Efficient DES Key Search

Michael J. Wiener

Bell-Northern Research, P.O. Box 3511 Station C, Ottawa, Ontario, K1Y 4H7, Canada

1993 August 20

Abstract. Despite recent improvements in analytic techniques for attacking the Data Encryption Standard (DES), exhaustive key search remains the most practical and efficient attack. Key search is becoming alarmingly practical. We show how to build an exhaustive DES key search machine for $1 million that can find a key in 3.5 hours on average. The design for such a machine is described in detail for the purpose of assessing the resistance of DES to an exhaustive attack. This design is based on mature technology to avoid making guesses about future capabilities. With this approach, DES keys can be found one to two orders of magnitude faster than other recently proposed designs. The basic machine design can be adapted to attack the standard DES modes of operation for a small penalty in running time. The issues of development cost and machine reliability are examined as well. In light of this work, it would be prudent in many applications to use DES in a triple-encryption mode.

1. Introduction

From the time that details of the Data Encryption Standard (DES) [4] were made available, there has been a great deal of controversy surrounding its short 56-bit keys. One of the first criticisms came from Diffie and Hellman who argued that it was feasible to build a machine that could attack DES by exhaustively searching through all 2^{56} DES keys, and that the cost of building this machine would decrease rapidly with time [6]. Since then, several attempts have been made to assess the cost and time required for exhaustive key search [7, 8, 10, 15].

In recent years, analytic techniques have been found to attack DES. Biham and Shamir's differential cryptanalysis can be used to find a DES key given 2^{47} chosen plaintexts [1, 2]. Matsui's "linear cryptanalysis" can find DES keys given 2^{47} known plaintexts [12]. However, unless these attacks are improved greatly to lower the number of plaintexts required, they will be of only theoretical interest. For now, the most practical and economical way to attack DES is by exhaustive key search.

The purpose of this paper is to evaluate the resistance of DES to a known-plaintext key search attack. This is done by designing a machine which takes a plaintext-ciphertext pair (*P*, *C*) and finds the DES key or keys which encrypt *P* to produce *C*. There have been numerous unpublished and unverifiable claims about how fast the DES key space can be searched. To avoid adding to this list of questionable claims, a great deal of detail in the design of a key search machine is included in the appendices. This detailed work was done to obtain an accurate assessment of the cost of the machine and the time required to find a DES key. There are no plans to actually build such a machine.

2. Key Search Chip

The most important component in the key search machine is the key search chip. This chip takes a plaintext-ciphertext pair (P, C), a starting key K, and searches through keys until one is found that encrypts P to produce C. The detailed design of this chip is given in Appendix A.

In order to maximize the rate at which keys are tested, the implementation of DES within the chip is fully pipelined. This means that there is a separate implementation of each round of DES and the rounds are separated by registers. This is similar in concept to an assembly line where at any given time, items are at varying stages of processing. On each clock tick, the plaintext P and a new key enter the pipeline, and all previous encryptions advance through another round of DES. In this way there are 16 encryptions taking place simultaneously. Although it takes 16 clock ticks to complete an encryption, one encryption completes at the end of the pipeline on each clock tick. As each encryption completes, the result is compared to the ciphertext C that was originally loaded into the chip. If there is a match, then the chip stops and saves the key that lead to the match.

An important feature of this chip is that there is very little need for input and output. This makes it possible to clock the chip at high speed without the requirement of getting data in and out of it at high speed. The chip described in Appendix A can be comfortably clocked at 50 MHz, and would cost $10.50 per chip to manufacture. This precise assessment of speed and cost is made possible by the fact that the chip has been designed down to the gate level in a mature CMOS process. Some cost estimates for key search chips in the literature suffer from incompleteness of design and guesses about future technology.

In summary, this chip can test keys at the impressive rate of 50 million per second. This is made possible by the use of pipelining and the elimination of high-speed input and output.

3. Key Search Machine

The basic approach taken to building a key search machine out of key search chips is to build a hierarchy of controllers. At the lowest level, a group of ten key search chips is controlled by a microcontroller which assigns tasks and polls the chips for any found keys. The key search board design described in Appendix B shows that for the selected board size, twelve groups of ten key search chips fit on a board. The twelve microcontrollers are themselves controlled by a master microcontroller. The master microcontroller assigns a plaintext-ciphertext pair and a range of keys to each slave microcontroller. The slave microcontrollers subdivide the range of keys into ten subranges for the ten key search chips.

A description of the shelf and frame design is given in Appendix C. Each shelf contains twelve key search boards and a controller board. Four shelves are stacked together to form a frame which is controlled by a PC. Overall, a frame costs $100 000 and contains 5760 key search chips.

A key search machine would consist of a number of frames which would find one or more candidate keys. Each candidate key would have to be tested to see if it is correct (perhaps by using another plaintext-ciphertext pair).

4. Performance

Using the design described here, DES keys can be found alarmingly quickly. On average, one would expect to search through 2^{55} keys before the correct key is found. Using the fact that a key search chip tests 50 million keys per second and a frame costs $100 000 and contains 5760 key search chips, we get the following table of machine cost versus key search time.

Key Search Machine Cost	Expected Search Time
$100 000	35 hours
$1 000 000	3.5 hours
$10 000 000	21 minutes

5. Development Cost

A key search machine could be built in about ten months by three people (not including support groups for manufacture and chip and board layout). One person would be needed to capture the chip design, simulate it, and oversee its layout and fabrication. A second person would design, simulate, and oversee the fabrication of the key search board and the shelf controller board. This person would also handle the shelf and frame design. A third person would write software for the PC, shelf controller board, and the microcontrollers. Taking into account loaded labour rates, the cost of employing these people is about $100 000 each. Allowing an additional $200 000 for support in the chip layout, chip fabrication, and board layout brings the total development cost to about $500 000.

6. Comparison to Other Recent Key Search Designs

The approach to DES key search proposed here is considerably faster than other recently proposed designs. Comparisons to other designs are given below.

Garon and Outerbridge [8] used an approach to DES key search that is similar to the one used here; design a specialized key search chip and pack as many of them as possible into a machine. Based on Garon and Outerbridge estimates, a $1 million machine could search half of the DES key space in 9 days in 1995 and 43 hours in the year 2000. However, using the design in this paper, this task would take only 3.5 hours using today's technology. This is 60 times faster than the 1995 estimate and 12 times faster than the year 2000 estimate.

At Crypto '92, Eberle [7] described a 1 Gbit/s gallium-arsenide (GaAs) DES chip. This chip costs $300. Using $1 million worth of these chips, it would take 8 days to search through half of the key space. This is 55 times slower than the design in this paper. This difference is mainly due to the difference in chip cost and the use of pipelining. Using Eberle's chip has the advantage of not having to do a chip design, but the gain from designing a new chip is considerable.

During the rump session of Crypto '92, Wayner [15] proposed the idea of searching the DES key space using a content-addressable search engine. This approach has the advantage that it works on a general-purpose machine. If the content-addressable machine has already been acquired for general use by some organization, then one only needs to write a small amount of software and acquire time on the machine. Based on Wayner's estimate, one could search through half of the key space in 30 days on a $1 million content-addressable machine. The drawback of this approach is that it is 200 times slower than the design in this paper.

An often proposed method of attacking DES is to take advantage of the spare cycles on a large installed base of PCs and workstations. However, DES is poorly suited to software implementation. On a Sun SparcStation-2, a key can be tested in 50 µs. This means that it would take 2500 workstations to keep up with a single key search chip of the type described in Appendix A. One key search frame has the equivalent key searching power of 14 million workstations.

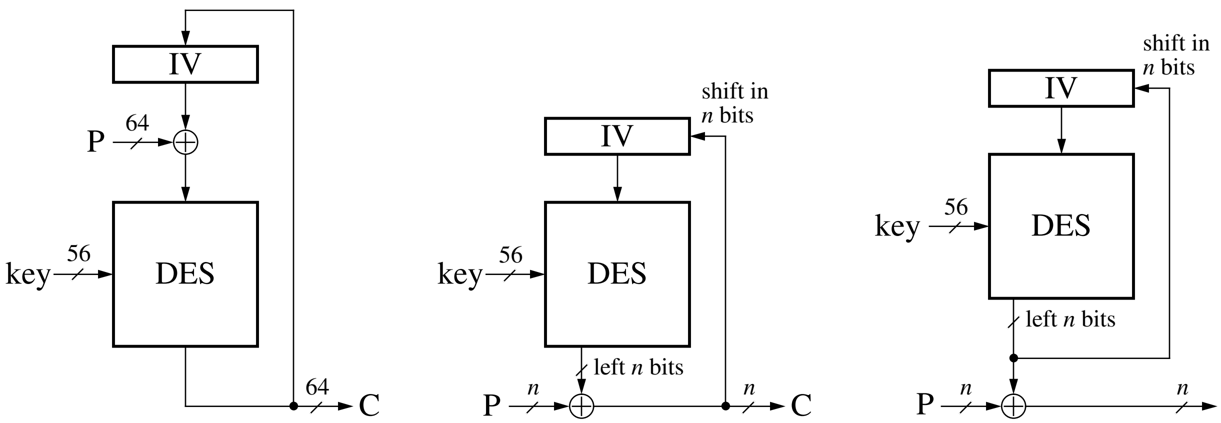
7. Reliability and Testing

When DES key search machines were first proposed, reliability was a difficult issue to deal with. But today, large systems are routinely built that are much more complex and have higher reliability requirements than a key search machine.

The analysis in Appendix D shows that the mean time between failures of a machine consisting of n key search frames is about $9500/n$ hours. From Section 4, the expected time of a single key search for n key search frames is $35/n$ hours. Dividing these two expressions, we find that a failure is expected every 270 keys found. Because the failure rate is low, a reasonable approach to testing is to perform a self-test between key searches and replace any faulty cards. There is no need to do testing during key searches.

8. Modes of DES

Thus far we have concentrated on attacking DES when it is used in the electronic codebook (ECB) mode. However, there are other standard modes of DES [5]. All of the 64-bit modes can be attacked using the same machine design (this is demonstrated below). To support modes which produce less than 64 bits of ciphertext at a time requires modifications to the chip design. The modes of DES are shown in the following figure. In all cases a 56-bit key and a 64-bit initialization vector (IV) are required. On each step, the key, IV, and plaintext block *P* are used to produce a ciphertext block *C*.



cipher-block chaining (CBC) *n*-bit cipher feedback (CFB) *n*-bit output feedback (OFB)

Figure 1: Encryption using the various modes of DES

For the 64-bit modes, it is possible to construct a pair (P', C') which can be used as plaintext and ciphertext inputs to the existing key search design to find the key. For CBC mode, if a plaintext block P_i is known, then use $P' = P_i \oplus C_{i-1}$ and $C' = C_i$. For 64-bit CFB mode, if a plaintext block P_i is known, then use $P' = C_{i-1}$ and $C' = P_i \oplus C_i$. For 64-bit OFB mode, if two consecutive plaintext blocks P_i and P_{i+1} are known, then use $P' = P_i \oplus C_i$ and $C' = P_{i+1} \oplus C_{i+1}$.

Among the modes of DES that produce less than 64 bits of ciphertext at a time, two popular ones are 1-bit and 8-bit CFB. The modification to the key search design required to support these modes as well as ECB mode are considered in Appendix E. We assume that there are 64 bits of known plaintext available and that multiple encryptions with the same key are required to obtain a full match between plaintext and ciphertext. The conclusion is that the new key search chip would cost \$12. This increases the cost of a key search frame to the point where only nine frames could be built for \$1 million. This slows down the attack on ECB mode somewhat. The CFB modes are slowed down further by the need to perform additional encryptions when there are partial matches with the ciphertext. For 1-bit CFB, there is a 50% chance that the output bit from an encryption matches the stored ciphertext bit and that another encryption will have to be performed with the same key. As soon as a ciphertext bit does not match, the key is rejected. This makes the expected time to find a key for 1-bit CFB mode twice as long as for ECB mode. Using similar reasoning, 8-bit CFB mode takes approximately 256/255 times as long as ECB mode. This information is summarized in the following table.

Mode of DES	Expected Key Search Time for a $1 million Machine
ECB	4 hours
CBC	4 hours
64-bit OFB	4 hours
64-bit CFB	4 hours
8-bit CFB	4 hours
1-bit CFB	8 hours

9. Triple-DES

One method of improving the security of DES greatly is to use triple-DES. With triple-DES encryption, each block of plaintext P is processed three times using either two or three independent keys to produce the ciphertext C as shown in Figure 2.

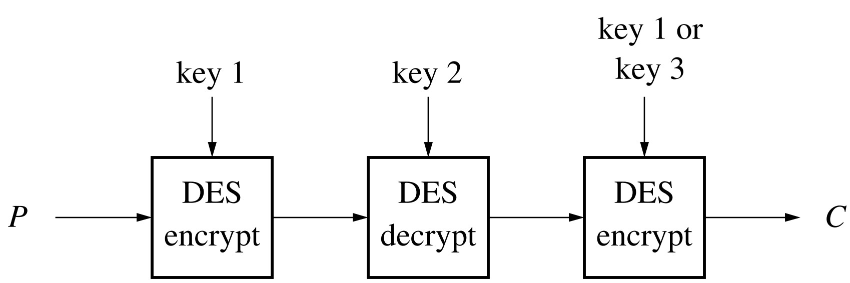


Figure 2: Triple-DES Encryption

The reason for making the middle operation a decryption is for compatibility with single-DES. If all keys are made equal, then triple-DES collapses to single-DES.

The two-key version of triple-DES was estimated to be 10^{13} times stronger than single-DES by van Oorschot and Wiener [14]. The three-key version is even stronger. In cases where security must be improved for a large installed base of equipment using DES, it is likely to be much less costly to switch to triple-DES than it is to switch to a different cryptosystem.

10. Conclusion

It is possible to build a $1 million machine that can attack DES and recover a key in an average time of 3.5 hours. The machine uses a known plaintext to exhaustively search through the DES key space and could be developed for $500 000 in about 10 months. Because a great deal of detail has gone into the design of the key search machine, we can have high confidence in the assessment of its cost and speed. The key search design presented here is one to two orders of magnitude faster than other recently proposed designs.

Even cryptosystems with 64-bit keys may be vulnerable. If DES were modified to use 64-bit keys, there would be 2^8 times as many keys to search through, and a $10 million machine would take an average of 3.7 days to find a key.

It is possible to build a key search machine that can support a range of modes of DES with little penalty in run-time. A $1 million machine would take 8 hours on average to find a key used in 1-bit CFB mode and 4 hours on average for any of ECB, CBC, 64-bit OFB, 64-bit CFB, or 8-bit CFB mode.

This work shows that exhaustive DES key search is alarmingly economical. If it ever was true that attacking DES was only within the reach of large governments, it is clearly no longer true. A fairly painless way to improve security dramatically is to switch to triple-DES.

Appendix A: Key Search Chip Detailed Design

This appendix contains the detailed design of a DES key search chip. The chip takes a plaintext *P*, a ciphertext *C*, and a starting key and searches through keys until one is found that encrypts *P* to produce *C*. An important feature of this chip is that the implementation of DES is pipelined. This means that there is separate silicon for each of the 16 rounds, and the rounds are separated by registers. At any given time, there are 16 keys in the pipeline. On each clock tick a new key enters the pipeline, other keys advance by one round, and the oldest key's encryption completes. As each encryption completes at the end of the pipeline, the result is compared to the ciphertext value *C*. Using this pipelined approach, keys can be tested at the rate of one key every clock tick.

The chip is designed using the standard cells in a 0.8 μm CMOS process. The chip is fully synchronous; all flip-flops use the same clock. The design is presented in a top-down fashion. The first subsection in this appendix contains the top-level design consisting of a number of blocks. In the following subsections, each block is subdivided into smaller blocks. At the end of this hierarchical decomposition, the standard cells (basic components such as and-gates, or-gates, and flip-flops) used in the design are described.

To help in distinguishing high-level blocks from standard cells, high-level blocks have arrowheads on their inputs and bold letters in their names. For standard cells, names are not bolded and there are no arrows on input lines.

Throughout the design, standard bit numbering is used with 0 as the least significant (rightmost) bit. This differs from the DES description [4] where the numbering of a 64-bit quantity begins with 1 for the most significant bit up to 64 for the least significant bit.

Sufficient buffering was included in the chip design to limit the fan-out of all gates to at most 8. For lines interconnecting the topmost blocks in the top level design, the fan-out was limited to 4.

Most modern large chip designs include built-in self test (BIST) circuitry. Because the key search chip is small and can be fully tested from its external pins, no BIST circuitry has been included in its design. If BIST were added, the cost of the key search chip would increase by about 30%.

The worst case delays from the output of one flip-flop to the input of another flip-flop occur in the rounds of encryption. Taking into account this delay, the delay through the flip-flop, and clock skew across the chip, this chip can be clocked comfortably at 50 MHz. This means that once the pipeline is full, a key is tested every 20 ns.

The most important factor in determining the cost of this chip is its area. In the CMOS process used here, units of area are called "sites". A site is 3.6 µm × 54.4 µm. Each of the standard cells requires an integral number of sites (see section A.44). For this chip, the total area of all gates not including clock regeneration or pads for external connections is 73637 sites. Clock regeneration requires 210 sites for each 100 flip-flops. Because there are 2164 flip-flops, 4620 sites are needed for clock regeneration. This makes a total of 78257 sites not including pads. A standard way to quote the size of a chip is to specify the number of equivalent gates using a 2-input nand gate as the standard. For this CMOS process, a 2-input nand gate requires 3 sites. Therefore, this chip design has 26086 equivalent gates. Taking into account the area required for interconnections and pads, this chip's area would be 260 mils × 260 mils.[1] Although, this chip only needs 27 pins, it is too large to fit in a 28-pin PLCC (plastic leaded chip carrier) package. The next standard size is a 44-pin PLCC package. Using a 44-pin PLCC package, the cost would be $10.50 per chip to fabricate and test in volumes of at least 10 000.

The power consumption of this chip would be about 450 mW. This is somewhat less than one might expect for a 50 MHz device. However, the chip is small and the I/O to it is quite slow. The I/O consists of loading values into the chip initially and polling it periodically to see if it has found the desired key. This means that very little power is dissipated in the pads.

[1] 1 mil = 0.001 inch = 25.4 µm.

The following table gives the area and page number for each block in the hierarchical design.

Block name	Area (sites)	Page number	Block name	Area (sites)	Page number
top level	73637	9	S_output	277	29
stop	21	12	minterm_4	15	30
compare	383	12	register_32	320	31
compare_16	94	12	register_8	80	31
compare_4	22	13	xor_48	192	32
plaintext and ciphertext	1148	13	xor_32	128	33
write_control	36	14	xor_8	32	33
mux2_register8	136	15	E_perm	0	34
key_counter	1437	16	P_perm	0	34
write_control7	33	17	key_round_1	560	34
poly_register	207	18	key_round_2	560	35
count_register	195	19	register_28	280	35
output_key	1395	21	register_7	70	36
mux2_register7	122	22	PC2_perm	0	36
mux7_1	51	23	rotate_left	0	36
buffer8	24	23	rotate_left_2	0	37
buffer3	9	24	initial_perm	0	37
pipelined_DES	67968	24	inverse_initial	0	37
encrypt_round	3688	26	PC1_perm	0	37
S_box	2728	27	inverse_PC1	0	38
dual_dec38	64	27	external_interface †	137	38
S1_wiring to S8_wiring	0	28			

† The area listed for the external interface does not include the pads.

A.1 Top-Level Design

The key search chip has a microprocessor interface which allows a plaintext, a ciphertext, and a starting key value to be written in, and the found key to be read out. In addition, the microprocessor can read and write the status ("go" or "stop") of the chip. The chip contains four registers which are visible to the microprocessor (see Figure A.1). The write-only registers include the 64-bit plaintext register, the 64-bit ciphertext register, and the 56-bit key counter register. There is also the read-only 56-bit output_key register. Note that in this design, key parity bits as defined in DES [4] are not included. The microprocessor can also read and write the stop bit which indicates the state of the chip.

A total of 27 pins are used. Five pins are allocated to each of power and ground. One pin is for the 50 MHz clock input, and there is a tri-state pin for testing purposes. The remaining 15 pins are the microprocessor interface. There is a chip enable line, a read/write line, 5 address lines, and 8 data lines for byte-wide access. The following table shows the address map of the chip.

Address	Read	Write
00-07	not used (zeros are read)	plaintext (bytes 0-7)
08-0F	not used (zeros are read)	ciphertext (bytes 0-7)
10-16	output_key (bytes 0-6)	key_counter (bytes 0-6)
17	not used (zeros are read)	not used
18-1F	stop bit (in bit 0 with 7 leading zeros)	stop bit (from bit 0 - other bits ignored)

While the chip is in the go state, the key_counter steps through the keys and puts one key per clock tick into the DES pipeline. This key is used with the contents of the plaintext register to perform the first round of encryption. All other stages of the pipeline perform one round of a previous encryption. From the last stage of the pipeline comes the result of an encryption and the key used in that encryption. The key is loaded into the output_key register (regardless of whether it is the desired key or not). The result of encryption is compared to the contents of the ciphertext register. If the ciphertext matches, then the chip goes into the stop state.

In the stop state, the key_counter does not count, and it is possible for the microprocessor to write a new value into the key_counter. The DES pipeline continues to run in the stop state, but the output_key register ceases to take in the keys coming out of the pipeline. In this way, the output_key register will be left holding the key which caused a match with the ciphertext register. In the case where the microprocessor stops the chip, the output_key register will simply hold the last key which came out of the pipeline before the chip was stopped.

The intended way for a microprocessor to use this chip is for it to first stop the chip, fill up the plaintext, ciphertext, and key_counter registers, and then put the chip in the go state. The microprocessor then polls the chip periodically to see if it has found a key and has stopped. This continues until a key is found or the microprocessor stops the chip to start a new task.

To save circuitry, the key_counter does not count linearly through the key space. The carry look-ahead logic for a 56-bit counter would be large, and it would be difficult to get it to run at the required speed. A linear-feedback shift register based on a primitive generator polynomial is used to step through all the non-zero keys. The all-zero key must be tested separately by the processor which controls the collection of key search chips. The generator polynomial used is $g = x^{56} + x^7 + x^4 + x^2 + 1$. If the first key tried is k, then the subsequent keys tried are $kx \bmod g$, $kx^2 \bmod g$, $kx^3 \bmod g$, and so on. The processor which controls the collection of key search chips will have to take this counting method into account when calculating the starting key values to load into each chip's key_counter.

In Figure A.1, clock, power, and ground are not shown leaving the external interface, but they go to all other blocks.

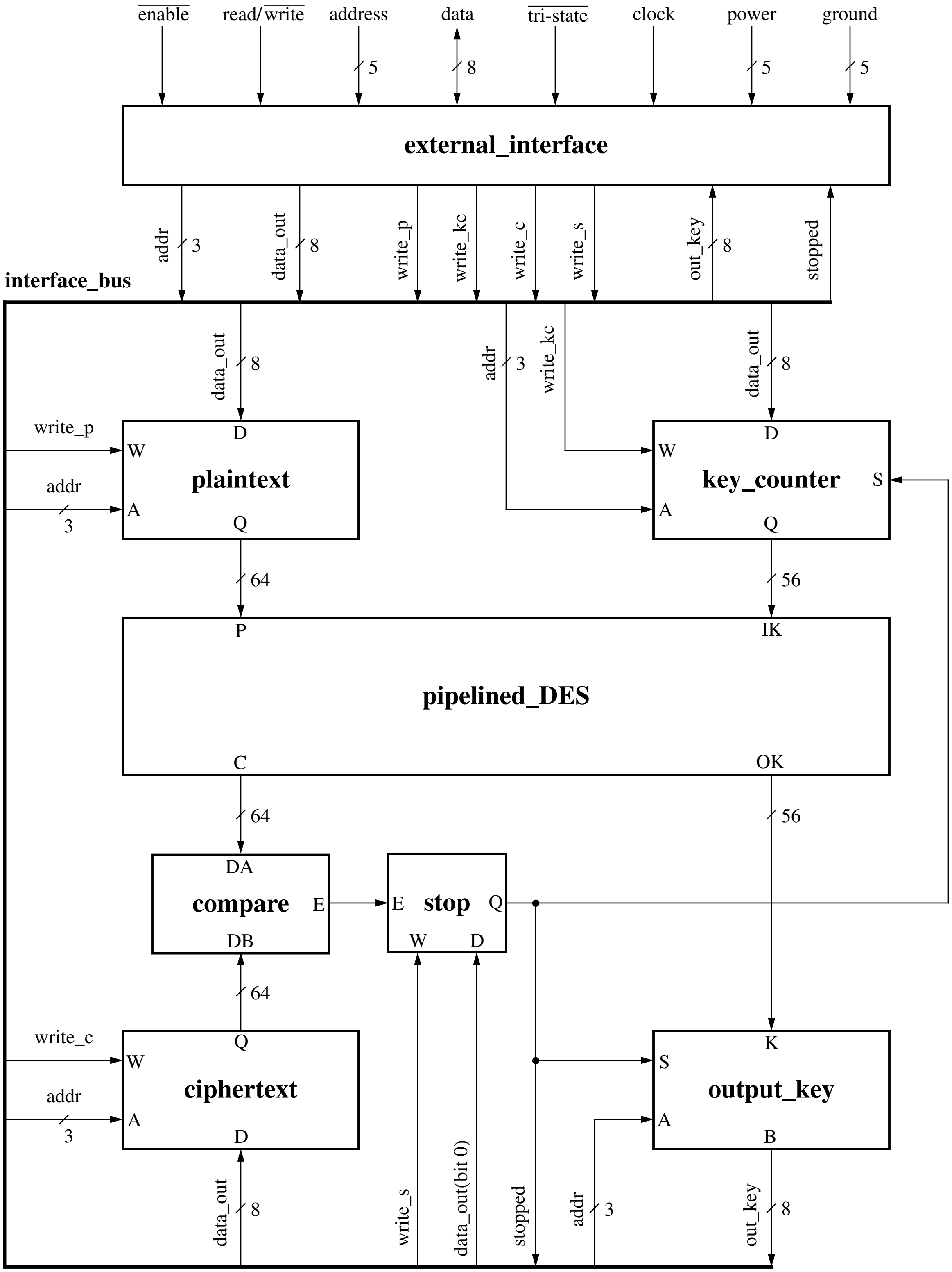


Figure A.1: Top-Level Key Search Chip Design

A.2 Stop Block

The stop block contains the flip-flop which indicates whether the chip is in the stop state (Q = 1) or the go state (Q = 0). The inputs to this block are a write line W, a data line D, and an equal line E which is a 1 if there has been a match with the ciphertext register. If W = 1, the stop bit takes on the value of D. If W = 0 and E = 1, the stop bit becomes a 1. If W = 0 and E = 0, the stop bit retains its old value.

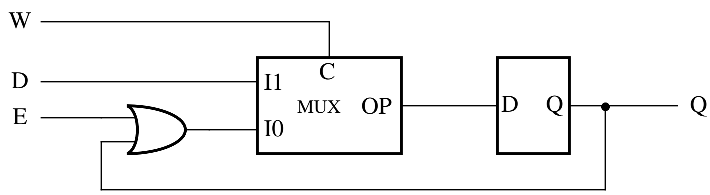


Figure A.2: Stop Block

A.3 Compare Block

The compare block outputs a 1 if all 64 bits of DA are equal to the corresponding bits of DB. The output is 0 otherwise.

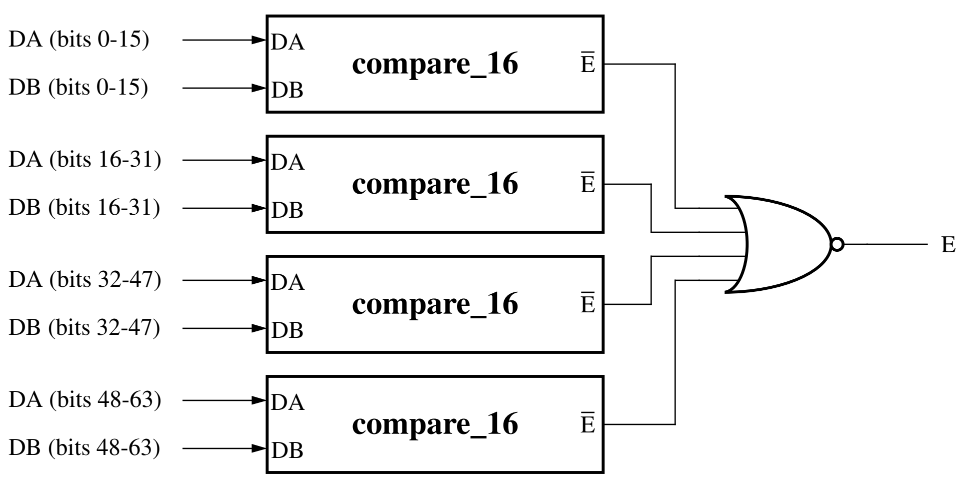


Figure A.3: Compare Block

A.4 Compare_16 Block

The compare_16 block outputs a 0 if all 16 bits of DA are equal to the corresponding bits of DB. The output is 1 otherwise.

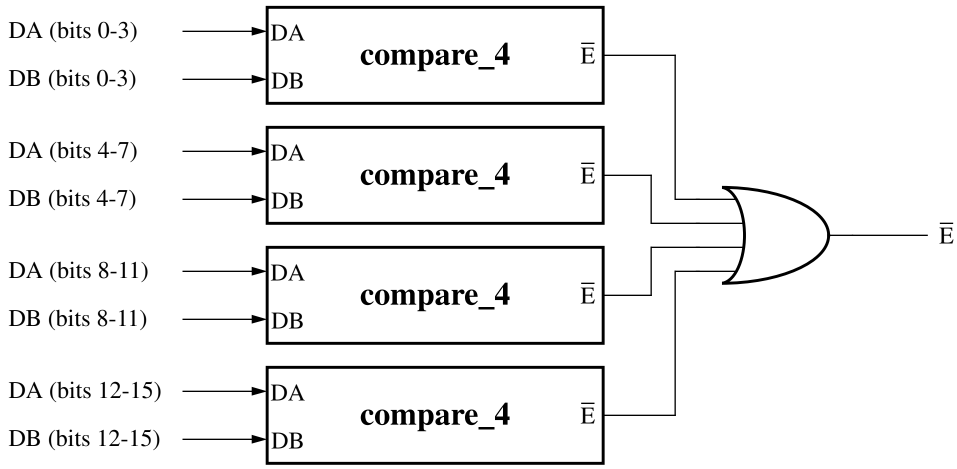


Figure A.4: Compare_16 Block

A.5 Compare_4 Block

The compare block outputs a 0 if all 4 bits of DA are equal to the corresponding bits of DB. The output is 1 otherwise.

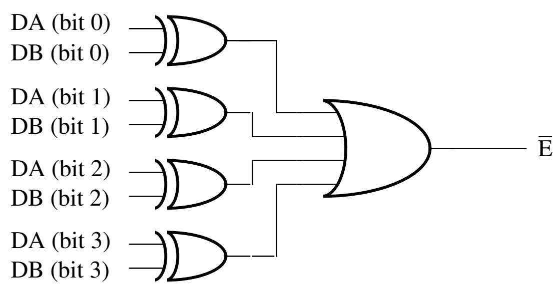


Figure A.5: Compare_4 Block

A.6 Plaintext and Ciphertext Blocks

The plaintext and ciphertext blocks are identical and each contains a 64-bit register. The inputs are a write line W, 3 address lines A, and 8 data lines D. If W = 0, the 64-bit register retains its old value. If W = 1, the data lines are written into the register byte specified by the address lines and the remaining register bytes retain their old values.

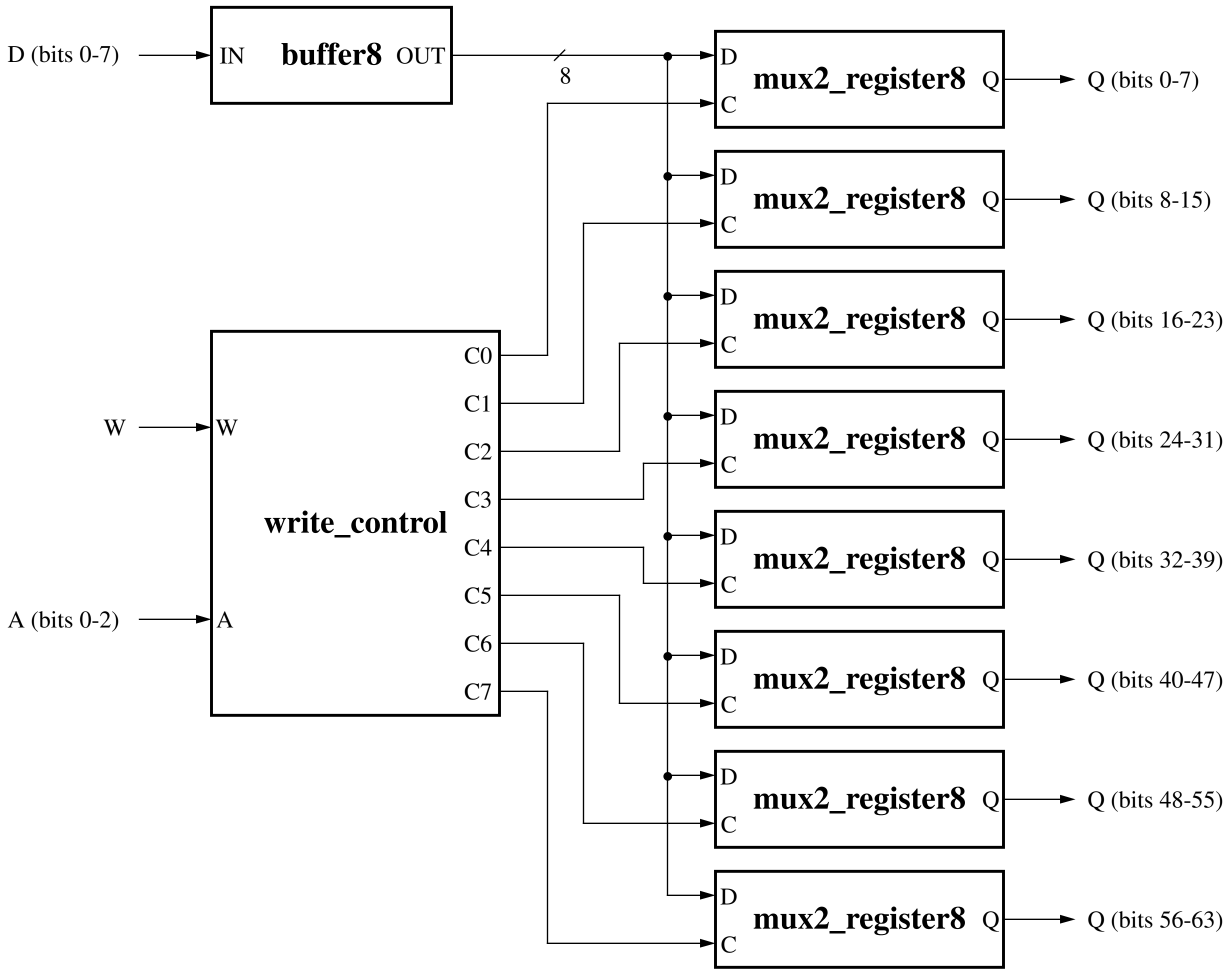


Figure A.6: Plaintext and Ciphertext Blocks

A.7 Write_control Block

The write_control block takes a write line W, 3 address lines A, and produces 8 output control lines. If W = 0, the outputs are all 0. If W = 1, the output corresponding to the address lines is a 1 and all other outputs are 0.

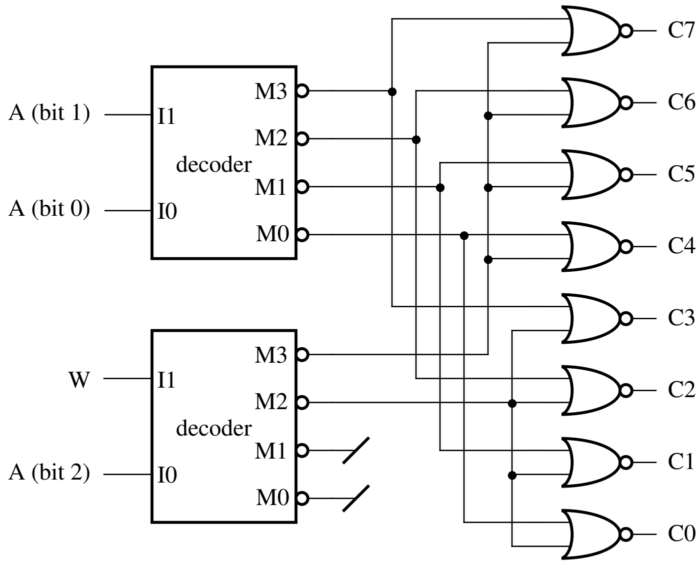


Figure A.7: Write_control Block

A.8 Mux2_register8 Block

The mux2_register8 block contains an 8-bit register. The inputs are a control line C and 8 data lines D. If C = 1, the register takes on the values on the data lines. If C = 0, the register retains its old value.

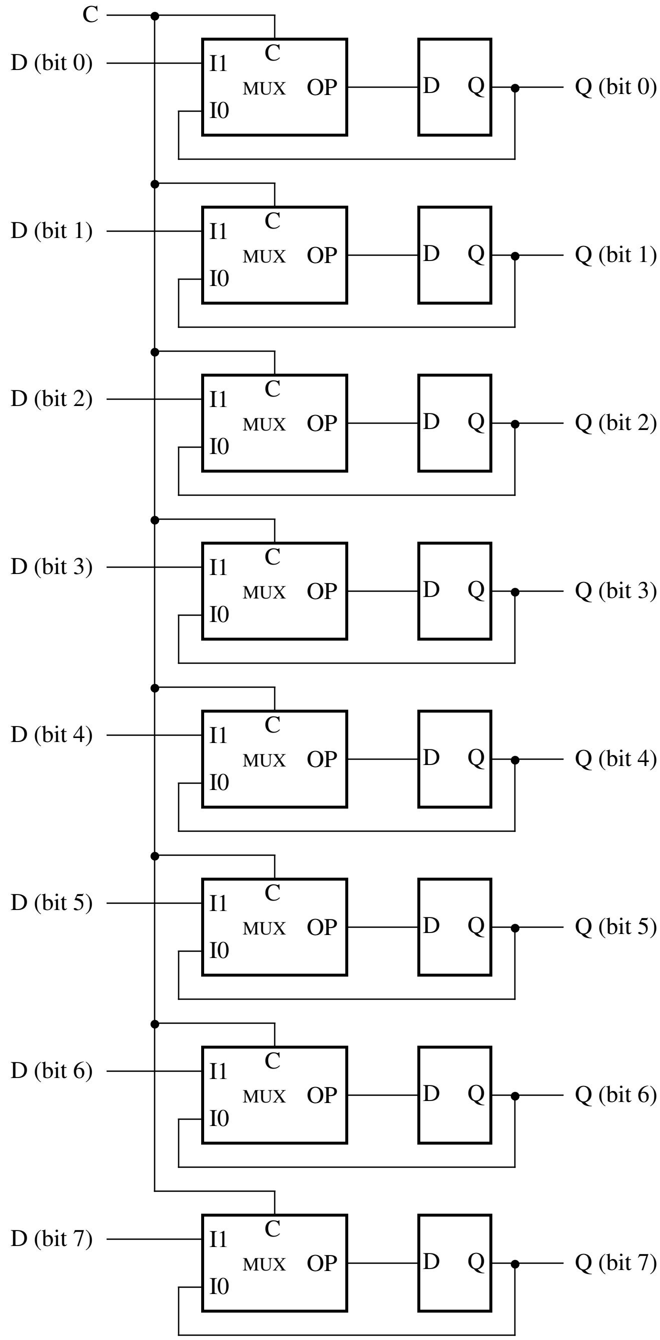


Figure A.8: Mux2_register8 Block

A.9 Key_counter Block

The key_counter block contains a 56-bit register. The inputs are the stop bit S indicating the state of the chip, a write line W, 3 address lines A, and 8 data lines D. If S = 1 and W = 0, the 56-bit register retains its old value. If S = 1 and W = 1, the byte of the register corresponding to the value of the address lines takes the value of the data lines, and all other register bytes retain their old values (if the address is 7, the entire register retains its old value). If S = 0, the register counts to the next key value. In terms of polynomials, the next key value is $kx \bmod g$, where k is the current key value and $g = x^{56} + x^7 + x^4 + x^2 + 1$ is the generator polynomial. This is the same as shifting the register left and if a 1 shifts from the top end, exclusive-or-ing the generator polynomial onto the register.

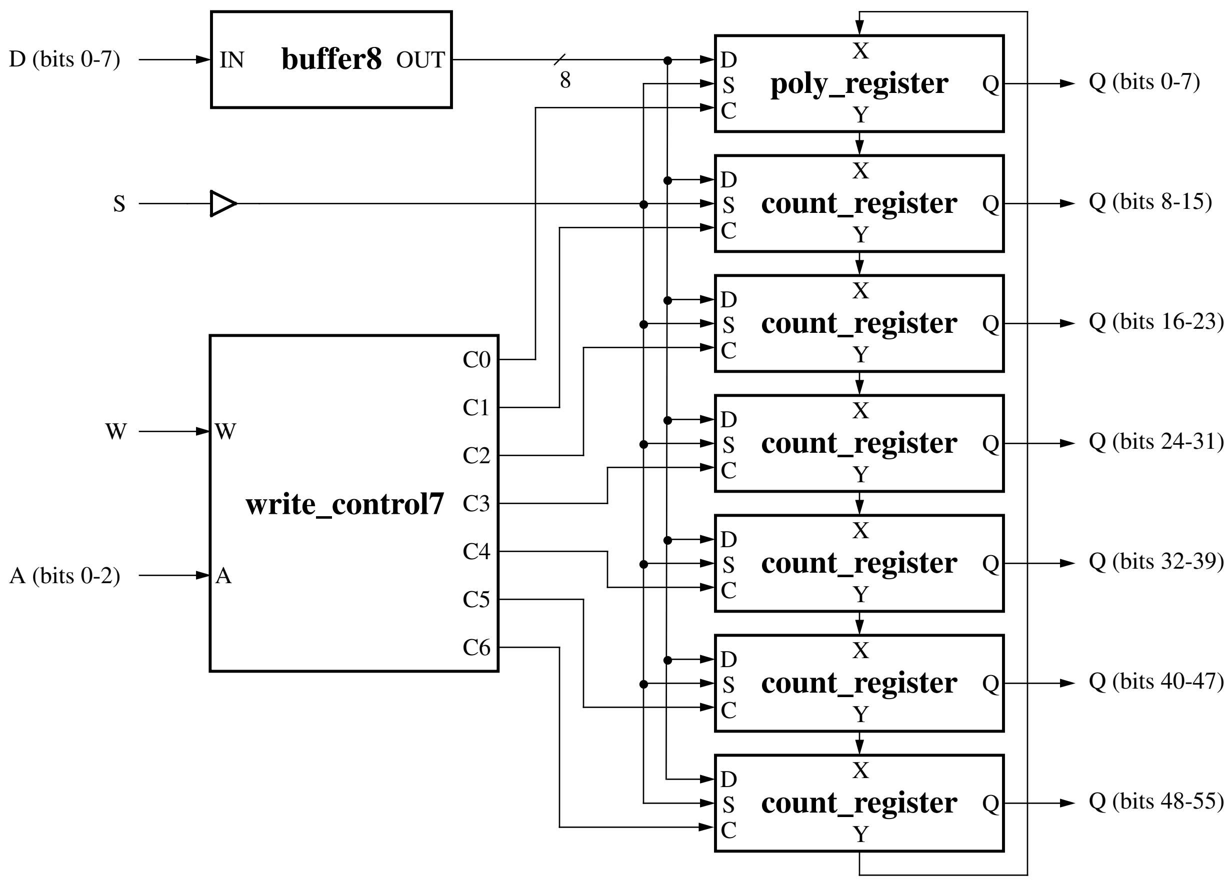


Figure A.9: Key_counter Block

A.10 Write_control7 Block

The write_control7 block takes a write line W, 3 address lines A, and produces 7 output control lines. If W = 0, all outputs are 0. If W = 1, the output corresponding to the address lines is a 1 and all other outputs are 0 (if the address is 7, all outputs are 0).

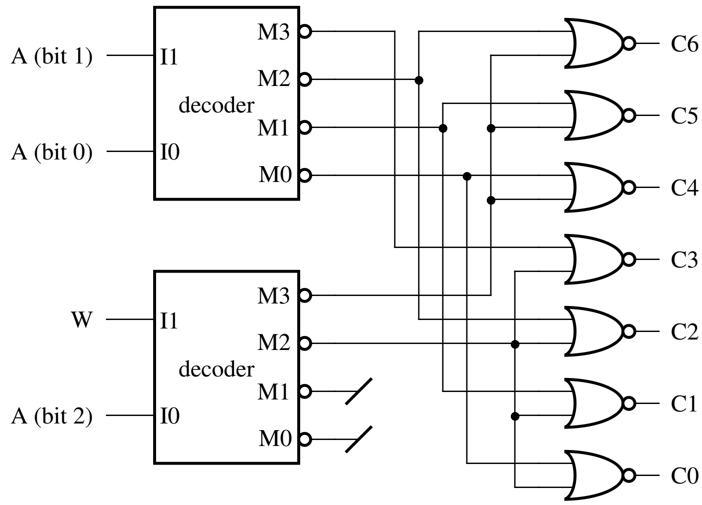


Figure A.10: Write_control7 Block

A.11 Poly_register Block

The poly_register block contains the least significant register byte of the key_counter. This is the only byte of the key_counter that is affected when we exclusive-or the generator polynomial onto the key_counter. The inputs are the stop bit S, the previous contents of the top bit of the key_counter X, a line to control writing C, and 8 data lines D. The outputs are the flip-flop outputs Q and the shifted out bit Y (which is just bit 7 of Q). If S = 1 and C = 0, the register retains its old value. If S = 1 and C = 1, the register takes the value of the data lines. If S = 0, the register is shifted left with X indicating whether we exclusive-or the bottom 8 bits of the generator polynomial onto the register.

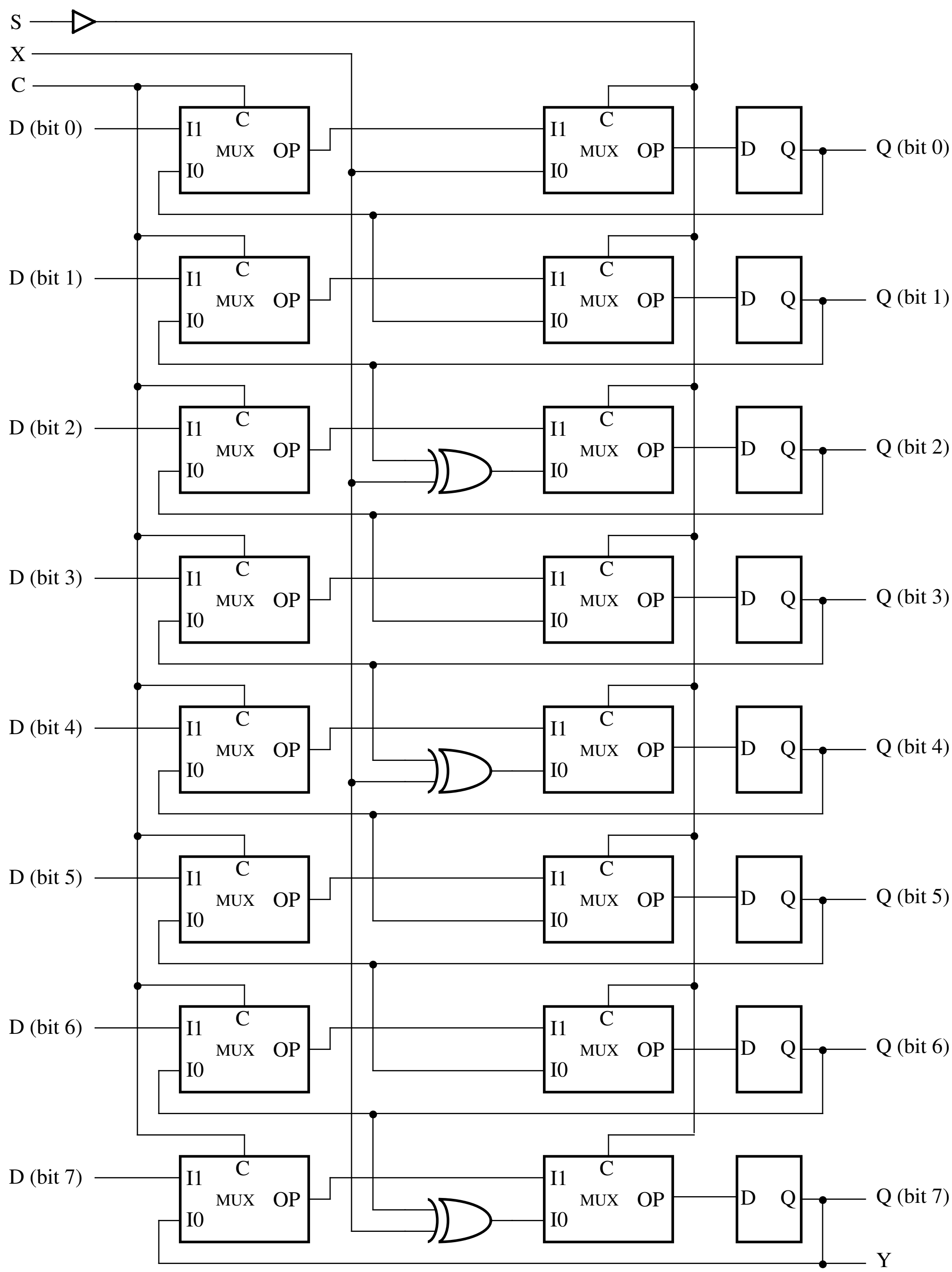


Figure A.11: Poly_register Block

A.12 Count_register Block

The count_register block contains a register byte of the key_counter. The inputs are the stop bit S, the bit to shift into the least significant flip-flop X, a line to control writing C, and 8 data lines D.

The outputs are the flip-flop outputs Q and the shifted out bit Y (which is just bit 7 of Q). If S = 1 and C = 0, the register retains its old value. If S = 1 and C = 1, the register takes the value of the data lines. If S = 0, the register is shifted left with X shifting into the least significant bit.

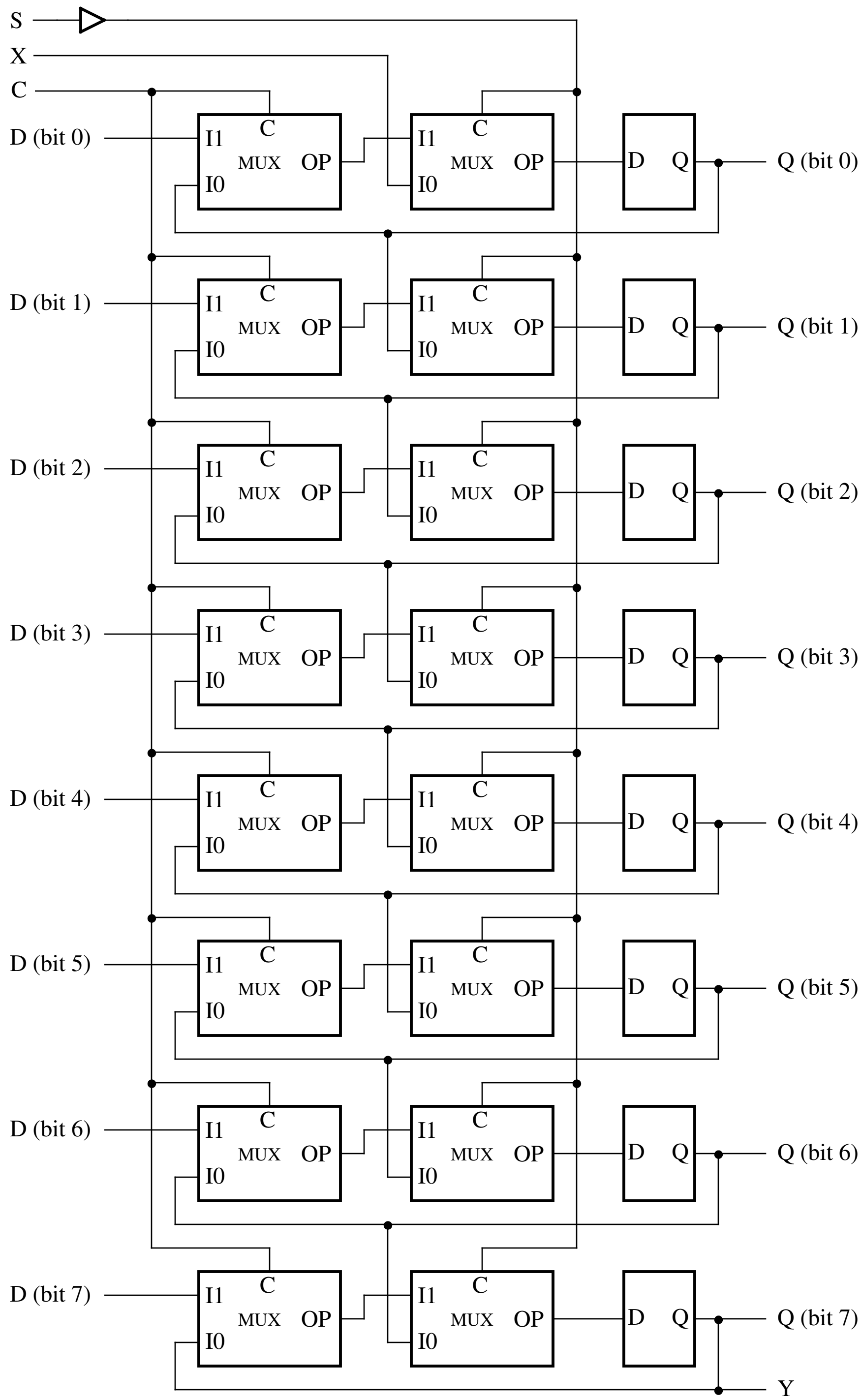


Figure A.12: Count_register Block

A.13 Output_key Block

The output_key block contains a 56-bit register. The inputs are the stop bit S, 3 address lines A, and 56 key lines K. If S = 0, the register takes on the value of the input key K. If S = 1, the register retains its old value. The output from this block is the byte B of the register specified by the address lines. If the specified address is 7, then all bits of B are 0.

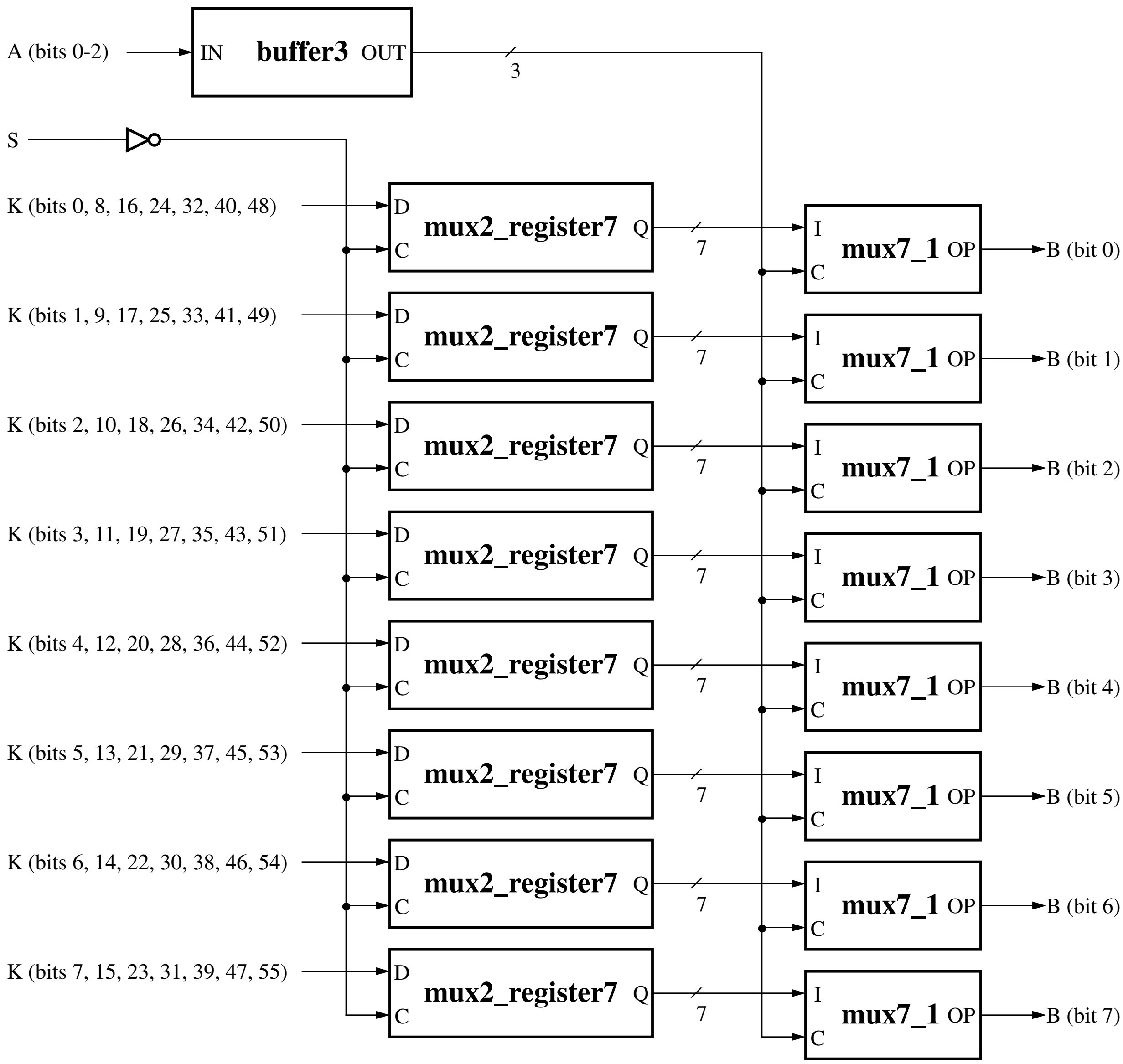


Figure A.13: Output_key Block

A.14 Mux2_register7 Block

The mux2_register7 block contains a 7-bit register. The inputs are a control line C and 7 data lines D. If C = 1, the register takes on the value of the data lines. If C = 0, the register retains its old value.

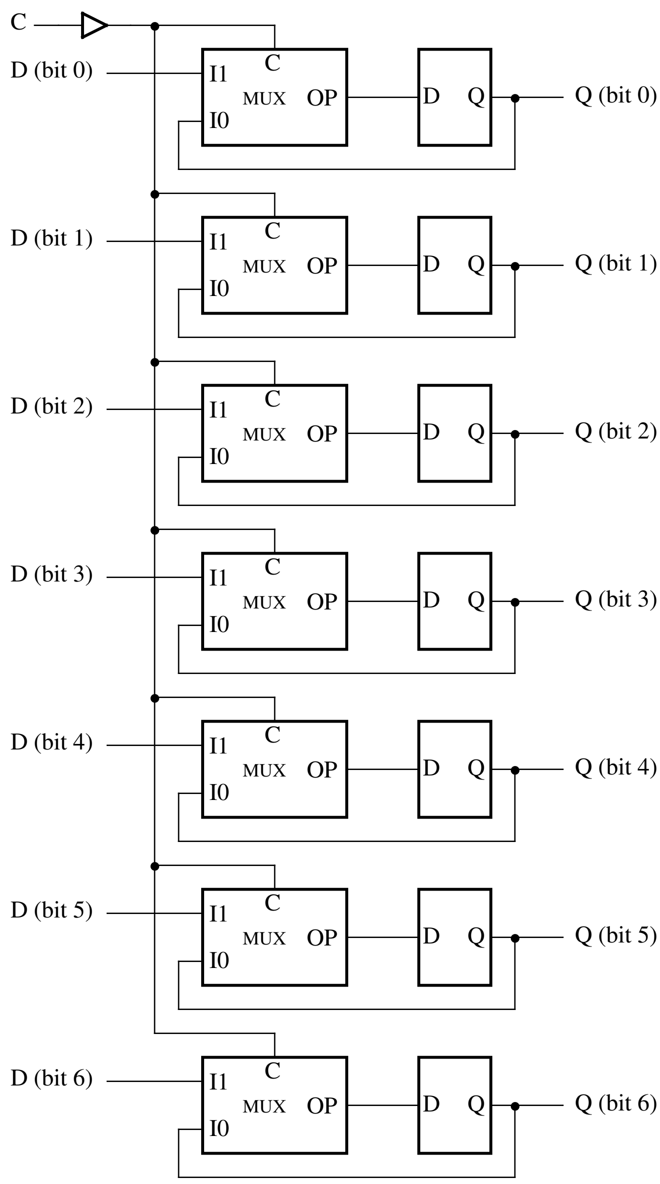


Figure A.14: Mux2_register7 Block

A.15 Mux7_1 Block

The inputs to the mux7_1 block are 3 control lines C and 7 input lines I. The output OP is equal to the input line that is specified by the control lines. If the control lines specify an address of 7, then OP = 0.

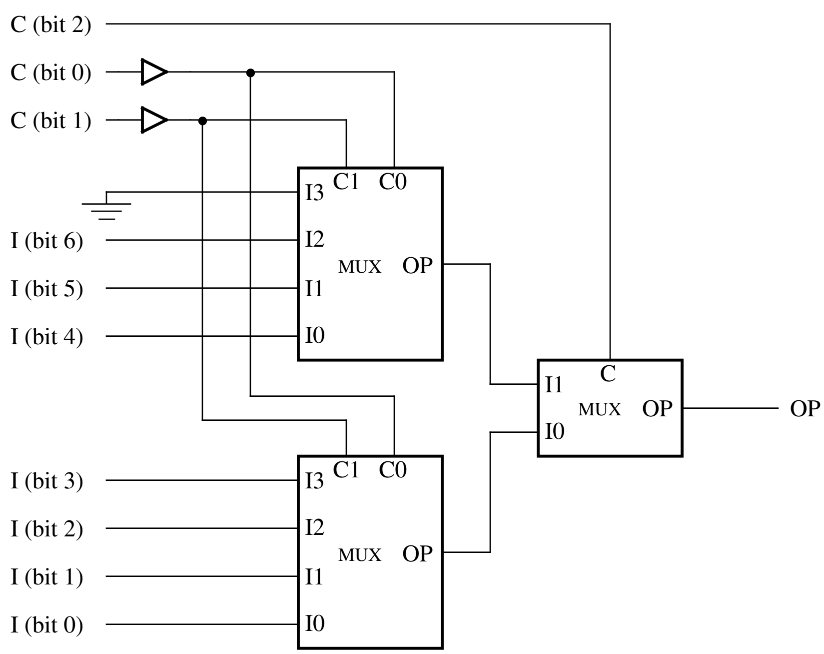


Figure A.15: Mux7_1 Block

A.16 Buffer8 Block

The buffer8 block simply buffers each of 8 input lines.

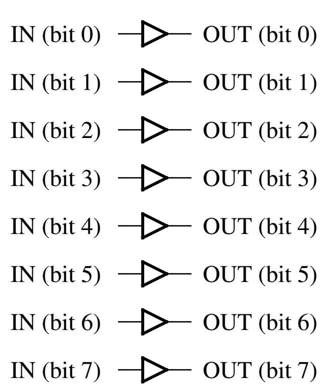


Figure A.16: Buffer8 Block

A.17 Buffer3 Block

The buffer3 block simply buffers each of 3 input lines.

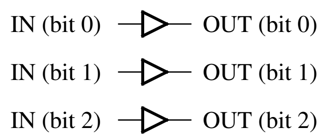


Figure A.17: Buffer3 Block

A.18 Pipelined_DES Block

The pipelined_DES block performs the DES calculations at the core of the key search chip. The inputs are a 64-bit plaintext P and a 56-bit input key IK. The outputs are a 64-bit ciphertext C and a 56-bit output key OK. At the top of the block, the DES initial permutation is applied to the plaintext, and the PC1 permutation is applied to the input key. This is followed by a 16-stage pipeline. Each stage of the pipeline consists of key scheduling applied to the key to produce a sub-key, a round of encryption using the current data and current sub-key, and a 120-bit register to pipeline the 64 bits of data and 56 bits of key. After the pipeline, the data is permuted to form the ciphertext, and the key is permuted to restore the original order of the key bits. It takes 16 clock ticks to produce a ciphertext, but there are 16 encryptions in progress at any given time. Intermediate results from 16 different encryptions are held in the 120-bit registers within the pipeline stages. On each clock tick, an encryption completes producing a ciphertext and the associated key.

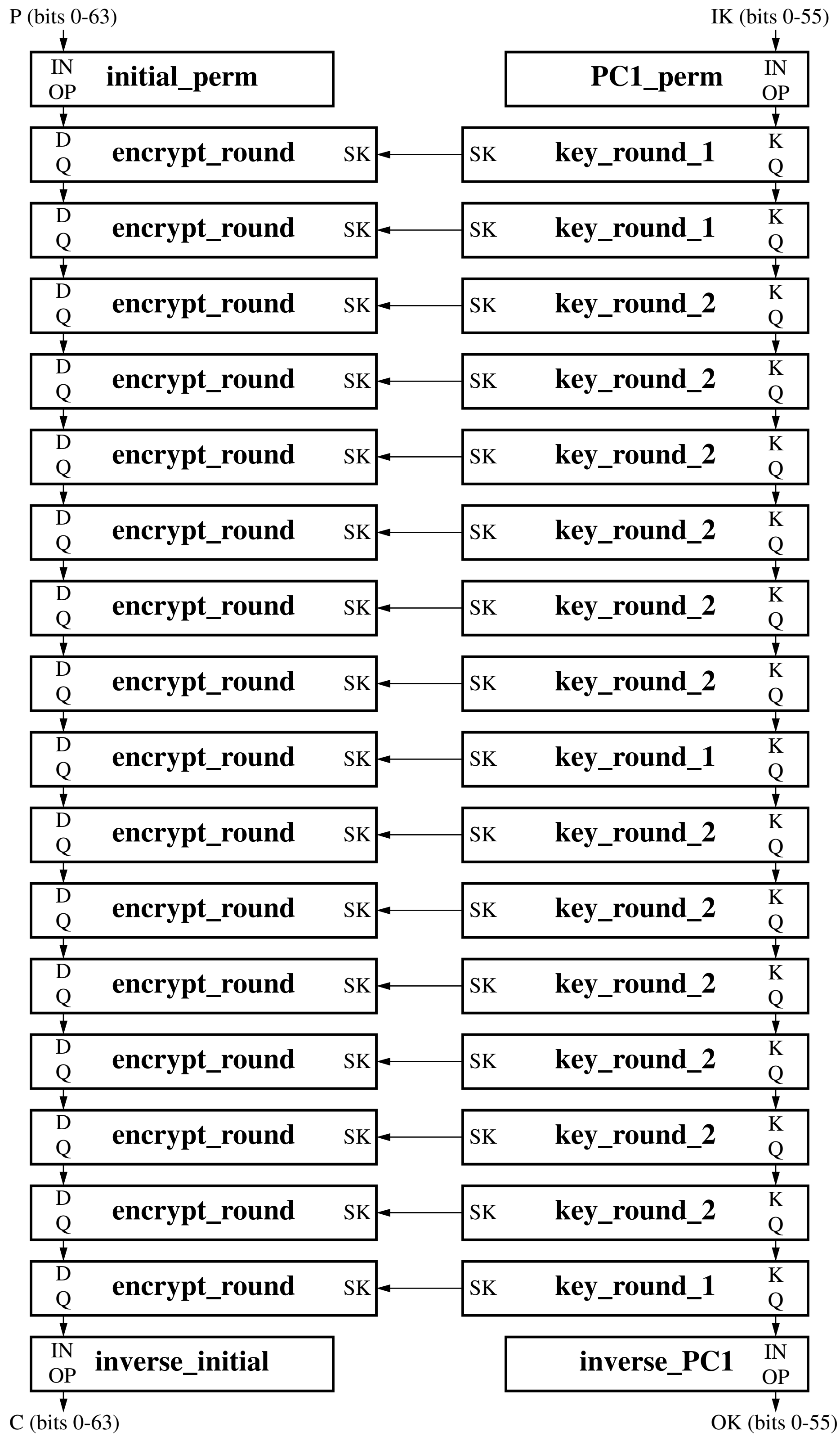


Figure A.18: Pipelined_DES Block

A.19 Encrypt_round Block

The encrypt_round block performs one round of DES not including key scheduling. The inputs are 64 bits of data D and 48 bits of sub-key SK. A round of DES begins with applying the E permutation to the right half of D expanding it to 48 bits, and exclusive-or-ing this result with the sub-key. The resulting 48 bits go through the S-boxes producing 32 bits which then go through the P permutation. The result of the P permutation is exclusive-or-ed with the left half of D to produce the right half of the output data for this round of DES. The left half of the output data for this round of DES is simply the same as the right half of the input data. The output data goes into a 64-bit pipelining register.

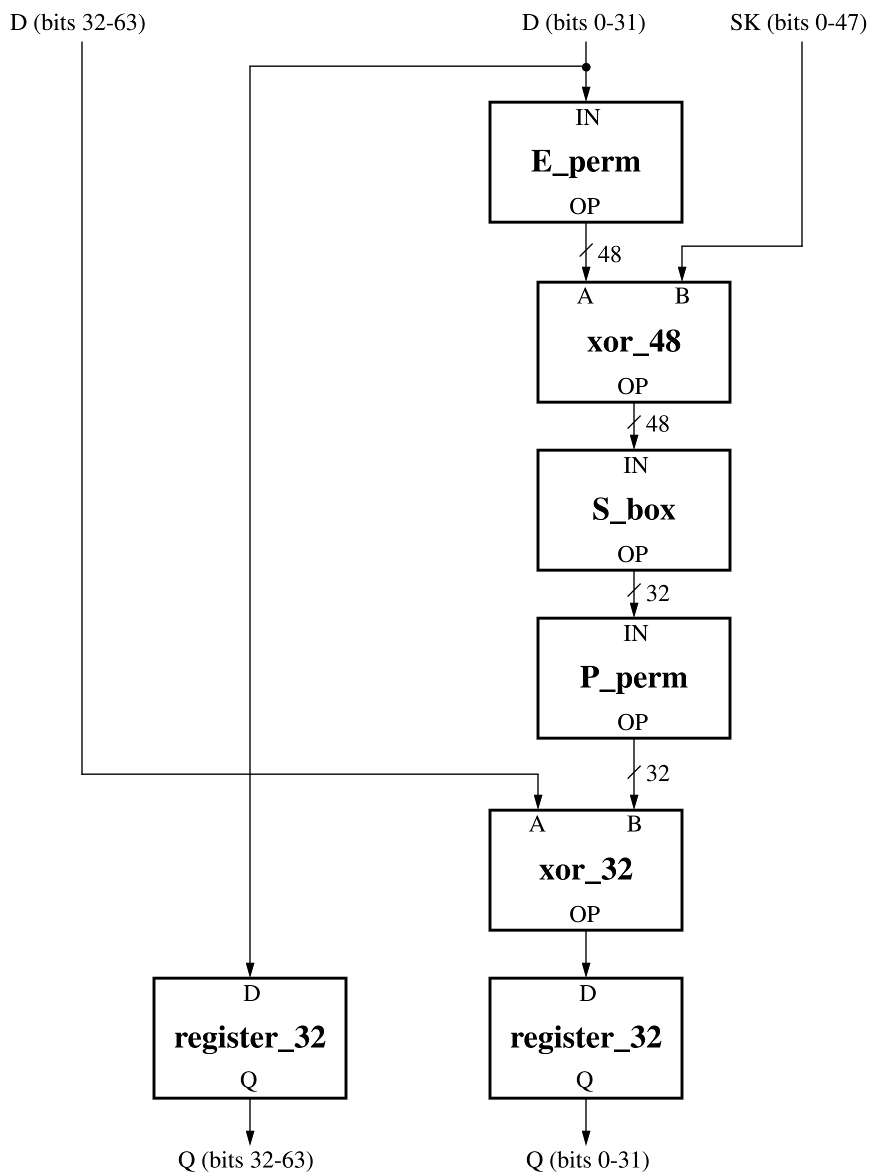


Figure A.19: Encrypt_round Block

A.20 S_box Block

There are 8 DES S-boxes. Each is a look-up table with 6 input bits and 4 output bits. The structure of each table is such that each of the outputs 0 to 15 occurs exactly 4 times. The approach to implementing the S-boxes taken here is to fully decode the 6 input bits, and then combine the 4 minterms for each of the non-zero table outputs. The 15 signals corresponding to the non-zero table outputs are then combined to form the 4 S-box output bits.

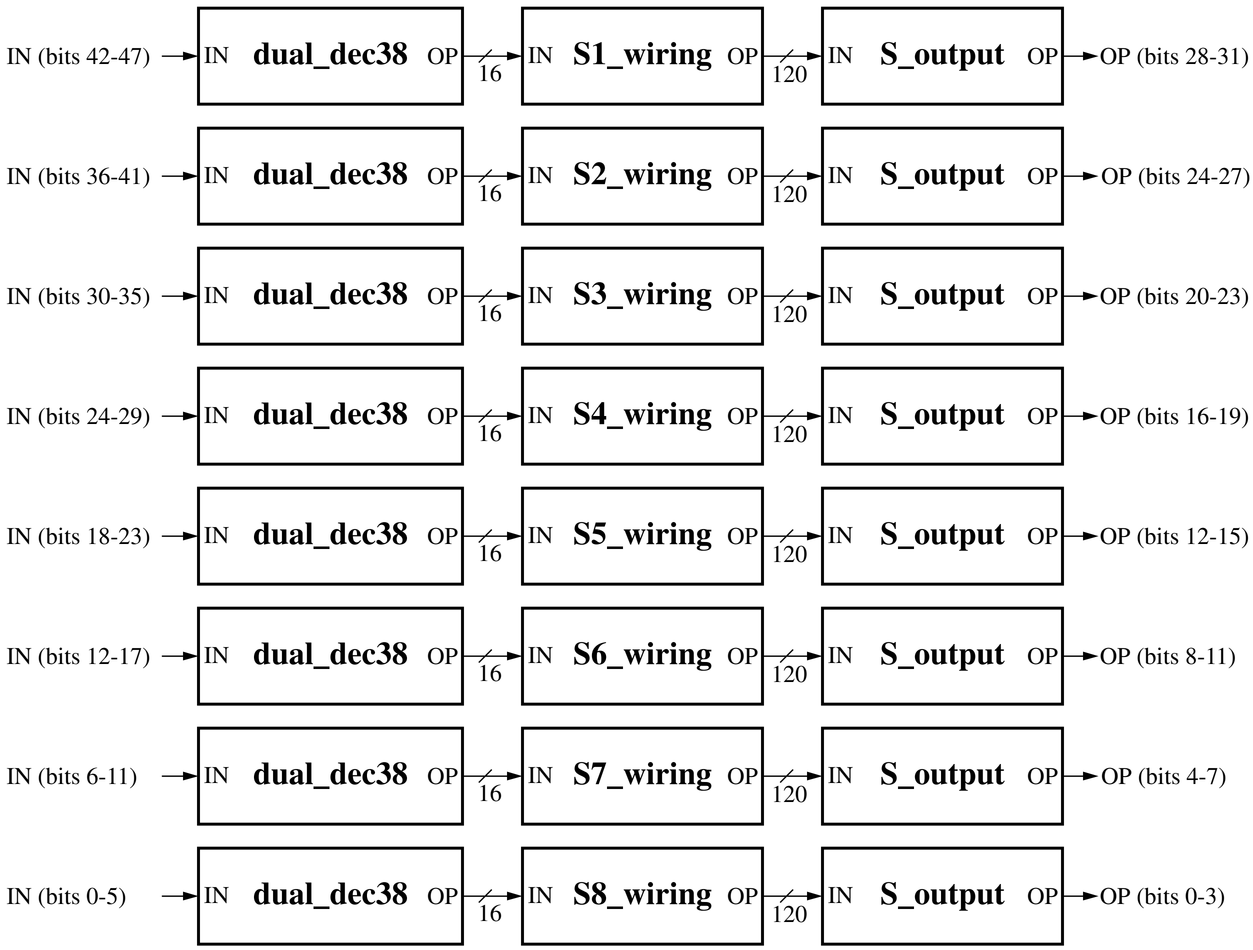


Figure A.20: S_box Block

A.21 Dual_dec38 Block

The dual_dec38 block takes 6 inputs which are split into two groups of 3 bits. Each group of 3 bits is fully decoded to 8 inverted outputs. The outputs from this block can be used to produce a minterm of the 6 inputs by combining one signal from each group of 8 outputs.

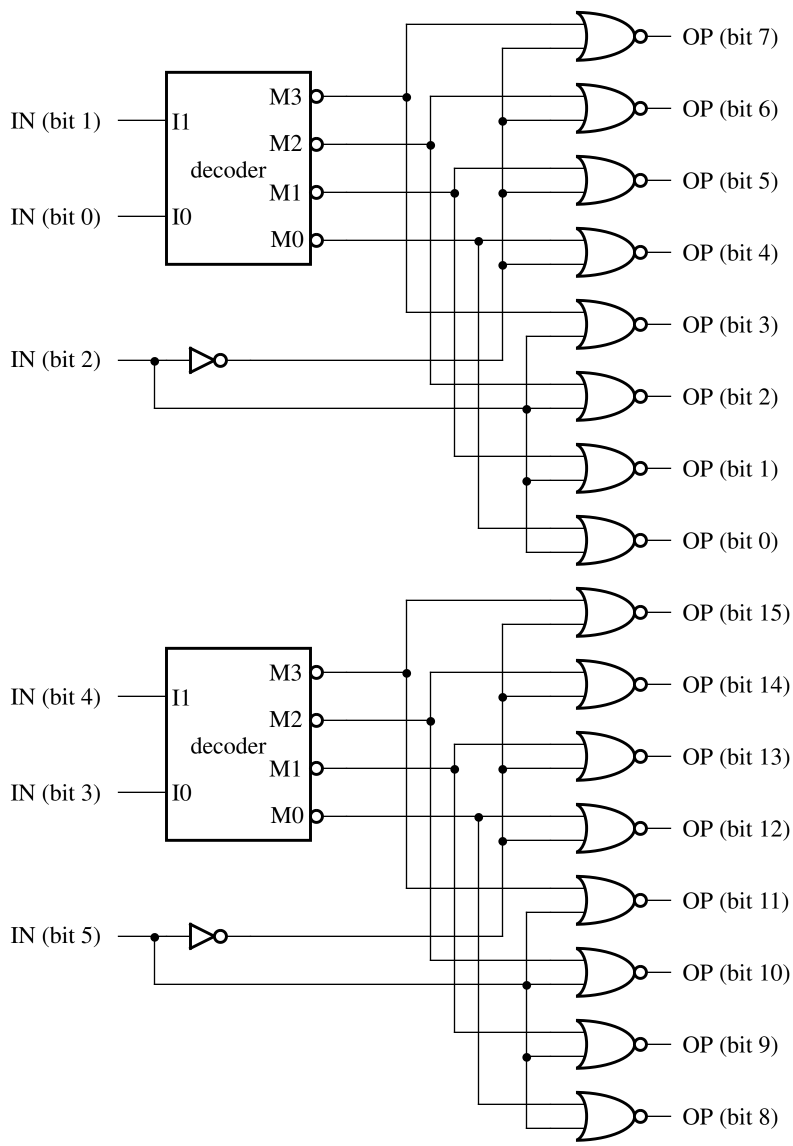


Figure A.21: Dual_dec38 Block

A.22 S1_wiring to S8_wiring Blocks

These wiring blocks do not contain any gates. Each output bit is connected to one of the input bits. Each pair of output bits specifies a minterm of the 6 S-box inputs. Each group of 8 output bits specifies the 4 minterms for one of the non-zero S-box table entry values. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

S1_wiring

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

S2_wiring

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

S3_wiring

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

S4_wiring

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

S5_wiring

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

S6_wiring

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

S7_wiring

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

S8_wiring

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

Figure A.22: S1_wiring to S8_wiring Blocks

A.23 S_output Block

The S_output block takes the permuted version of the decoded S-box inputs and produces the S-box outputs. The first 8 input bits are used to produce the signal associated with an S-box table entry of 1. The next 8 bits are used for a table entry of 2, and so on to a table entry of 15. The signals for S-box table entries are then combined to form the S-box outputs. Output bit 0 is created by or-ing the odd table entry signals. Output bit 1 is created by or-ing table entry signals 2, 3, 6, 7, 10, 11, 14, and 15. Output bit 2 is created by or-ing table entry signals 4, 5, 6, 7, 12, 13, 14, and 15. Output bit 3 is created by or-ing table entry signals 8 to 15.

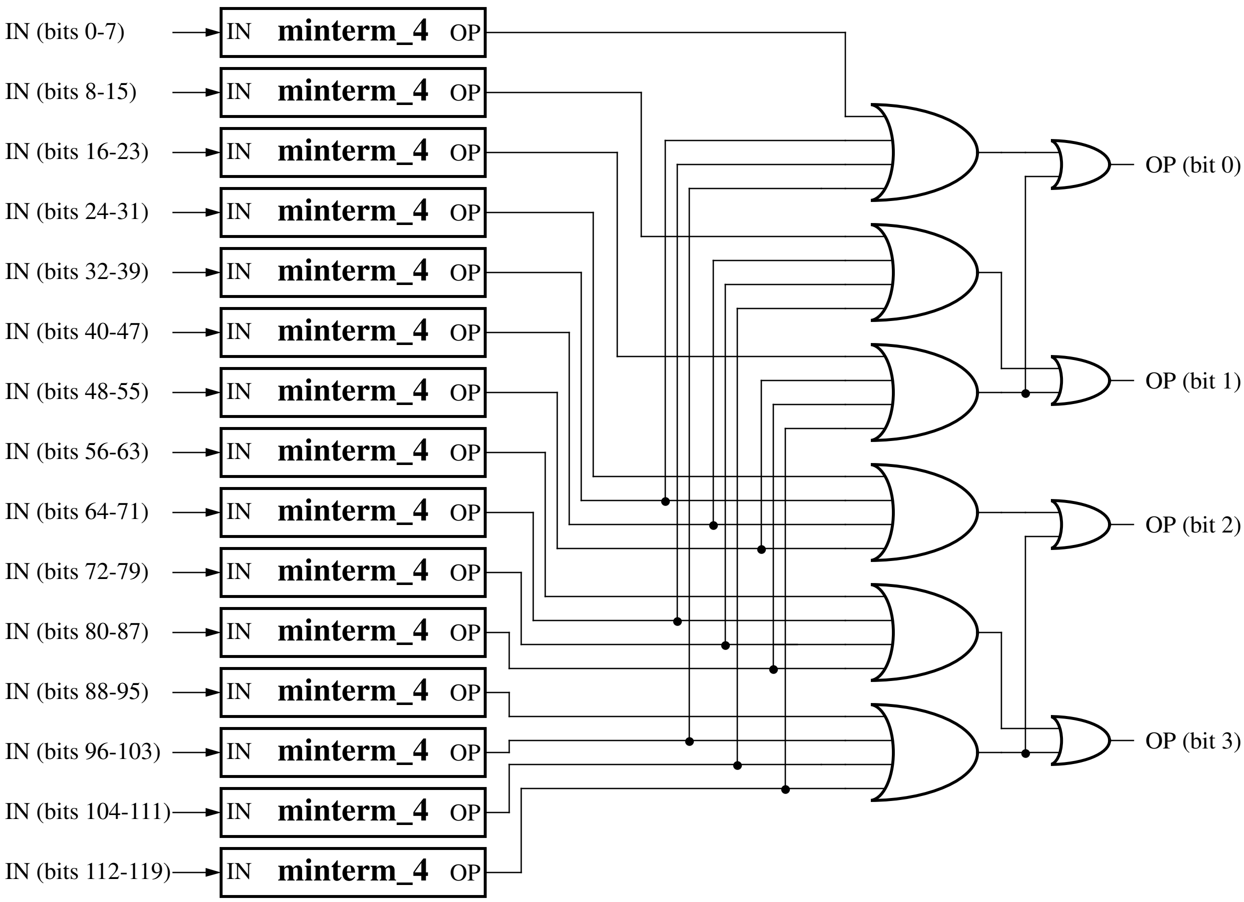


Figure A.23: S_output Block

A.24 Minterm_4 Block

The input to the minterm_4 block is 4 pairs of signals. The signals in each pair are and-ed together, and the resulting 4 signals are or-ed together.

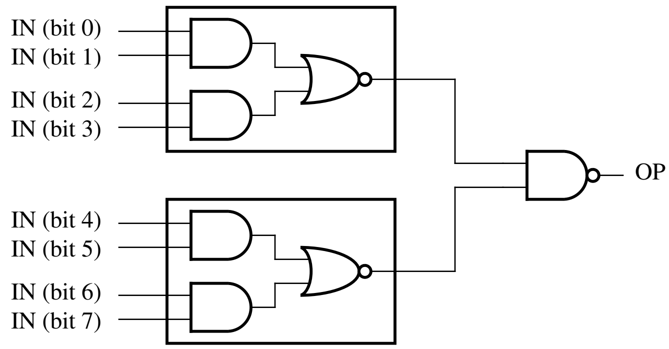


Figure A.24: Minterm_4 Block

A.25 Register_32 Block

The register_32 block consists of 32 flip-flops.

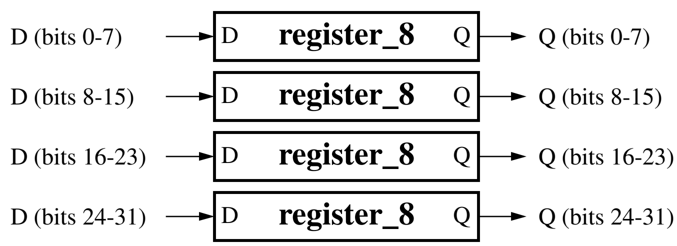


Figure A.25: Register_32 Block

A.26 Register_8 Block

The register_8 block consists of 8 flip-flops.

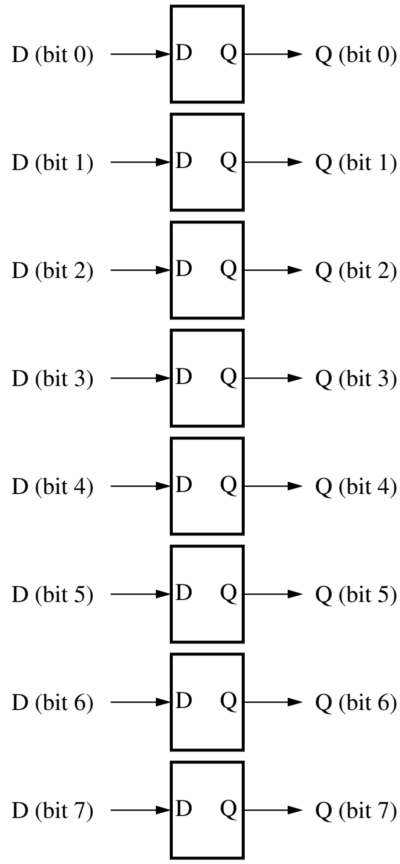


Figure A.26: Register_8 Block

A.27 Xor_48 Block

The xor_48 block exclusive-ors 48 pairs of signals together.

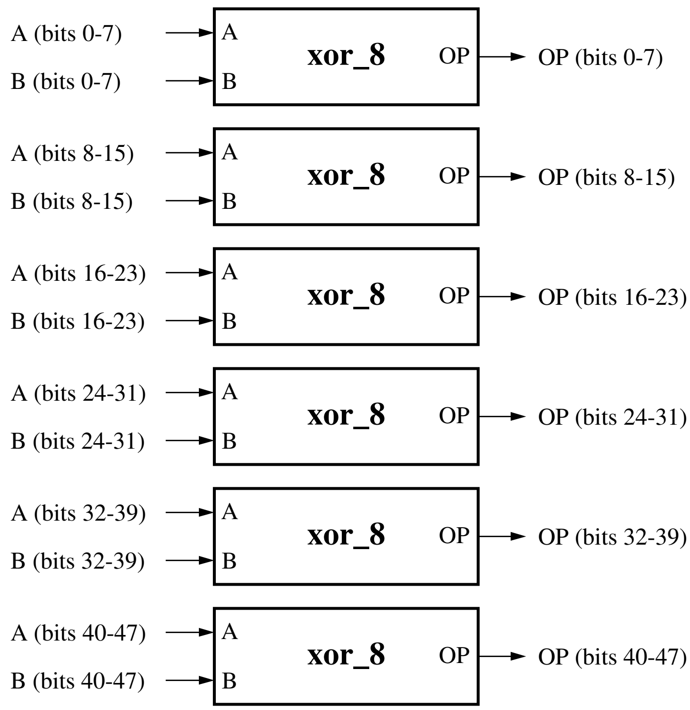


Figure A.27: Xor_48 Block

A.28 Xor_32 Block

The xor_32 block exclusive-ors 32 pairs of signals together.

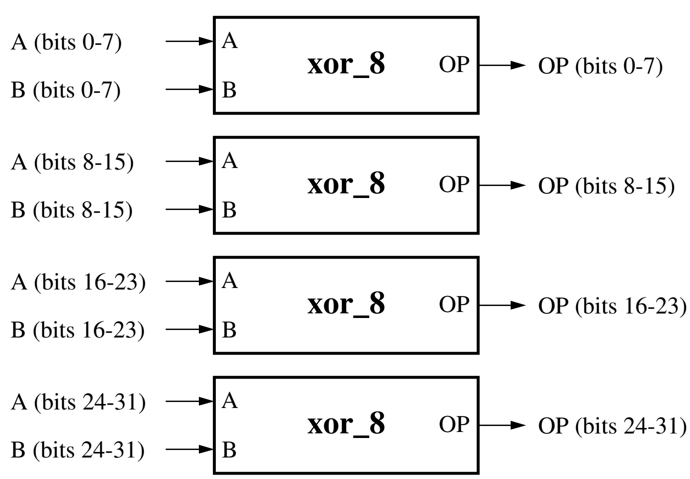


Figure A.28: Xor_32 Block

A.29 Xor_8 Block

The xor_8 block exclusive-ors 8 pairs of signals together.

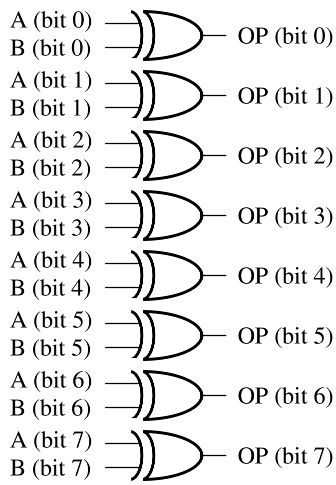


Figure A.29: Xor_8 Block

A.30 E_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The E permutation is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

0 31 30 29 28 27 28 27 26 25 24 23 24 23 22 21 20 19 20 19 18 17 16 15
16 15 14 13 12 11 12 11 10 9 8 7 8 7 6 5 4 3 4 3 2 1 0 31

Figure A.30: E_perm Block

A.31 P_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The P permutation is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

16 25 12 11 3 20 4 15 31 17 9 6 27 14 1 22 30 24 8 18 0 5 29 23 13 19 2 26 10 21 28 7

Figure A.31: P_perm Block

A.32 Key_round_1 Block

This block performs one round of the key scheduling part of DES. There are actually two types of key rounds. They only differ in the number of rotates performed at a certain point in the block. In this block, the 56-bit input key K is split into two 28-bit halves, each is rotated left one position, and the halves are rejoined to form the value which goes into the 56-bit pipeline register. The 48-bit sub-key output SK is formed by applying the PC2 permutation to the rotated key.

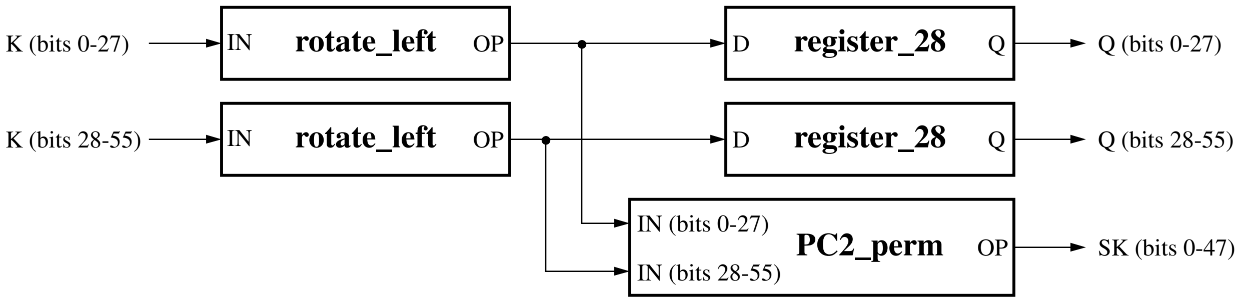


Figure A.32: Key_round_1 Block

A.33 Key_round_2 Block

This block performs one round of the key scheduling part of DES. There are actually two types of key rounds. They only differ in the number of rotates performed at a certain point in the block. In this block, the 56-bit input key K is split into two 28-bit halves, each is rotated left two positions, and the halves are rejoined to form the value which goes into the 56-bit pipeline register. The 48-bit sub-key output SK is formed by applying the PC2 permutation to the rotated key.

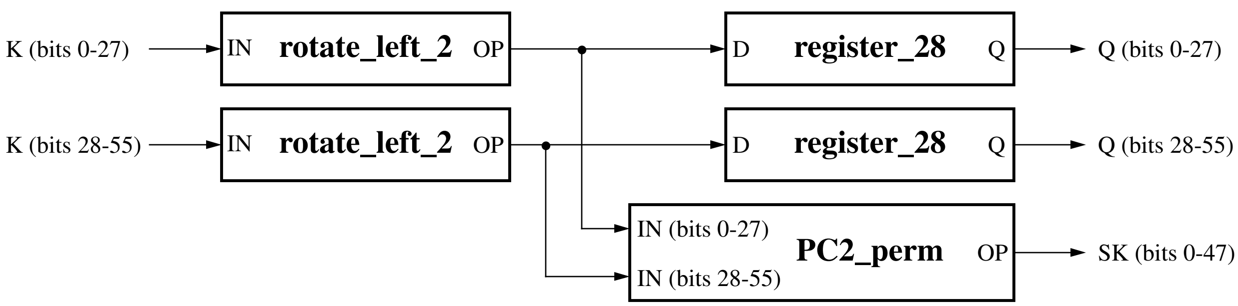


Figure A.33: Key_round_2 Block

A.34 Register_28 Block

The register_28 block consists of 28 flip-flops.

Figure A.34: Register_28 Block

A.35 Register_7 Block

The register_7 block consists of 7 flip-flops.

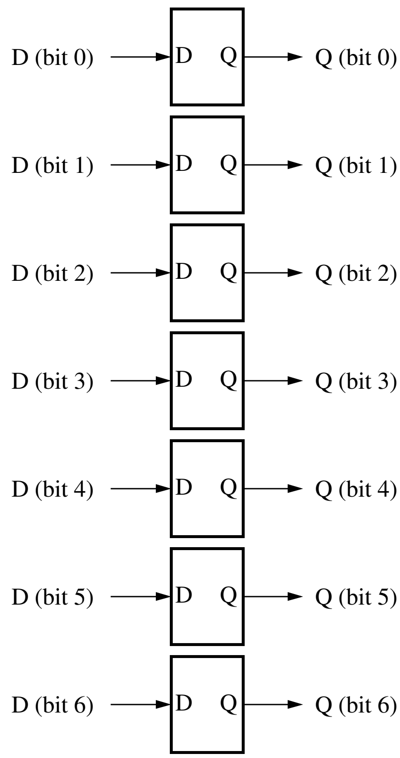


Figure A.35: Register_7 Block

A.36 PC2_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The PC2 permutation is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

42	39	45	32	55	51	53	28	41	50	35	46	33	37	44	52	30	48	40	49	29	36	43	54
15	4	25	19	9	1	26	16	5	11	23	8	12	7	17	0	22	3	10	14	6	20	27	24

Figure A.36: PC2_perm Block

A.37 Rotate_left Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The 28 input lines are rotated left one position. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 27

Figure A.37: Rotate_left Block

A.38 Rotate_left_2 Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The 28 input bits are rotated left 2 positions. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 27 26

Figure A.38: Rotate_left_2 Block

A.39 Initial_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The initial permutation (IP) is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

6 14 22 30 38 46 54 62 4 12 20 28 36 44 52 60 2 10 18 26 34 42 50 58 0 8 16 24 32 40 48 56
7 15 23 31 39 47 55 63 5 13 21 29 37 45 53 61 3 11 19 27 35 43 51 59 1 9 17 25 33 41 49 57

Figure A.39: Initial_perm Block

A.40 Inverse_initial Block

This block does not contain any gates. Each output bit is connected to one of the input bits. This permutation is not exactly the inverse of initial_perm. It also swaps the left and right halves. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

56 24 48 16 40 8 32 0 57 25 49 17 41 9 33 1 58 26 50 18 42 10 34 2 59 27 51 19 43 11 35 3
60 28 52 20 44 12 36 4 61 29 53 21 45 13 37 5 62 30 54 22 46 14 38 6 63 31 55 23 47 15 39 7

Figure A.40: Inverse_initial Block

A.41 PC1_perm Block

This block does not contain any gates. Each output bit is connected to one of the input bits. The PC1 permutation is one of the elements of DES. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

6	13	20	27	34	41	48	55	5	12	19	26	33	40	47	54	4	11	18	25	32	39	46	53	3	10	17	24
0	7	14	21	28	35	42	49	1	8	15	22	29	36	43	50	2	9	16	23	30	37	44	51	31	38	45	52

Figure A.41: PC1_perm Block

A.42 Inverse_PC1 Block

This block does not contain any gates. Each output bit is connected to one of the input bits. This block performs the inverse of the PC1_perm block. The following figure shows the input bit numbers for each output bit starting from the most significant output bit.

48	40	32	0	4	12	20	49	41	33	1	5	13	21	50	42	34	2	6	14	22	51	43	35	3	7	15	23
52	44	36	28	8	16	24	53	45	37	29	9	17	25	54	46	38	30	10	18	26	55	47	39	31	11	19	27

Figure A.42: Inverse_PC1 Block

A.43 External_interface Block

The external interface block is the interface between an external processor and the core of the key search chip. Included in the external_interface are 27 pads (not all of which are shown in Figure A.43) which serve as a buffer between the external pins and the inside of the chip. There are input pads for the chip enable, read/write line, tri-state line, and the 5 address lines. There are 8 input/output pads for the data lines. Not shown in Figure A.43 are the pad for the clock signal, 5 power pads, and 5 ground pads.

The flip-flops at the top of the block produce delayed versions of the chip enable. The chip enable is delayed to synchronize it to the chip's clock and to give the other signals coming into the chip time to become valid. When the delayed enable line makes a transition from 1 to 0 and the read/write line is a 0, then data is being written into the chip and address lines 3 and 4 select one of the write lines (write_s, write_kc, write_c, and write_p) to become a 1. Address lines 0, 1, and 2 are passed through as outputs from the external_interface. If the undelayed enable line is 0, the read/write line is a 1, and the tri-state line is a 1, then data is being read from the chip and the input/output pads on the data lines are set as outputs. The purpose of the tri-state line is to force the chip into an input mode to allow testing of a board. When data is being read from the chip and address lines 3 and 4 are both 1, then the stop bit is put out on data pin 0 and the rest of the data pins are 0. When data is being read from the chip, address line 3 is a 0, and address line 4 is a 1, then the output from the output_key block goes out on the data pins. A read from any other address causes zeros to be put out on the data pins. When data is written into the chip, the signals on the data pins are passed through to the data_out signals.

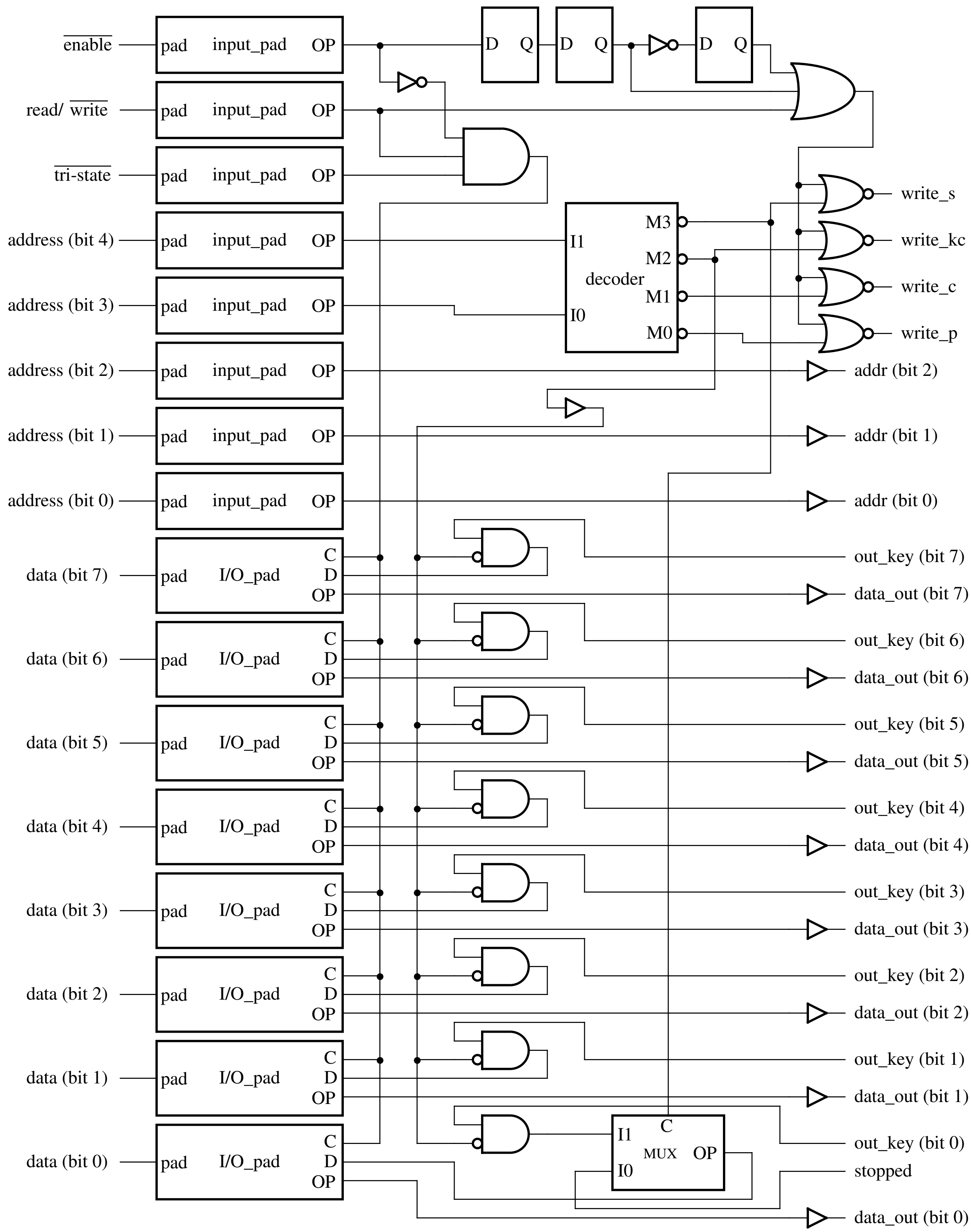
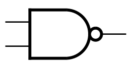
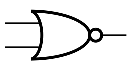
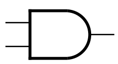
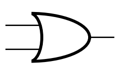
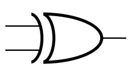
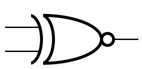
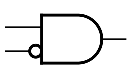
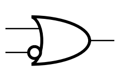
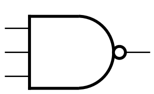
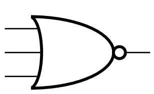
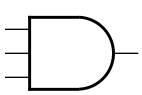
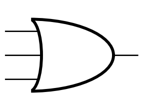
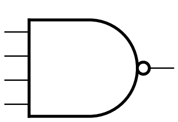
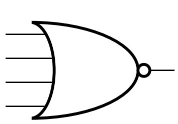
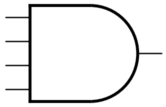


Figure A.43: External_interface Block

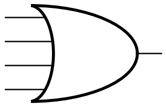
A.44 Standard Cells

This section contains descriptions of each of the standard cells, their symbols, and their areas (in units of sites, where a site is 3.6 µm × 54.4 µm).

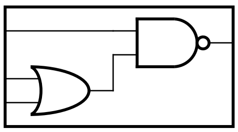
	2 sites	inverter
	3 sites	buffer
	3 sites	2-input nand gate
	3 sites	2-input nor gate
	4 sites	2-input and gate
	4 sites	2-input or gate
	4 sites	2-input exclusive-or gate
	5 sites	2-input exclusive-nor gate
	4 sites	2-input and gate with one inverted input
	4 sites	2-input or gate with one inverted input
	4 sites	3-input nand gate
	4 sites	3-input nor gate
	5 sites	3-input and gate
	5 sites	3-input or gate
	5 sites	4-input nand gate
	7 sites	4-input nor gate



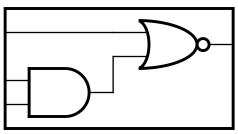
6 sites 4-input and gate



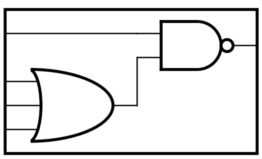
6 sites 4-input or gate



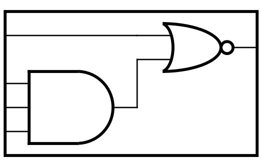
4 sites composite gate: 1,2 or-and-invert



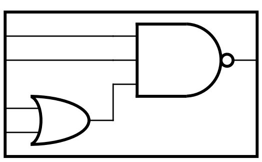
4 sites composite gate: 1,2 and-or-invert



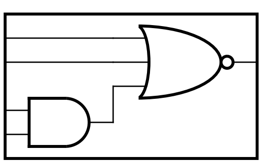
5 sites composite gate: 1,3 or-and-invert



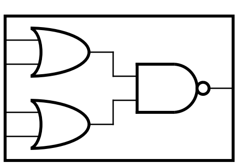
5 sites composite gate: 1,3 and-or-invert



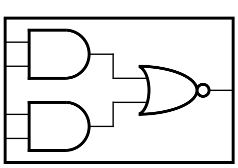
5 sites composite gate: 1,1,2 or-and-invert



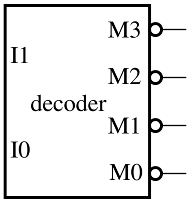
5 sites composite gate: 1,1,2 and-or-invert



6 sites composite gate: 2,2 or-and-invert



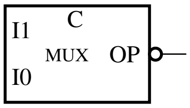
6 sites composite gate: 2,2 and-or-invert



6 sites

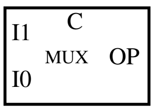
2-input inverting decoder:

I1	I0	M3	M2	M1	M0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1



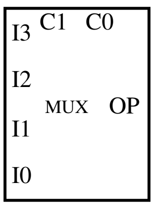
5 sites

2-input inverting multiplexor:
if C = 1, then OP = $\overline{\mathrm{I1}}$
if C = 0, then OP = $\overline{\mathrm{I0}}$



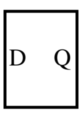
7 sites

2-input multiplexor:
if C = 1, then OP = I1
if C = 0, then OP = I0



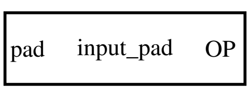
19 sites

4-input multiplexor:
if C1 = 1 and C0 = 1, then OP = I3
if C1 = 1 and C0 = 0, then OP = I2
if C1 = 0 and C0 = 1, then OP = I1
if C1 = 0 and C0 = 0, then OP = I0

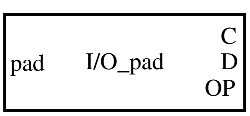


10 sites

flip-flop:
Q holds its value until the rising edge of the system-wide clock when Q takes the value of the input D.



An input pad is a buffer between an external input pin and the inside of the chip. The output is simply equal to the pad input. If the pad input is floating, the output is 1.



An I/O pad is a buffer between an external I/O pin and the inside of the chip. If C = 1, the pad is an output and the logic level at D is output. If C = 0, the pad is an input and OP is equal to the pad input. If C = 0 and the pad input is floating, then OP = 1.

Appendix B: Key Search Board Design

The main goal in designing a key search board is to pack as many key search chips onto it as possible with a minimum of overhead and additional cost. The majority of the overhead is microcontrollers to control the key search chips. The microcontroller selected for this design is the Motorola MC68HC705C8 [13]. The main features of this member of the 6805 family are 8192 bytes of one-time programmable ROM, 176 bytes of RAM, a serial peripheral interface (SPI), a serial communications interface (SCI), 24 general-purpose I/O pins, and it is available in a 44-pin PLCC (surface-mount) package. The masked ROM version of this chip is less expensive, but the one-time programmable version was selected to avoid masking charges.

Each microcontroller can control 10 key search chips using its 24 I/O pins. Eight data lines, five address lines, and a read/write line are connected to all 10 key search chips. Each of the remaining 10 I/O pins goes to the enable pin of one of the key search chips.

The board needs a master microcontroller (also an MC68HC705C8) to control all of the other microcontrollers. The master is connected to its slaves using the serial peripheral interface. The SPI is designed to connect a master to several slaves directly without any other logic. The master microcontroller's serial communications interface is used as the board's external interface.

Other components that will be required are an octal buffer/line driver (74HC244) for the board's external interface lines, a 50 MHz oscillator, clock drivers to drive the clock to the key search chips, a counter (74HC191) to divide the clock by 16 to produce a suitable clock for the microcontrollers, a clock driver for the microcontrollers, decoupling capacitors on the underside of the board, a fuse for the power entry to the board, and the unpopulated board itself.

For the shelf and frame technology that this system is designed for, the usable area of each board is 9 inches $\times$ 13 inches. Using surface-mount components for everything except the fuse, there is enough room on the board for 120 key search chips.[1] To handle this number of chips, 5 clock drivers and 13 microcontrollers are needed. A possible layout of this board is shown in Figure B.1. The unlabeled chips are the key search chips. The layout shows connections between certain chips indicating the control of the slave microcontrollers by the master and the control of key search chips by each slave.

The board itself is a 6-layer board. There is one layer for each of power and ground. The remaining 4 layers are for routing clocks and the connections between chips. Because most of the connections are very localized, it may be possible to use a 4-layer board, but we will assume that a 6-layer board is necessary. The cost of fabricating, populating, and testing the board is about $300.

[1] A 44-pin PLCC package covers 0.72 inch $\times$ 0.72 inch, and 0.2 inch spacing between chips is required.

The following table shows the costs to build this board.

Component	Quantity	Unit Cost ($)	Extended ($)
key search chip	120	10.50	1260
MC68HC705C8	13	8	104
50 MHz oscillator	1	4	4
clock driver	5	8	40
74HC191	1	1	1
74HC244	1	1	1
decoupling capacitors and fuse			15
board and cost to populate and test it			300
Total Board Cost			$ 1725

The total power consumption of all of the key search chips is 120×450 mW = 54 W. Adding the power consumption of the other components brings the total board power to 60 W. This is fairly high power consumption. The issues of providing this power and cooling the boards will need to be addressed at the shelf and frame level.

The program in the slave microcontroller is not very complicated. Through its SPI it receives tasks which consist of a plaintext P, a ciphertext C, a starting key K, and chip step value s. The chip step value specifies the spacing between starting keys for each of the 10 key search chips. If each chip is responsible for m keys, then the chip step value supplied to the slave microcontroller is $s = x^m \bmod g$, where g is the generator polynomial. The 10 starting keys are K, $Ks \bmod g$, $Ks^2 \bmod g$, …, $Ks^9 \bmod g$. To get from one chip's starting key to the next, the slave microcontroller performs a polynomial multiply of the previous starting key and the chip step value and reduces the result modulo the key_counter's generator polynomial g. This is a fairly simple calculation. After starting the key search chips, the slave polls them until one of them has the desired key or a new task comes in. Any found keys are reported back to the master. The slave should also be capable of performing a simple known-answer test on each of its key search chips and reporting any failures to the master.

The master must be able to take tasks through its SCI which consist of P, C, starting key K, slave step value t, and chip step value s. The values P, C, and s are passed directly to each slave. The starting keys sent to the 12 slaves are K, $Kt \bmod g$, $Kt^2 \bmod g$, …, $Kt^{11} \bmod g$. The master should also be capable of performing a known-answer test on each of its slaves. The master reports any found keys or hardware failures over its SCI.

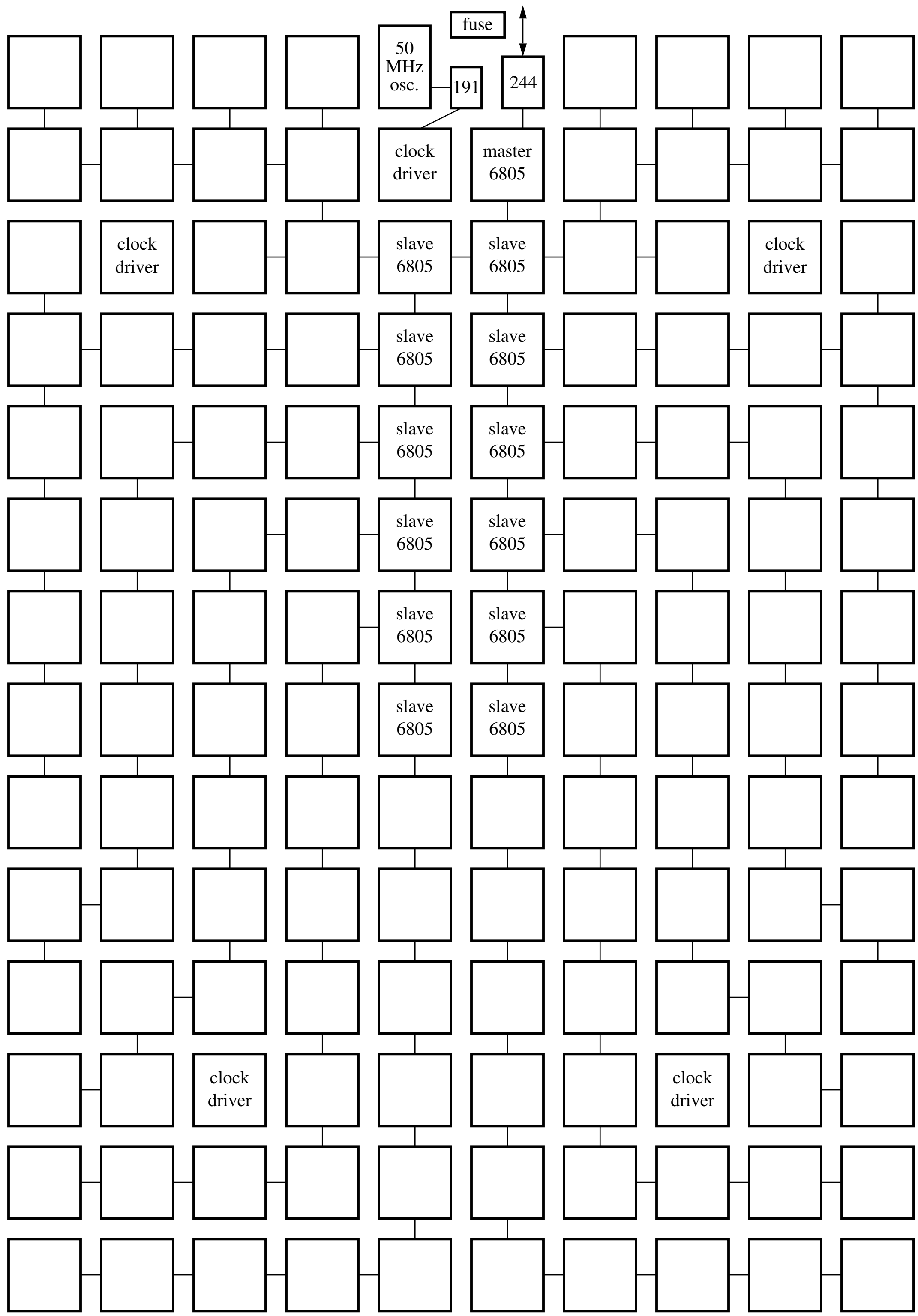


Figure B.1: Key Search Board Layout

Appendix C: Key Search Frame Design

Given that we have key search boards as described in Appendix B, we need a structure to hold, interconnect, and control all of these boards. An existing technology for this purpose was selected as the basis for this design. Boards are housed in frames. Each frame consists of 4 stacked shelves, and each shelf has 26 board slots and 4 slots for power supplies.

Each shelf requires a board to control the key search boards. This controller board needs only a processor and sufficient logic to interface serially to the key search boards. The controller boards could be made for about $500 each. Although there are 25 slots left in the shelf, the total power requirements for 25 key search boards at 60 W each is too much for the 4 shelf power supplies. To stay comfortably within the power supply limits and leave more space for power dissipation, only 12 key search boards are on each shelf. The following table shows the cost of a shelf.

Component	Quantity	Unit Cost ($)	Extended ($)
key search board	12	1725	20700
controller board	1	500	500
power supply	4	300	1200
mechanical structure, board interconnections, testing		1000	1000
Total Shelf Cost			$ 23400

Each frame consists of 4 shelves and fans for cooling. The frame also requires a PC to control the 4 shelf controller boards and to check the all-zero key. The following table shows the frame cost.

Component	Quantity	Unit Cost ($)	Extended ($)
key search shelf	4	23400	93600
mechanical structure, fans, testing		4000	4000
PC	1	2000	2000
Total Frame Cost			$ 99600

A frame costs about $100 000 and contains 5760 key search chips each of which can test 50 million DES keys per second. Overall, a frame can test 2.88×10^{11} keys per second.

Appendix D: Reliability of the Key Search Machine

A common criticism of past key search designs is that they are not sufficiently reliable to complete a single key search without breaking down. Today, this is no longer a concern. To show that the key search machine described in this paper is sufficiently reliable, we will calculate its mean time between failures (MTBF). All information on component reliability was obtained through internal BNR component testing.

The typical failure rate of chips designed in the CMOS process used for the key search chip is 10 FITs.[1] The actual failure rate for the key search chip may be somewhat less because it is small and contains no analog circuitry, but we will use a failure rate of 10 FITs. The total failure rate for a key search board is shown in the following table.

Component	Quantity	Unit Reliability (FITs)	Total Reliability (FITs)
key search chip	120	10	1200
MC68HC705C8	13	20	260
50 MHz oscillator	1	3	3
clock driver	5	10	50
74HC191	1	4	4
74HC244	1	4	4
fuse	1	3	3
decoupling capacitors	400	0.1	40
board	1	100	100
Total Board Failure Rate			1664

A key search shelf consists of 12 key search boards, a controller board and four power supplies. The controller board and power supplies each contain fewer components than a key search board, and so we will estimate their failure rates at 1000 FITs. The total failure rate for a shelf is shown in the following table.

Component	Quantity	Unit Reliability (FITs)	Total Reliability (FITs)
key search board	12	1664	19968
controller board	1	1000	1000
power supply	4	1000	4000
board interconnections	1	100	100
Total Shelf Failure Rate			25068

A key search frame consists of four shelves and a PC. We will estimate a PC's failure rate at 5000 FITs. The total failure rate for a frame is shown in the following table.

Component	Quantity	Unit Reliability (FITs)	Total Reliability (FITs)
key search shelf	4	25068	100272
PC	1	5000	5000
Total Frame Failure Rate			105272

[1] FIT is short for Failure unIT. One FIT is equal to one failure in 10^9 hours.

The failure rate for a frame is one failure every 9500 hours. For a machine consisting of *n* frames, the MTBF is 9500/*n* hours. In particular, a $1 million machine consisting of 10 frames would have an MTBF of 950 hours or about 40 days. The expected time to find a DES key using *n* frames is 35/*n* hours. Therefore, we expect one failure every 270 keys found regardless of the machine's size. This failure rate is much lower than is necessary to get useful key search results.

Appendix E: Support for 1-bit and 8-bit Cipher Feedback

The most popular modes of DES which are not handled by the key search chip design in Appendix A are the 1-bit and 8-bit cipher feedback (CFB) modes. A number of changes to the basic chip design would have to be made in order to support these modes. For the following discussion it will be useful to refer to Figure A.1.

From the external microcontroller's point of view, the main changes would be that it would have to specify which mode to use (ECB, 1-bit CFB, or 8-bit CFB), and for the CFB modes the microcontroller would have to specify the initialization vector (IV) value which immediately preceded the encryption of the known plaintext. We will assume that there are still 64 consecutive bits of known plaintext and ciphertext that are supplied to the key search chip. Within the chip, additional registers are needed to hold the mode and IV, and additional write logic is required in the external_interface.

The strategy for the CFB modes is to begin by taking each key and performing the encryption for the first bit or byte of plaintext. If there is no match with the corresponding bit or byte of ciphertext, then go on to the next key. If there is a match, then the key_counter must be stopped for one clock tick, the key from the bottom of the DES pipeline must be inserted into the top of the pipeline, and the next IV for CFB must be computed and fed into the plaintext input at the top of the DES pipeline. To keep track of the number of matches there have been for each key, we need an extra 6 flip-flops in each stage of the pipeline. Not until there has been a full 64 bits of matching with the ciphertext register will a key be declared found and stored in the output_key register.

We will require multiplexors at the inputs to the DES pipeline to choose between a new key and continuing with an old key. This adds too much delay to the first pipeline stage making it necessary to add an additional pipeline register at the beginning of the DES pipeline.

All of this additional circuitry enlarges the key search chip by 2000 gates. This increases the chip's cost from $10.50 to $12, which increases the cost of a key search frame to $108 240.

References

[1] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", *Journal of Cryptology*, vol. 4, no. 1, 1991, pp. 3-72.

[2] E. Biham and A. Shamir, "Differential Cryptanalysis of the full 16-round DES", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '92 Proceedings*, Springer-Verlag, to appear.

[3] K.W. Campbell and M.J. Wiener, "DES is not a Group", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '92 Proceedings*, Springer-Verlag, to appear.

[4] "Data Encryption Standard", National Bureau of Standards (U.S.), Federal Information Processing Standards Publication (FIPS PUB) 46, National Technical Information Service, Springfield VA, 1977.

[5] "DES Modes of Operation", National Bureau of Standards (U.S.), Federal Information Processing Standards Publication (FIPS PUB) 81, National Technical Information Service, Springfield VA, 1981.

[6] W. Diffie and M. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer*, vol. 10, no. 6, June 1977, pp. 74-84.

[7] H. Eberle, "A High-Speed DES Implementation for Network Applications", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '92 Proceedings*, Springer-Verlag, to appear.

[8] G. Garon and R. Outerbridge, "DES Watch: An examination of the Sufficiency of the Data Encryption Standard for Financial Institution Information Security in the 1990's", *Cryptologia*, vol. XV, no. 3, July 1991, pp. 177-193.

[9] M. Hellman, "DES will be totally insecure within 10 years", *IEEE Spectrum*, vol. 16, July 1979, pp. 32-39.

[10] F. Hoornaert, J. Goubert, and Y. Desmedt, "Efficient hardware implementation of the DES", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '84 Proceedings*, Springer-Verlag, pp. 147-173.

[11] B. Kaliski, R. Rivest, and A. Sherman, "Is the Data Encryption Standard a Group? (Results of Cycling Experiments on DES)", *Journal of Cryptology*, vol. 1, no. 1, 1988, pp. 3-36.

[12] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", *Lecture Notes in Computer Science: Advances in Cryptology - Eurocrypt '93 Proceedings*, Springer-Verlag, to appear.

[13] "MC68HC705C8 Technical Data", Motorola Inc., 1989.

[14] P.C. van Oorschot and M.J. Wiener, "A Known-Plaintext Attack on Two-Key Triple Encryption", *Lecture Notes in Computer Science: Advances in Cryptology - Eurocrypt '90 Proceedings*, Springer-Verlag, pp. 318-325.

[15] P. Wayner, "Content-Addressable Search Engines and DES-like Systems", *Lecture Notes in Computer Science: Advances in Cryptology - Crypto '92 Proceedings*, Springer-Verlag, to appear.