daß die jeweils zwei ausgerollten CLUs in dem Design *DES_ED8x2* schneller sind als in dem Design *DES_ED8*. Sie sind überraschenderweise nicht viel langsamer als eine einzelne CLU in dem Design *DES_ED16x2*.

6 Vergleich und Fazit

Alle Designs wurden für handelsübliche Bausteine der Firma Xilinx mit einem mittleren Speed-Grade entwickelt.

Mit handelsüblichen FPGAs wurden Geschindigkeiten von bis zu 402.7 Mbit/sec in einem Design mit einer vierstufigen Pipeline erreicht.

Vergleicht man dieses Ergebnis mit der Geschwindigkeit von Hochgeschwindigkeits ASICs (1600 Mbit/sec) [12] und hoch optimierten Software-Implementationen (12 Mbit/sec) [12], so ergibt sich, daß *DES_ED16x4* ca. 30 mal schneller als eine Software Implementation ist und nur 5 mal langsamer als ein ASIC. Obwohl es schwierig ist einen fairen Vergleich zu treffen kann gesagt werden, daß DES auf FPGAs wesentlich schneller als Software-Implementationen ist, und Verschlüsselungsraten erreicht werden können, die für die allermeisten Hochgeschwindigkeitsanwendungen (z.B. 155 Mbit/sec ATM Netzwerke) ausreichend sind.

Design	**Chip**	**CLBs**	**Min CLK** in ns	**Data Rate** p. Pipeline in Mbit/s	**Data Rate** in Mbit/s
DES_ED8x2	XC4028EX-3-PG299	733	48.0	166.5	**333.0**
DES_ED16x2	XC4013E-3-PG223	433	43.5	91.9	**183.8**
DES_ED4	XC4013E-3-PG223	443	54.0	148.2	**148.2**

Tabelle 4: Vergleich von Gemischtem Design mit anderen

Literatur

[1] S.A. Vanstone, A.J. Menezes, and P.C. Van Oorschot. *Handbook of Applied Cryptography*. Discrete Mathematics and its Application. CRC Press, Florida, USA, 1997.

[2] H. Eberle and C.P. Thacker. A 1 Gbit/second GaAs DES chip. In *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pages 19.7/1-4, New York, NY, USA, 1992, IEEE.

[3] A.G. Broscius and J.M. Smith. Exploiting parallelism in hardware implementation of the DES. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91 – Proceedings*, number 576 in Lecture Notes in Computer Science, pages 367-376, Berlin, Germany, 1992. Int. Assoc. Cryptologic Res, Springer-Verlag.

[4] I. Eslick, J. Brown, E. Tau and D. Chen. A first generation DPGA implementation. In *FPD'95 – Third Canadian Workshop of Field-Programmable Devices*, 1995.

[5] H. Eberle. A high-speed DES implementation for network applications. In E.F. Brickell, editor, *Advances in Cryptology – CRYPTO'92. 12th Annual International Cryptology Conference Proceedings*, Lecture Notes in Computer Science, pages 521-539, Berlin, Germany, 1993. Springer Verlag.

[6] J. Goubert, F. Hoornaert, and Y. Desmedt. Efficient hardware implementations of the DES. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO'84*, number 196 in Lecture Notes in Computer Science, pages 147-172, Berlin, Germany, 1985. International Association for Cryptologic Research, Springer-Verlag.

[7] G.M. Haskins. Securing asynchronous transfer mode networks. Masters thesis, WPI, Worcester, Massachusetts, USA, May 1997.

[8] J. Vanderwalle, I. Verbauwhede, F. Hoornaert, and H.J. De Man. Security and performance optimization of a new DES data encryption chip. *IEEE Journal of Solid-State Circuits*, 23(3):647-656, June 1988.

[9] J. Leonard and W.H. Magione-Smith. A case study of partially evaluated hardware circuits: keyspecific DES. In P.Y.K. Cheung, W. Luk, and M. Glesner, editors, *Field programmable Logic and Applications. 7th International Workshop, FPL'97*, Berlin, Germany, 1997. Springer-Verlag.

[10] A. Matusevich, R.C. Fairfield, and J. Plany. An LSI digital encryption processor. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO'84*, number 196 in Lecture Notes in Computer Science, pages 115-143, Berlin, Germany, 1985. International Association for Cryptologic Research, Springer Verlag.

[11] B. Schneier. *Applied Cryptography Second Edition: protocols, algorithms, and source code*, In C. Wiley & Sons, New York, USA, 2nd edition, 1996.

[12] D.R. Stinson. *Cryptography: Theory and Practice*. Discrete Mathematics and its Applications. CRC Press, Florida, USA, 1995.

[13] M.E. Hellman, W. Diffie. Exhaustive cryptanalysis of the NBA Data Encryption Standard. *Computer*, 10:74-84, June 1977.

[14] Xilinx, San Jose, California, USA. *The Programmable Logic Data Book*, 1996

[15] Xilinx, San Jose, California, USA. *Xilinx University Program Workshops*, 1997.