The Rogue Chip: A Bidirectional Data Compress-Encrypt Chip

1997 Student VLSI Contest

Experienced Class Entry

Steve Haynal

Mike Torres

Advisor: Forrest Brewer

The University of California, Santa Barbara

Abstract

A high performance data compress/encrypt and decrypt/decompress chip has been designed and fabricated as a project for an Advanced VLSI design class at the University of California, Santa Barbara. The Rogue chip, designed in HP's CMOS26G 0.8 μm technology, is designed to perform the Data Encryption Standard, DES, algorithm with a throughput of 40 megabytes per second. A novel data compression algorithm matches the high DES throughput while still providing usable, entropy increasing compression. The design was created using structural and behavioral VHDL. Cascade Design Automation's Epoch generated a floorplan, placement and route from the VHDL specification. Epoch's manual intervention capabilities were used extensively to fine-tune the design to meet area, performance and verification constraints. Thorough simulations, using Epoch's VHDL model of the chip, a custom VHDL test bench and Mentor Graphic's QVSIM, were performed. DRC and LVS was done with Mentor Graphic's Checkmate tools. The resulting 53,458 transistor, 3349 μm × 3238 μm die design was fabricated by MOSIS. Testing of the 65 pin PGA package utilized the Rogue chip's built in self-test and a custom microcontroller based test setup. Testing confirmed functionality of the DES subsection to 33 megabytes per second. This paper describes in detail the design, implementation, simulation and testing of the Rogue chip.

1 Introduction

Today's information age demands quick confidential electronic communication. The Data Encryption Standard, DES, is widely used to provide this data security. DES's inherent level of security can be increased by compressing data, to increase its entropy, before encryption. Implementing DES and compression in hardware as opposed to software provides significantly faster data throughput rates. Furthermore, a hardware implementation, especially a single chip VLSI implementation, provides bigger barriers to potential code crackers. This paper describes the design, implementation, simulation and testing of the Rogue chip: a data compress-encrypt and decrypt-decompress VLSI chip.

The original design specification, as assigned in an advanced VLSI design course at the University of California, Santa Barbara, called for a data compress-encrypt chip with data throughput greater than 20 megabtyes per second. Furthermore, the fabrication budget dictated a die size of less than 12 mm^2 using HP's CMOS26G 0.8 μm technology. Finally, Cascade Design Automation's Epoch Integrated Circuit Design System was assigned as the prime CAD tool.

The adopted design methodology, centering around Epoch, first involved describing the Rogue chip in VHDL. Control was described with a subset of behavioral VHDL and synthesized with Epoch. The remainder of the chip was described in Epoch style structural VHDL. With this input, Epoch provided placement and route, gate sizing, timing analysis, synthesis and simulation models all utilizing HP's CMOS26G 0.8 μm standard cell library and ruleset. Epoch's manual intervention capabilities were used extensively to fine-tune floorplanning, placement and gate sizing. Thorough simulations, using Epoch's VDHL model of the chip, a custom VHDL test bench and Mentor Graphic's QVSIM, were performed. Finally, design rules checks, DRC, and layout versus schematic checks, LVS, were done with Mentor Graphic's CheckMate tools.

This paper is organized as follows. Section 2, the largest portion of this paper, details the Rogue chip design in a top-down manner. At each described level, relevant design issues and innovations are discussed. Section 3 briefly describes verification and simulation strategies. Section 4 presents testing methods and results. Finally, Section 5 concludes and summarizes chip statistics.

2 Top-Down Design Description

2.1 The End User's View

From the perspective of an end user, interfacing to and operation of the Rogue chip involves a straightforward asynchronous data transfer protocol and operation mode selection. Figure 1, the Rogue chip block diagram, overviews the signals for which an end user is responsible. In general, input signals are shown on the left side and output signals are on the right.

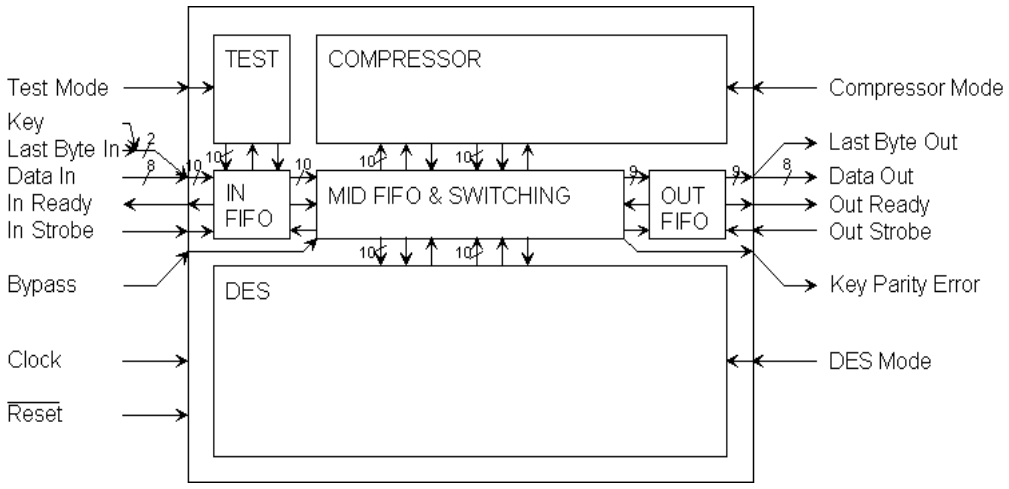

Figure 1: Rogue Chip Block Diagram.

2.1.1 Operation Modes

The Rogue chip supports six basic operation modes. The appropriate control signals for these modes are shown in Table 1. Six modes were defined for testing and operation flexibility. Every desirable configuration of compress and/or encrypt functions is possible. For non-test applications, the end user selects the desired mode, resets the chip, and proceeds with data input/output. Each of the six basic operation modes may also be run in test mode. In test mode, an internal module supplies a predefined input data stream. The end user compares the output data stream with the known correct output data stream to verify correct operation.

Table 1: Rogue Chip Mode Selection

Operation	Compressor Mode	DES Mode	Bypass
Compress	0	1	1
Decompress	1	0	1
Encrypt	0	0	1
Decrypt	1	1	1
Compress & Encrypt	0	0	0
Decrypt & Decompress	1	1	0

2.1.2 Data Input and Output

Data input uses a standard asynchronous protocol. After *reset* the Rogue chip is ready for data input. The interfacing device must present a 10-bit input packet to the Rogue chip. This packet consists of 8 bits of “Data In” plus 2 bits of global control. One of the 2 control bits, “Key”, is used to signal that the data payload contains key information for the DES. The other control bit, “Last Byte In”, signals that the data payload contains the last byte for processing. Limiting the data bandwidth to only 8 bits was one technique used to meet the less than 12 mm^2 area constraint. Once a valid input packet has been presented to the Rogue chip, “In Ready” is checked. If it is high, the Rogue chip is able to accept the input packet. At this point, the interfacing device pulses “In Strobe” for at least half of the Rogue chip clock period. Another input cycle may now begin. Data output uses a similar asynchronous protocol. In this case, the Rogue chip provides 8-bits of data and the “Last Byte Out” signal. The end user signals the Rogue chip by pulsing “Out Strobe” for more data after the current data has been safely read.

These simple asynchronous input/output protocols were chosen for three reasons. First and most importantly, the Rogue chip is completely stallable at either the input or output side. For example, if data is not being read from the output, all internal buffers will eventually fill and input data will no longer be allowed. Normal operation will resume without error once output data is read. This flexible interfacing feature significantly reduces the timing constraints imposed on the end user. Second, complete independence between the Rogue clock and the interface logic clock exists. If separate clocks are used, the Rogue chip arbitrates the handshaking signals in a punt or cycle-stall fashion to insure correct operation. Finally, this interface is easily adapted to comply with standard DMA protocols.

2.1.3 DES Key Input

DES requires a 64-bit user specified key. For any mode involving encryption or decryption, the first 8 bytes of input data after *reset* must be the key. These bytes must still be distinguished from normal data by setting “Key”. Multiple files can be processed with the same key. On the other hand, once data has been flushed from the chip, a new key may be entered without *reset*. Again, “Key” must be set for 8 contiguous input bytes of key data. Finally, DES requires that each byte of key data maintain even parity. If at anytime during key entry even parity is not maintained, “Key Parity Error” will go high for two Rogue chip clock cycles.

Operation will proceed as if the key is valid. The end user must monitor and latch this signal if key parity checking is desired.

2.1.4 File Demarcation

File demarcation is signaled using “Last Byte In/Out”. The end user must set the last input packet’s “Last Byte In”. If this is done, the Rogue chip will set the last output packet’s “Last Byte Out”. There are two reasons why this is needed. First, DES requires processing of 64-bit internal data packets. It does not process a packet until it is completely formed. If the last few bytes of data do not form a 64-bit packet, the “Last Byte In” signal forces the remaining slots in the 64-bit packet to be stuffed and flushes all internal data paths. Second, compression changes the number of bytes in the input and output data files. The “Last Byte In/Out” signals indicate when processing has completed.

2.2 The Internal High-Level View

Figure 1 also shows the high-level blocks within the Rogue chip. Internally, the chip is divided into three subsections: test, compressor and DES. Asynchronous FIFOs and small glue logic blocks provide the necessary data routing and buffering. There are four paths for data to take as it flows through the chip:

1. In FIFO→Compressor→Out FIFO (Compress/Decompress only modes.)
2. In FIFO→DES→Out FIFO (Encrypt/Decrypt only modes.)
3. In FIFO→Compressor→Mid FIFO→DES→Out FIFO (Compress-Encrypt mode.)
4. In FIFO→DES→Mid FIFO→Compressor→Out FIFO (Decrypt-Decompress mode.)

In test mode the test subsection supplies data to the In FIFO. Design considerations and trade-offs at this level are discussed in the next three sections.

2.2.1 Use of FIFOs

As seen in Figure 1, there are three FIFOs in the Rogue chip design. This heavy use of FIFOs simplified the design and increased flexibility. First, FIFOs provide a standardized hassle-free interface both internally and externally. The end user’s interface protocol is ready-to-go with no additional control specification. Second, the global control signals are piped through all the FIFOs. By doing this, there is no question of when data and associated control are active. Internally, both data and external control arrive at the same time from the FIFO’s output. Third, FIFO use isolated each internal subsection which bypassed need for extensive global control and timing. Control is easily distributed and localized into each module. Fourth, data is easily stallable both externally and internally. For example, stalls inherent to compressing and decompressing data are easily accommodated. A final advantage of FIFOs is less restrictive clocking. Since the only dependencies between the internal subsections are routed through FIFOs, there is no constraint for clock skews between these modules. Only clock branches feeding the same module have to be carefully clock skew balanced. Finally, because module clocks could be widely skewed, the peak inrush current could be reduced to diminish ground bounce and power-rail constraints.

2.2.2 Clocking

Figure 2 shows the Rogue chip clock distribution scheme. In addition to Epoch’s automated skew balancing which adjusts skew by varying clock route wire lengths, clock skews for each branch could be adjusted by manually sizing appropriate buffers. This allowed for greater flexibility, control and area savings. Given the clocking freedoms described in 2.2.1, only branches common to each internal subsection were balanced with respect to each other. They were balanced by first roughly equalizing the gate load on each branch. Fifteen leaves with loads of 15 to 32 gates provided the best experimental results. Then, each branch buffer was resized manually using Epoch’s TACTIC static timing analyzer. Two reroute and TACTIC gate size adjustments were performed before obtaining an acceptable skew. TACTIC estimated the worst skew as 250 ps between two branches in the final design’s DES clock distribution tree.

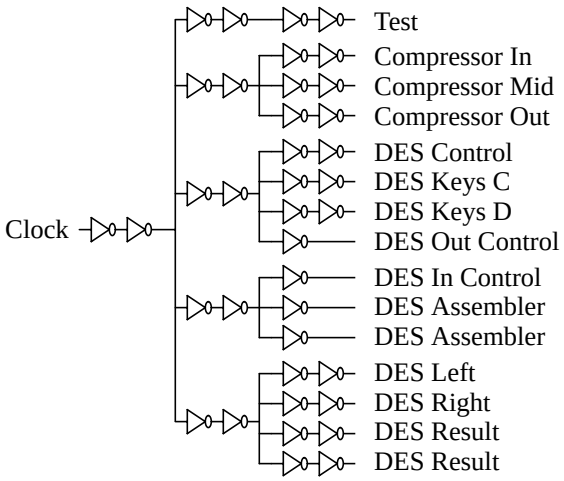

Figure 2: Rogue Clock Distribution.

2.2.3 Top-Level Floorplanning

As stated earlier, the fabrication budget placed severe limits on chip size. A 5.95 mm^2 core size was obtained only after more than thirty manual floorplanning sessions. Figure 3 shows the top-level floorplan. Application of two techniques resulted in this minimal core size. First, large standard cell groups were preferred over more numerous smaller standard cell groups and bit-sliced data paths. For example, the large standard cell group at the top of Figure 3 contains test, data routing (switch), compressor data path and control standard cells. Even the compressor data path, which does have easily identifiable bit-slices, is included as part of this large standard cell group. (To minimize timing problems, standard cells in the compressor data path were manually placed in a bit-slice configuration on the right side of the standard cell group.) This use of large standard cell groups was experimentally shown to reduce area requirements by roughly 35% for this design. Second, no general automated standard gate sizing was used. Each standard cell was originally set to the lowest possible gate size. Only paths identified as critical for the 80 MHz target clock by TACTIC were subjected to gate sizing. Although this blurred signal edges, transition times could still be bounded. This technique reduced the design size by approximately 15%.

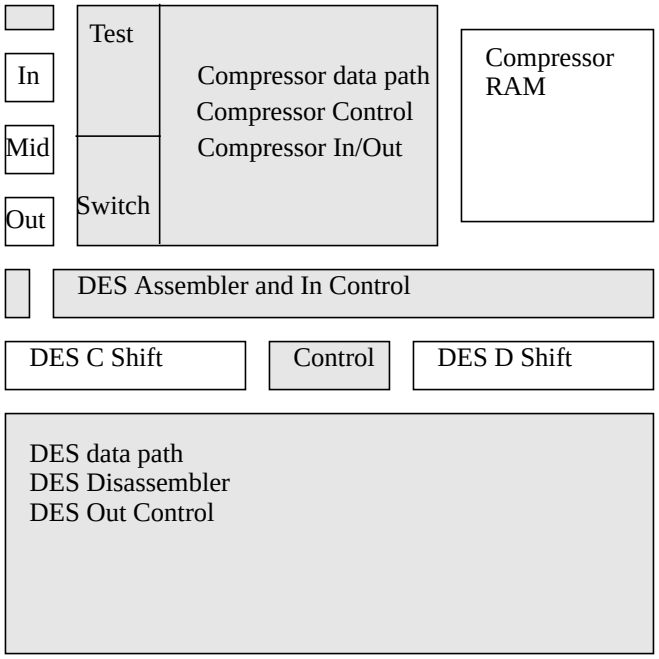

Figure 3: Rogue Top-Level Floorplan. Standard cell groups are shaded. Data paths are unshaded.

2.3 DES Subsection

Figure 4 shows a block diagram of the DES subsection. Standard DES, as defined by the IEEE and described in [2] is performed. At full capacity, this DES subsection can process 64-bits of data every 16 cycles. Throughput is maintained at one byte every two cycles with no bubbles since the assembler, disassembler and key scheduler operate in parallel with the DES data path. The DES subsection is fully stallable from the input or output. Finally, partitioning control into three machines, *In*, *DES* and *Out*, greatly simplified designing the DES subsection and enhanced data throughput performance. Considering the processes occurring in parallel, each machine was easier and clearer to define as a separate but interacting entity. Simple synchronous handshaking using only *ready* and *acknowledge* signals, was used to coordinate the three machines.

2.3.1 DES Data Flow and Control

DES is defined to operate over 64-bit words. Therefore, a 64-bit Word Assembler and Disassembler were created to interface with the external 8-bit interface. Since this DES subsection requires 16 cycles to process one word, the assembler and disassembler operate at half of the DES speed to load/unload the eight bytes required to define a word. When a packet has been assembled, DES control is signaled and the word passes to the DES left and right registers through a hardwired permutation.

Once data is present in the DES, the right register contents are wire expanded to 48-bits, XORed with 48-bits of key data and then presented to 8 substitution boxes. Each unique substitution box is multilevel combinational standard cell logic synthesized with SIS to produce a 4-bit output given a 6-bit input. The 4-bits from each of the 8 substitution boxes are concatenated to form a 32-bit result. After passing through another hardwired permutation, this result is XORed with the left register. To end the cycle, the original right register is placed into the left register and the processed 32 bits are placed into the right register. After 16 iterations, the final result is ready and passed to the 64-bit word disassembler.

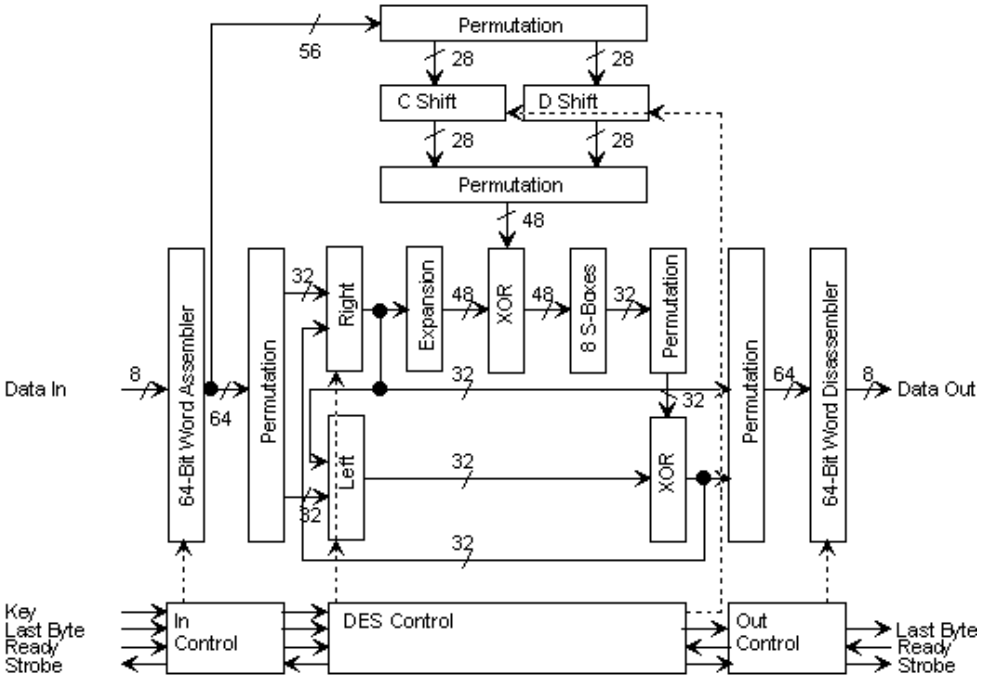

Figure 4: DES Data path.

The key scheduler accepts 56-bits of permuted data from the word assembler if the assembler receives a key set control bit for all eight input bytes. This key is stored only in the C and D shift registers. During each cycle of the 16 cycle DES computation, a new key is produced by shifting the C and D registers either one or twice. The key is shifted left or right depending on the mode of operation, encryption or decryption. The shift registers consist of standard D-registers with 3 levels of 2-input

multiplexers to direct the input. Control, shown with dashed lines, selects whether the shift registers will hold, load, shift right once or twice, or shift left once or twice. At the end of 16 cycles, the key will automatically be restored to its original value.

2.3.2 DES Floorplanning

The DES subsection floorplanning, seen at the bottom of Figure 3, consists primarily of large standard cell groups. The main reason for using large standard cell groups was the absence of clear bit slices in the structure. Given the many hardwired permutations, 32 to 48-bit expansion, and 32-bit crossing busses, there was no obvious data path configuration. Large standard cell groups provided the best area utilization.

2.4 Compressor Subsection

The next three sections describe the compressor subsection. Compression was included in this chip to decrease data regularity. Since most code cracking techniques exploit data regularity, compressed-encrypted code is more secure. A novel compression scheme, closely matching the DES throughput and meeting area constraints, is implemented. This scheme was chosen after experiments with Lempel-Ziv type algorithms indicated considerable area and time complexity.

2.4.1 Compression Algorithm

The compression algorithm, shown in Figure 5, is an extension of run-length encoding. The compressor adds one status bit to each data byte. This status bit is set if the current byte, used as a pointer into the 256-byte RAM dictionary, points to a value that matches the following byte. Since a match occurred and is indicated with a set status bit, the following byte is not sent. Thus, 8-bits of data have been compressed to one status bit. Furthermore, strings of consecutive matches are also compressed. A count is recorded for the number of consecutive matches. If this count is greater than one, then it is also sent with its status bit set. Repeating strings such as *abcdefabcdefabcdefabcdef* are easily compressed to six bytes and status bits, *abcdef* (needed to create the dictionary), plus two bytes and status bits (initial string byte and count). This ability to compress strings coupled with the 256-byte dynamic dictionary provide clear performance advantages while preserving traditional run-length encoding functionality.

Valid decompression requires meeting two criteria. First, the decompressor must reconstruct the dynamic dictionary using only initial string bytes, counts, and status bits. The compressor constructs its dynamic dictionary in a way that insures this coherency is maintained. Second, to maintain good compression for the common case of only a single match (i.e. a compressible string of

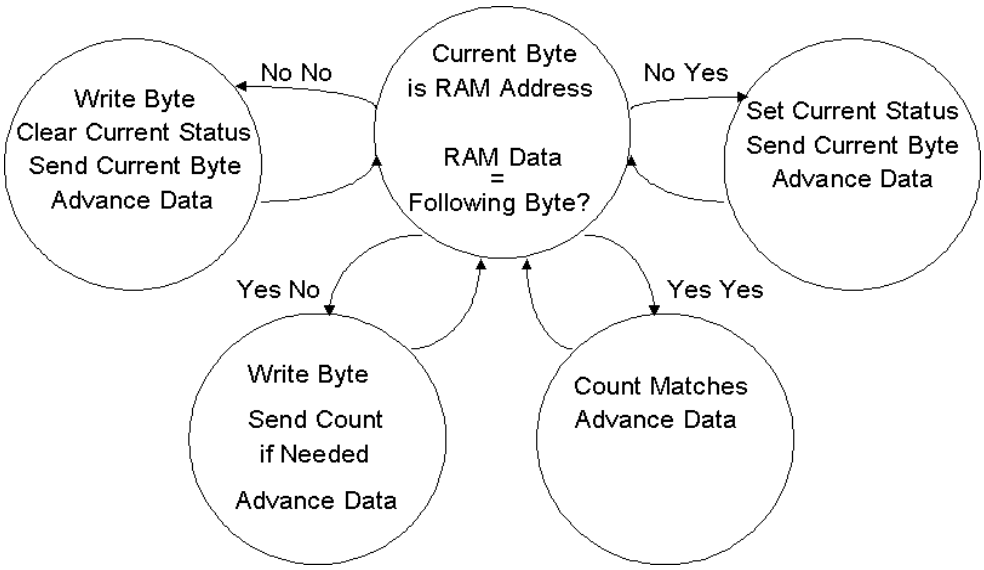

Figure 5: Compression Algorithm. No No means no previous match and no current match.

two bytes), the compressor only sends the initial byte with status bit set but no count byte. The decompressor can't distinguish between back-to-back 2-byte strings and an initial string byte and count since both will have two set status bits. To remedy this situation, the compressor never sends compressed 2-byte strings back-to-back. This insures that whenever two bytes with set status bits are received back-to-back, they are always an initial string byte and a count.

Table 2 shows compression statistics for this algorithm. This algorithm was chosen after experiments with slight variations (i.e. 2-byte strings only, greater than 2-byte strings only) proved it best. Typical file size reductions are between 15 to 25%. The worst case expansion, on a previously compressed file, was 12.2%.

Table 2: Compression Statistics

File	Type	Compressed Size/Original Size
TCW.EXE	Executable	0.743
VCR.BMP	Graphic	0.816
SALETRAC.XLW	Spreadsheet	0.843
DESREG.VHD	Text	0.803
TANGO2.ZIP	Compressed	1.122
ABC.TXT	abc 400 times	0.013

2.4.2 Compression Implementation

Figure 6 shows the compressor data path. Similar to DES, three distributed control units were implemented in the compressor. This allowed for clearer control specification and simpler parallel processing. Similar to DES, the three machines were coordinated through simple synchronous *ready* and *acknowledge* signals. By defining three machines and Ping-Pong input and output registers, a throughput of 1 byte every 2 cycles is sustainable which matches the DES data throughput. In the worst case, the end user may see a data flow rate of up to 12.5% slower than this due to the compresssor's stuffing of status bits into the data stream.

To ease interface to the DES subsection, status bits are sent separately from their associated data bytes. After 7 bytes, the compressor dumps 7 bits of status information. This status byte plus the 7 data bytes eventually form the 64-bit DES word. For decompression, the DES disassembler automatically reorders the data stream so the decompressor receives the status byte first. Finally, Figure 6 shows 10 bits entering and 10 bits leaving the compressor subsection. The global control bits are piped along with the data through parts of the data path. Although not shown in Figure 6, some of the multiplexers and registers are actually 9 or 10 bits wide to accommodate control bits which must be piped through that section of the data path.

There are two techniques used in the compressor to make Rogue encrypted data less crackable. First, the one unused bit in the status byte is stuffed with a salt bit. The test module's free-running LFSR counter supplies this pseudorandom data flow dependent salt bit. This trick effectively adds noise to the data. Second, a different LFSR is used for counting matches. This provides a pseudorandom appearing count progression to further obscure the true meaning of the data.

2.4.3 Compressor Critical Path

The only critical path for the Rogue chip was through RAM, Comparator A and Compressor Control. After floorplanning, TACTIC reported this path delay at roughly 1 ns over the target clock period of 12.5 ns. To meet the design specification, three solutions were incorporated. First, another comparator was added to the design. Originally, RAM data was routed through Mux

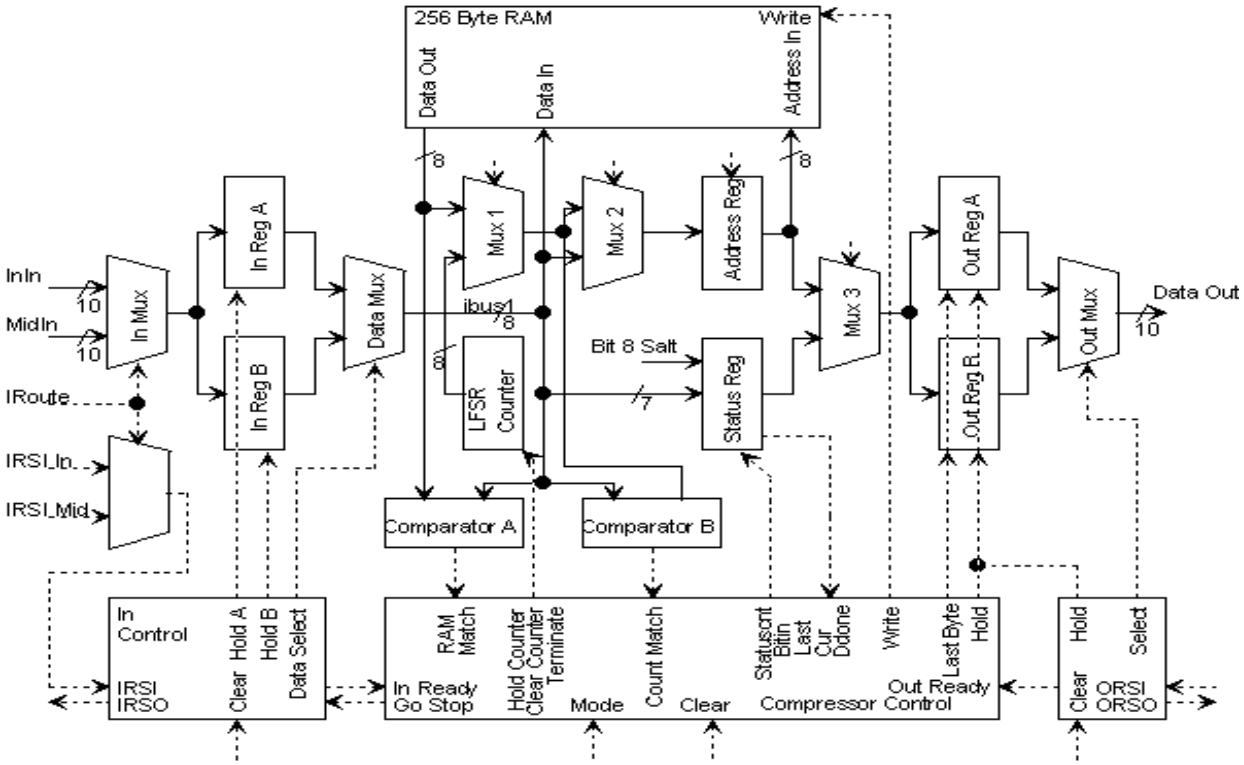

Figure 6: Compressor Data Path.

1 to Comparator B and finally to Compressor Control. By adding Comparator A, one multiplexer delay was removed from the critical path. Second, to reduce wiring delays, control standard cells included in this critical path were manually selected and placed nearest relevant data path circuitry. Other path lengths and area increased slightly. Finally, all standard cell gate sizes were reduced to the minimum size. Only gates essential to propagating the critical path signals were resized. This reduced capacitive delay along the critical path. Although other paths were lengthened, no other paths became critical. The critical path was eliminated. This critical path plus a handful of other paths were still very close to the target clock period of 12.5 ns and constrained the design's top clock rate.

2.5 Test Subsection

The Rogue chip includes a simple test pattern generator. It is not designed for full fault coverage. Instead it is used for determining correct chip functionality with minimum support logic. When test mode is active, the test module supplies input to the In Fifo. If no output stalls occur, input is sustained at 1 byte every 2 cycles. The input sequence is generated by a 16-bit LFSR counter. This pseudorandom input sequence is deterministic even with output stalls. Individual subsections may be tested by selecting appropriate operation modes. When the DES subsection is included, the first 8 bytes of input data are used for the key. To check for correct chip functionality, the output data signature is compared to a known correct signature.

3 Simulation and Verification Strategies

Simulation centered around Epoch generated complete pin-level timing-included VHDL chip models. Two of these models were included as components in a VDHL test bench. This test bench fed large real and random data files to the first chip model. The output of this model was fed into the second chip model which was set to simulate the inverse function of the first model. The final output data was compared with the original data. To make the simulations more realistic, various interfacing conditions were modeled. Simulations with random stalls, random stall lengths, full-speed data input/output and out of sync clocks were performed. These simulations were extremely helpful and were a prime tool used in debugging the chip. Before

fabrication, over 10 megabytes of data produced flawless results in over 30 hours of simulation time. Complete SPICE electrical simulation was not attempted because of the size, 53,458 transistors, of the design.

Before being submitted to MOSIS for fabrication, Epoch independent design rules and layout versus schematic checking was done. Mentor Graphic's CheckMate mask check and net check detected several critical errors. Epoch's manual intervention capabilities allowed for correcting these errors.

4 Testing Methods and Results

MOSIS fabricated and packaged 25 Rogue chips. A Motorola microcontroller based test setup was custom built for the Rogue chip. In the first series of tests, the microcontroller assembly routines checked for correct output signatures in all valid test modes. The Rogue's built in self test module greatly simplified this process and allowed the first test results to be obtained after only a day of work. To further test the chip, the microcontroller provided input data, collected the results, inverted the Rogue function, input the collected results and compared to see if the final output matched the original input. Random and real data files of varying lengths were used for this test. The maximum clock rate for the microcontroller was 8 MHz. The Rogue chip was clocked independently to test maximum throughput. Although stalls were inevitable with the microcontroller's low clock frequency, once information was inside the chip it would be processed at the Rogue's clock rate.

Out of ten chips tested, nine have fully functional DES subsections. DES functionality at a clock frequency of 66 MHz and data throughput of 33 megabytes per second has been confirmed. Operation at frequencies higher than 66 MHz and ultimately at the target frequency of 80 MHz must still be tested. The compressor has a design error which simulation failed to reveal. A race condition exists between the "Out Mux Select" and "ORSO" FIFO strobe of Figure 5. Data Out is being clocked into a FIFO before Data Out has settled. This error was reproduced in simulations once more accurate timing data was supplied. From tests and simulations, this appears to be the only error in the design. Data Out is correct enough to identify the error but incorrect enough to prohibit usefulness of the compressor. Detecting this error during the design phase would have been more likely if FIFO models with data hold and setup violation warnings and best/worst case timing data had been used for simulations.

5 Summary and Conclusions

This paper described the design, implementation, simulation and testing of a data compress/encrypt and decrypt/decompress chip. The design methodology, employing Cascade Design Automation's Epoch and Mentor Graphic's QVSIM and Checkmate was predominately successful. The chip was designed to perform DES at 40 megabytes per second. The compressor, incorporated to make the encrypted data less crackable by reducing the data's organization, implemented a novel algorithm to match the throughput of DES. A tight area constraint of die size less than 12 mm^2 was met. The fabricated chip performed DES at 33 megabytes per second. A plot of the final chip is shown in Figure 7. Table 3 summarizes chip statistics.

Table 3: Rogue Chip Statistics

Die Size	3349 μm × 3238 μm	**Density**	4958 transistors/mm^2	**Target Clock Frequency**	80 MHz
Core Size	2391 μm × 2488 μm	**Operating Voltage**	5 V	**Tested Clock Frequency**	66 MHz
Transistors	53,548	**Calculated Total Power**	553 mW	**Tested DES Throughput**	33 megabytes/second
Technology	HP's CMOS26G 0.8 μm	**Package**	cda 64p400 (65 pin PGA)		

References:

[1] H.B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley Publishing Co., Reading, Mass., 1990.
[2] Simon Haykin, *Communication Systems*, John Wiley and Sons, Singapore, 1995, Appendix A 10.4.