Kryptomodul für schnelle DES basierende Verschlüsselungssysteme

Hagen Ploog, Mathias Schmalisch, Dirk Timmermann

Institut für Angewandte Mikroelektronik und Datentechnik, Universität Rostock

Fachbereich Elektrotechnik und Informationstechnik

Richard-Wagner-Str. 31, 18119 Rostock, Tel.: (0381) 498 35 34

email: hp@e-technik.uni-rostock.de

Abstract. Mit steigender Akzeptanz des Internets als alltägliches Arbeitsmedium steigt auch die Zahl erfolgter und erfolgreicher Einbruchsversuche in Firmennetze und in private Rechner. In diesem Beitrag wird die Entwicklung und der Aufbau eines Kryptomoduls für schnelle DES-basierende Verschlüsselungssysteme beschrieben.

1 Einführung

Mit steigender Akzeptanz des Internets als alltägliches Arbeitsmedium steigt auch die Zahl erfolgter und erfolgreicher Einbruchsversuche in Firmennetze und in private Rechner.

Dadurch ist die sichere Übertragung von und zum Auftraggeber ein wesentlicher Aspekt für Realisierung von Telearbeitsplätzen. Hierbei ist “sicher“ nicht nur im Sinne von fehlerfrei zu verstehen, sondern vor dem Hintergrund der Datenspionage und -sabotage. Ein Weg gegen Datenspionage ist sicherlich der Einsatz von kryptographischen Verfahren. Bei der Auswahl der Algorithmen ist ein Kompromiß zwischen dem Maß der Sicherheit und der Geschwindigkeit der jeweiligen Implementierung zu treffen.

In diesem Beitrag wird die Entwicklung und der Aufbau eines Kryptomoduls für schnelle DES-basierende Verschlüsselungssystem beschrieben.

Der Beitrag gliedert sich wie folgt:

In Kap. 2 geben wir eine kurze Übersicht über Verschlüsselungssysteme und ordnen in diese den DES Algorithmus ein, dessen Funktionsweise wir in Kap. 3 beschreiben. In Kap. 4 beschreiben wir dann die von uns auf einem XILINX FPGA implementierte Variante und geben in Kap. 5 eine Zusammenfassung.

2 Verschlüsselungssysteme

Prinzipien symmetrischer Verschlüsselungssysteme

Bei symmetrischen Verschlüsselungsverfahren wird ein Schlüssel zum Ver- und Entschlüsseln benutzt. Beim Versenden einer Nachricht wird diese zuvor auf der Sendeseite mit dem Schlüssel codiert. Der Empfänger kann dann diese Nachricht mit dem selben Schlüssel wieder dekodieren und somit lesbar machen. Bild 1 zeigt eine Prinzipskizze.

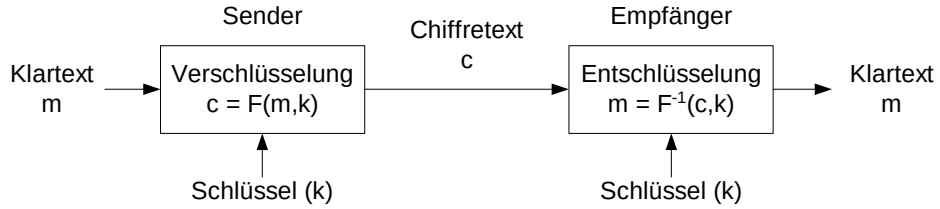

Bild 1 : Prinzip symmetrischer Verschlüsselungssysteme

Der Vorteil symmetrischer Verfahren besteht darin, daß sie gegenüber asymmetrischen Verfahren wesentlich schneller bei der Ver- bzw. Entschlüsselung sind. Allerdings müssen beide Seiten den Schlüssel kennen, wodurch das Problem des sicheren Schlüsselaustausches auftritt. Die Geheimhaltung des Schlüssels ist wesentlicher Bestandteil des Algorithmus. symmetrische Verschlüsselungssysteme werden deshalb auch als *private-key*-Verfahren bezeichnet. Ein Beispiel für symmetrischer Verschlüsselungssysteme ist der DES-Algorithmus, der in Abschnitt 3 genauer behandelt wird und ausführlich in [FUMY94] oder [NBS77] nachzulesen ist.

Prinzipien asymmetrischer Verschlüsselungssysteme

Die asymmetrischen Verschlüsselungsverfahren werden auch als *public-key*-Verfahren bezeichnet, da hierbei ein privater und ein öffentlicher Schlüssel zum Einsatz kommt. Den privaten Schlüssel kennt nur der Empfänger der Nachricht, während der öffentliche Schlüssel an alle anderen Kommunikationsteilnehmer versendet wird. Wenn eine Nachricht an den Empfänger versendet werden soll, wird diese zuvor mit dessen öffentlichen Schlüssel verschlüsselt. Der Empfänger kann dann diese Nachricht mit seinem privaten Schlüssel wieder entschlüsseln. Dieses Prinzip ist in Bild 2 dargestellt.

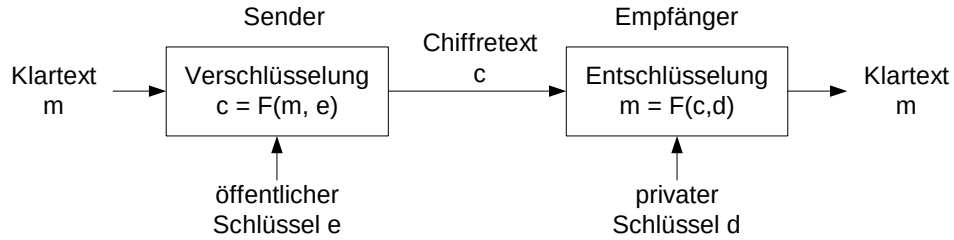

Bild 2 : Prinzip asymmetrischer Verschlüsselungssysteme

Der Vorteil asymmetrischer Verschlüsselungsalgorithmen besteht in der Bekanntgabe des öffentlichen Schlüssels, denn mit diesem kann die Nachricht nicht entschlüsselt werden. Nachteilig gegenüber symmetrischen Verfahren ist die Geschwindigkeit beim Ver- bzw. Entschlüsseln. Die Verfahren zum Generieren des Schlüsselpaares müssen garantieren, daß keine Rückschlüsse auf den privaten Schlüssel durch Bekanntgeben des öffentlichen Schlüssel möglich sind. Ein Beispiel für ein asymmetrisches Verfahren ist der RSA-Algorithmus [MEN97].

Prinzipien hybrider Verschlüsselungssysteme

Bei hybriden Verschlüsselungsverfahren werden jeweils ein symmetrisches und asymmetrischen Verfahren kombiniert und somit die jeweiligen Vorteile ausgenutzt. Das asymmetrische Verfahren wird eingesetzt, um den symmetrischen Schlüssel sicher auszutauschen. Dann wird das symmetrische Verfahren verwendet, um schnell die Daten zu verschlüsseln.

3 Funktionsweise DES-Algorithmus

Beim DES handelt es sich um eine symmetrische Blockchiffre. Von einer Blockchiffre spricht man, wenn die Nachricht *m* in Abschnitte fester Länge (Blöcke) geteilt wird, und diese dann nacheinander und unabhängig voneinander verschlüsselt werden. Dabei wird ein 64-Bit Schlüssel auf einen 64-Bit Klartext- (bzw. Chiffretext-) Block angewendet. Das 8. Bit jedes Schlüsselbytes ist ein Paritätsbit, so daß sich die wirksame Schlüssellänge damit auf 56-Bit verkürzt. Bild 3 zeigt die Struktur des DES. Ein zu verschlüsselnder Block unterliegt zuerst der Eingangspermutation IP (*initial permutation*). Daraus entstehen zwei 32-Bit breite Teilblöcke L und R, wobei L die höheren und R die niederen 32-Bit enthält. Anschließend folgen 16 Iterationen, bei denen auf den R-Block immer dieselbe Funktion, nur mit jeweils einem anderen 48-Bit Teilschlüssel, angewendet wird. Nach jeder Runde enthält der neue L-Block den vorherigen R-Block und der neue R-Block die auf den alten R-Block angewendete Funktion f XOR-verknüpft mit dem vorherigen L-Block:

$$L_i = R_{i-1} \qquad 1 \le i \le 16$$

$$R_i = L_{i-1} \otimes f(R_{i-1}, K_i)$$

Nach den 16 Verschlüsselungsschritten wird die Ausgangspermutation IP^{-1} auf die beiden erhaltenen 32-Bit Blöcke angewendet. Die Entschlüsselung verläuft analog zur Verschlüsselung. Es werden lediglich die Teilschlüssel in umgekehrter Reihenfolge auf die Funktion f angewendet.

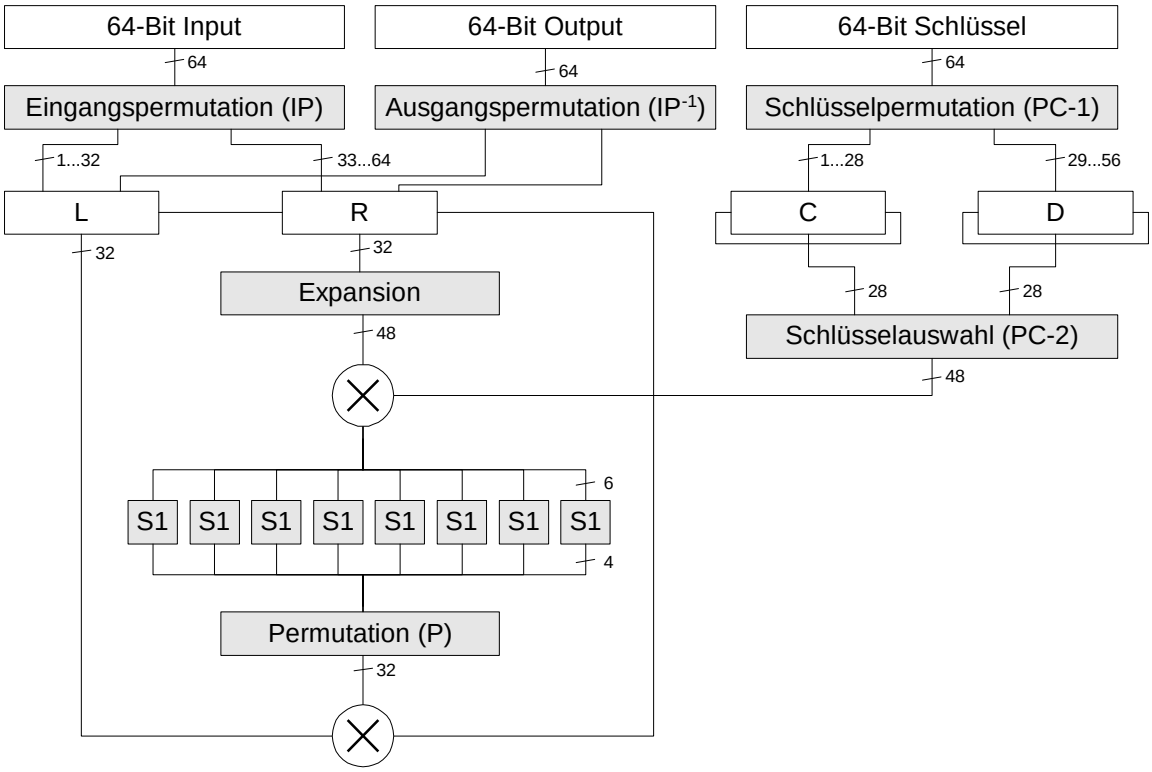

Bild 3 : Struktur des DES-Algorithmus

Die f-Funktion

In jeder der 16 Verschlüsselungsrunden wird zunächst der 32-Bit breite R-Block durch die Expansion auf 48-Bit erweitert. Diese werden dann mit dem 48-Bit Teilschlüssel XOR-verknüpft. Die resultierenden 48 Bit werden dann in acht 6-Bit Blöcke zerlegt, und diese dienen als Input für die acht S-Boxen. Jede der acht S-Boxen substituiert einen 6-Bit Input-Block durch einen 4-Bit Output-Block. Tabelle 1 zeigt S-Box 1.

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	0:	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1:	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2:	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3:	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tabelle 1 : S-Box 1

Dabei entscheiden das erste und letzte Input-Bit über die Zeilennummer und die mittleren 4 Bits über die Spaltennummer. Der dortige Wert wird binär auf die 4 Output-Bits gelegt. So entsteht wieder, mit den acht anderen S-Boxen zusammen, ein 32-Bit Block, der dann der P-Permutation unterworfen wird.

Schlüsselauswahl

Der 64-Bit Schlüssel wird zuerst der Permutation PC-1 (*permuted choice*) unterworfen (Bild 4). Dabei wird der Schlüssel auf 56 Bits verkürzt, da jedes 8. Bit ausgelassen wird. Diese 56 Bits werden dann in zwei 28-Bit Blöcke zerlegt, die C und D genannt werden. Der C-Block enthält dann die höheren und der D-Block die niederen 28 Bits. Vor jeder der 16 Iterationen werden die C- und D-Blöcke zyklisch um ein oder zwei Bit nach links geschoben. Die von der jeweiligen Iteration abhängige Anzahl der zu verschiebenden Bits ist in Tabelle 2 wiedergegeben.

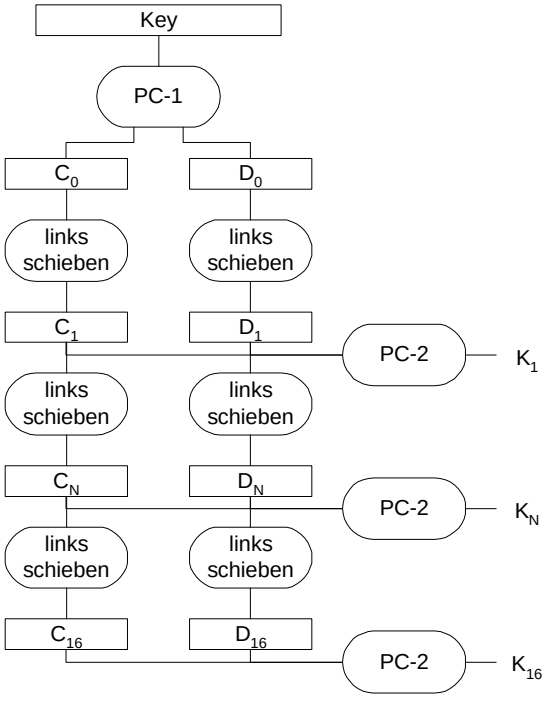

Bild 4 : Schlüsselauswahl

Insgesamt werden 28 Schiebeoperationen während der 16 Runden durchgeführt, so daß sich die Register C und D zum Schluß wieder in ihrem Ausgangszustand befinden. Dadurch sind die Register bei unveränderten Schlüssel nicht erneut zu laden, und weitere Chiffrier- oder Dechiffrieroperationen können durchgeführt werden. Nach der Schiebeoperation in jeder Iteration wird aus den Registern C und D durch die PC-2 Permutation ein 48-Bit Schlüssel generiert. Dieser wird dann mit dem expandierten R-Block XOR-verknüpft und wirkt so auf den Eingang der S-Boxen.

Iteration:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Anzahl Shifts:	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabelle 2 : Anzahl der Links-Shifts

3.1. Modi des DES-Algorithmus

Um den verschiedenen Anforderungen im praktischen Einsatz gerecht werden zu können, wurden unterschiedliche Betriebsarten für Blockchiffren entwickelt. Diese Modi wurden 1980 im Zusammenhang mit der Standardisierung des DES vom NBS genormt und in der FIPS *Publication* 81 veröffentlicht [NBS80]. Sie unterscheiden sich im Hinblick auf die erreichbare Verschlüsselungsrate, die Fortpflanzung von Übertragungsfehlern, die Auswirkung von Synchronisationsfehlern und den Schutz gegen bestimmte Angriffe. Die vier Standard-Modi sind:

- ECB-Modus (*Electronic Code Book*)
- CBC-Modus (*Cipher Block Chaining*)
- OFB-Modus (*Output Feedback*)
- CFB-Modus (*Cipher Feedback*)

Aufgrund immer leistungsfähiger Rechnernetze ist es in den letzen Monaten mehrmals gelungen, mit DES-verschlüsselte Nachrichten zu dekodieren. Anzumerken ist dabei, daß der als relativ einfach zu brechende ECB-Mode zur Verschlüsselung benutzt wurde. Sicherheitslücken dieser Art lassen sich mit Hilfe des Triple DES, auch DES3 genannt, schließen, der in Bild 5 skizziert ist..

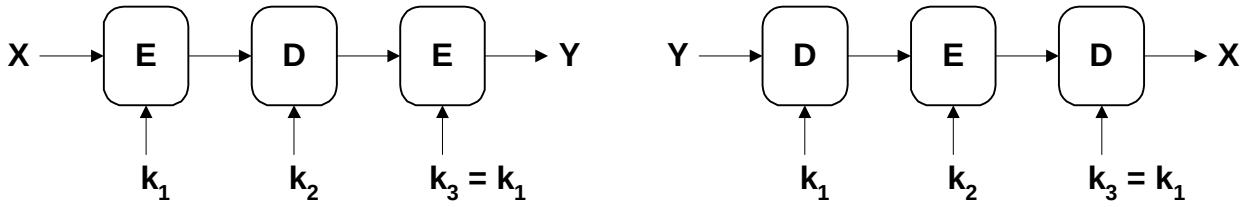

Bild 5 : DES3-Ver~ und Entschlüsselung

Die Schlüssellänge bei DES3 in dieser Variante beträgt 2^{128}, was auf absehbare Zeit nur durch Zufallstreffer zu brechen sein wird.

3.2. Software-Performance

Die normalerweise am Arbeitsplatz vorzufindende Hardwareausstattung liegt heute bereits im Leistungsbereich kleinerer Workstations. Die Effizienz der implementierten Algorithmen ist letztendlich ein Maß dafür, ob eine zusätzliche Hardware erforderlich ist oder nicht. Die Effizienz läßt sich am Beispiel des DES-Algorithmus über den Durchsatz (KByte / sec) bestimmen. Neben der Taktfrequenz des Prozessors beeinflußt auch das zur Verfügung stehende Betriebssystem den Datendurchsatz.

Prozessor	Taktfrequenz in MHz	Betriebssystem	Bytes pro Sekunde
AMD-K6	266	Windows NT 4.0	3.671.913
UltraSPARC	200	SunOS 5.5	2.854.358
UltraSPARC	167	SunOS 5.5	2.367.668
Pentium MMX	200	Windows NT 4.0	2.179.991
AMD-K6	266	MS-Dos 6.22	1.483.833
Pentium	133	Windows NT 4.0	1.402.339
Pentium	133	MS-Dos 6.22	431.157

Tabelle 3: Vergleich unterschiedlicher Computersysteme

3.3. Anforderungsprofil an Hardware-Lösungen

Um den Einsatz von zusätzlicher Hardware zu rechtfertigen, muß gegenüber der Software-Variante eine deutliche Leistungssteigerung erreicht werden. Damit die Kosten der Hardware-Lösung nicht deren Vorteile aufheben, haben wir uns für eine flächenminimale Implementierung entschieden, die dadurch auch in relativ kleinen FPGAs realisiert werden kann. Der serialisierte DES-Algorithmus benötigt für die Ent- bzw. Verschlüsselung von 8 Bytes 16 Takte. Der Datendurchsatz läßt sich dadurch wie folgt berechnen:

$$Durchsatz\ (MBytes/s) = \frac{fclk(MHz)\cdot 8}{16}\cdot\frac{1000^2}{1024^2}$$

Eine Steigerung des Datendurchsatzes gegenüber einer Software-Implementierung läßt sich bereits bei einer zugrunde gelegten Taktrate von 10 MHz nachweisen, die mit heutigen FPGAs problemlos und sehr kostengünstig realisierbar sind..

3.4. Optimierungen in HW

Beim DES-Algorithmus erfolgt die Verschlüsselung iterativ, d.h., durch Einfügen von Zwischenspeichern können die einzelnen Runden jeweils von einander entkoppelt werden. Auf Kosten zusätzlicher Hardware kann deshalb bei gleicher Taktfrequenz der 16fache Durchsatz erzielt werden. Diese Technik wird als *loop-unrolling* bezeichnet.

4 DES-Coprozessor

In [SCH98] wurde ein DES-Coprozessor aufgrund von Voruntersuchungen entwickelt und in synthesefähigem VHDL beschrieben. Nach der Synthese mit SYNOPSYS design analyzer 1998.08 wurde die erzeugte Netzliste mit XILINX M1.5 auf einen XC4020-3 abgebildet.

4.1. Lösung im FPGA

FPGAs sind freiprogrammierbare Bausteine, die die Grundzellen digitaler Logik bereits enthalten. Die Programmierung besteht darin, die entsprechenden Zellen auszuwählen und derart untereinander zu verbinden, daß die gewünschte Schaltungslogik entsteht.

Innerhalb des FPGAs ist eine gewisse Anzahl von Grundzellen zu größeren Einheiten, den CLB (konfigurierbaren Logik Blöcken), zusammengefaßt. Die CLBs sind in Zeilen und Spalten über das FPGA verteilt. FPGAs werden heutzutage in zwei Standardsituationen eingesetzt:

- *Rapid prototyping*

  Durch den Einsatz von FPGAs können Schaltungen mit bis zu ca. 100 MHz im System getestet werden, ohne daß ein ASIC vorhanden sein muß.
- *Kleinserien*

  Die Entwicklung von ASICs ist immer noch relativ kostenintensiv. Durch die Größe von FPGAs können inzwischen Schaltungen mit bis zu 1 Millionen Gattern kostengünstig implementiert werden. Die Stückzahl und damit die Kosten pro Stück bestimmen letztlich, ob eine Schaltung als ASIC oder FPGA realisiert wird.

4.2. Aufbau

Im folgenden beschreiben wir nun die Umsetzung der einzelnen Module des DES-Algorithmus.

Permutationen

Die Permutationen IP und IP^{-1} lassen sich in einer HW-Implementierung fast kostenfrei realisieren, da sie nur aus dem Vertauschen von Bitpositionen untereinander bestehen.

Die Schlüsselauswahlfunktion

Bei der Schlüsselauswahlfunktion benutzt man jeweils ein C- und D-Register zum Ver- und Entschlüsseln, wobei diese dann als Links- und Rechtsschieberegister dienen.

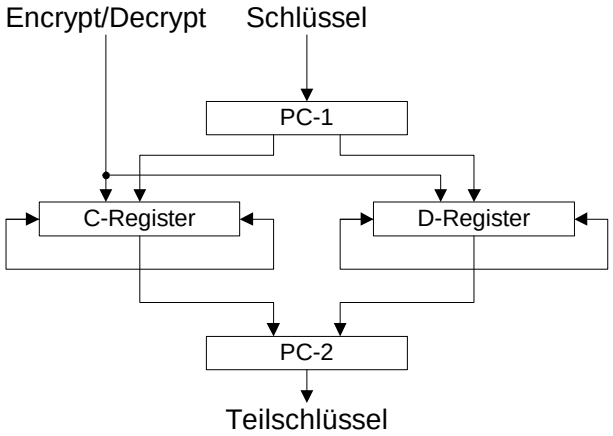

Bild 6 : Realisierung der Schlüsselauswahlfunktion

Die S-Boxen

Jede S-Box kann als ein ROM (*Read Only Memory*) mit 64 Adressen und jeweils 4 Bit Daten dargestellt werden. Um ein solches ROM in einem FPGA zu implementieren, existieren 3 Möglichkeiten:

- Beschreibung des ROMs in VHDL
- Verwendung von ROM-Primitiven
- Verwendung eines ROM-Generators

Bei einer Beschreibung des ROMs in VHDL wird dieser dann auf die zur Verfügung stehende Standardlogik abgebildet, was zu einem Design von 22 bis 60 CLBs pro S-Box führt.

Wesentlich flächeneffizienter und damit kostengünstiger lassen sich ROMs mittels ROM-Primitiven beschreiben. Dazu werden entsprechende Bibliothekselemente als Instanzen in den VHDL-Code eingebunden und im Synthesescript dann mit den entsprechenden Werten geladen.

Für ein 64 x 4 ROM werden acht dieser 32 x 1 ROM-Primitiven benötigt. Dabei werden vier für die höheren und vier für die unteren 32 Adressen genutzt, deren Datenausgänge dann mit Hilfe eines Multiplexers entsprechend am Ausgang durchgeschaltet werden können (Bild 7). Ein so aufgebautes Design benötigt genau 10 CLBs, wobei 8 auf die ROM-Primitiven und 2 auf den Multiplexer verfallen. Somit benötigen die 8 S-Boxen bei einer optimalen FPGA-Implementierung insgesamt 80 CLBs.

Die Verwendung des ROM-Generators besaß gegenüber der Verwendung von ROM-Primitiven keinen Vorteil.

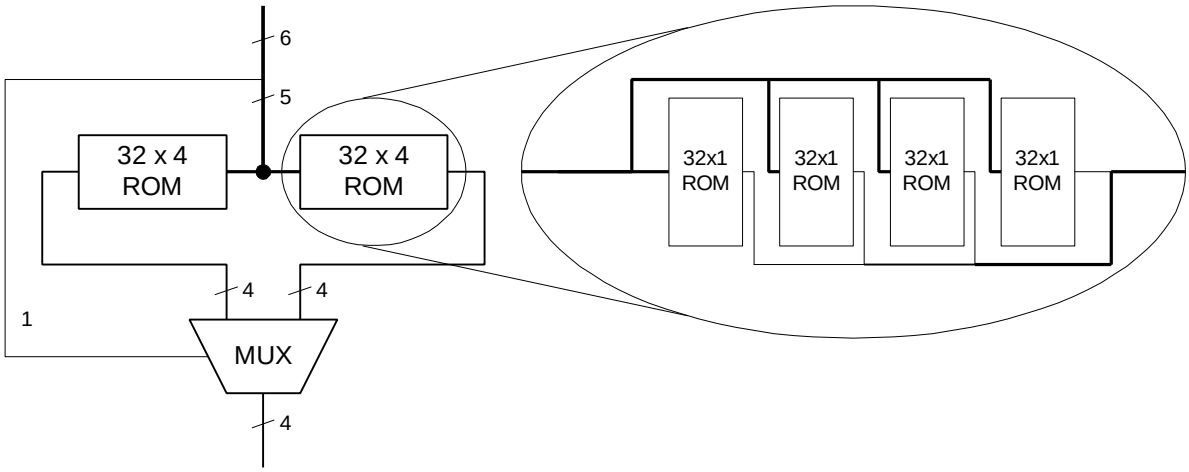

Bild 7 : Struktur einer S-Box

4.3. Performance

Der am Institut für Angewandte Mikroelektronik und Datentechnik der Universität Rostock entwickelte DES-Kern kann in der vorliegenden Implementierung bis 23.5 MHz getaktet werden und erreicht dabei einen Durchsatz von $94 \cdot 10^6$ Bit/s. Durch *loop-unrolling* läßt sich der Durchsatz um den Faktor 16 erhöhen.

4.4. Erweiterung um 8051 Interface

Kleine eingebettete Systeme verfügen nicht über die Rechenleistung von Arbeitsplatzrechnern. Um den DES-Kern auch in diesem Umfeld einsetzen zu können, ist in einem zweiten Schritt ein Interface-Schaltung für den Anschluß an einen 8051 µController entwickelt worden. Eine Performance-Analyse in [PLO98] zeigt, daß hier Leistungssteigerungen um den Faktor 115 gegenüber den schnellsten kommerziell erhältlichen Software-Implementierungen erreichbar sind.

5 Zusammenfasung

Am Institut für Mikroelektronik und Datenverarbeitung der Universität Rostock ist ein DES-Kern für schnelle Verschlüsselungssysteme entwickelt worden. Zu Demonstrationszwecken wurde der DES-Kern auf einem FPGA vom Typ XC4020-3 implementiert. Durch die Realisierung in einem FPGA läßt sich dieser Kern gleichermaßen auf einer Beschleunigungskarte in einem PC als auch als eigenständiger Coprozessor für kleine eingebettete Systeme einsetzen. Die Verschlüsselungsrate beträgt in der aktuellen Version bis $94 \cdot 10^6$ Bit/s. Der Kern benötigt 227 CLBs, inklusive des µContoller-Interface werden 419 CLBs benötigt. Da entspricht einem Auslastungsgrad des XC4020 von 53%.

Neben dem µController-Interface ist auch der ECB und OFB Mode implementiert.

Der DES-Kern samt 8051-Interface liegt in einer synthese-fähigen VHDL-Beschreibung vor, wodurch eine schnelle Portierung in andere Zieltechnologien gewährleistet ist.

Literaturverzeichnis

[FUMY94] Fumy, W.; Rieß, H. P.:
Kryptographie – Entwurf, Einsatz und Analyse symmetrischer Kryptoverfahren,
2.Auflage, Oldenbourg Verlag, München, 1994

[NBS77] National Bureau of Standards
Data Encryption Standard
FIPS Publication 46, 1977

[MEN97] Menezes, A. J.; van Oorschot, P. C.; Vanstone, S.A.:
Handbook of Applied Cryptography
CRC Press, Boca Raton, 1997

[NBS80] National Bureau of Standards
DES Modes of Operation
FIPS Publication 81, 1980

[SCH98] Schmalisch, M.:
Untersuchung und Implementierung des DES-Verschlüsselungsalgorithmus,
Universität Rostock, Kleiner Beleg, 1998

[PLO98] Ploog, H.
Hardwarelösungen für Smart Card Kryptographie
Vortrag im Rahmen des Graduiertenkolleg "Verarbeitung, Verwaltung, Darstellung und Transfer multimedialer Daten – technische Grundlagen und gesellschaftliche Implikationen", Universität Rostock, Rostock, Nov. 1998