wjert, 2021, Vol. 7, Issue 4, 257-267. Original Article ISSN 2454-695X

World Journal of Engineering Research and Technology

WJERT

www.wjert.org SJIF Impact Factor: 5.924

IMPLEMENTATION OF A SAP-1 COMPUTER WITH DATA ENCRYPTION STANDARD (DES) CO-PROCESSOR

Jet Orville Chua, Emil John Lopez and Josef Lyle Solis, Dino Ligutan and Cesar Llorente*

De La Salle University, Manila, Philippines.

Article Received on 26/05/2021 **Article Revised on 16/06/2021** **Article Accepted on 06/07/2021**

***Corresponding Author**
Cesar Llorente
De La Salle University, Manila, Philippines.

ABSTRACT

This paper presents the creation of a SAP-1 Computer with Data Encryption Standard (DES) Co-processor. The original SAP-1 was modified in such a way that it will be easily interfaced with a DES processor. The first significant change is the increase with the size of the data bus to 64-bit since DES performs encryption and decryption with this size. However, some components in SAP-1 have retained their input and output to 8-bit. There is addition of application specific instructions to make the system efficient. One possible improvement might be to implement separate data and address buses instead of the current single bus design. Furthermore, the overall efficiency of the system can be better tested by performing a larger volume of encryptions/decryptions in quick succession. This will also expose several potential inefficiencies and bottlenecks in the program. The system can also be sped up by having parallel DES processors working in parallel. This will allow multiple blocks to be encrypted/decrypted at the same time.

KEYWORDS: SAP, DES, algorithm, encryption, decryption, VHDL

INTRODUCTION

As we traverse the digital world, data has become more and more significant given that most of the things and processes that are previously done offline can now be done digitally. Sensitive information such as video or audio conferences, online bank transactions, trading platforms, digital wallets, and things of the like are now becoming widespread. And to counteract unauthorized access to these sensitive data, and secure the information,

cryptography is very essential (Stallings, W. 2011). One of the most basic process in cryptography involves converting distinguishable text into something that is not and then back into its original form; this is done when data is to be transferred through an unsecured network (Vaudenay, S. 2006) Through the development of technology, there is an ongoing demand for a low-cost, high-performance, large-scale integration of ciphers for embedded computing and systems with similar applications. An example of this is the Data Encryption Standard algorithm. The DES algorithm is defined as a symmetric key algorithm and is most sought for its capability of providing security while being accessible at a low cost. DES has a limit of 56-bit keys which begs the question with regards to its level of security such that it can potentially be vulnerable to brute force attacks. However, such methods would at least take months for computer software to decipher (Kelly, S, 2006). Through the years, this encryption method has become innovated into the new encryption standard, AES (advanced encryption standard (Daemen, J. and Rijmen, V, 2013). Although DES may seem to be outdated as it is the predecessor of AES, it is still used in some information security applications due to its easy implementation and low cost.

The research conducted by Pandey, et al. (2016) expounded on the development of a DES algorithm with a specialized VLSI architecture as a framework. Their research has its architecture such as the process of encryption would take 19 clock cycles. On the other hand, the process of decryption is nearly identical to the encryption process meaning that it would also take 19 clock cycles to accomplish this. The key generation process is obtained from the main datapath giving more direct access to the encryption and decryption blocks; the necessary 16 round keys are provided into the specified block (encryption or decryption) during the first clock cycle. This supplements the DES algorithm as the last round key is used to indicate the start of the decryption process. The proponents have also implemented an S-Box consisting of 5 multiplexers; additionally, they have also conducted some modifications to allow the system to operate in a pipelined mode.

The project will focus on the development of a DES Crypto Co-processor with the use of the modified SAP architecture as the structural framework of its operations. With the variety of the instructions needed by the DES Co-processor, the developers would still be creating additional components and implementing new instructions. The researchers have the goal of implementing a DES Crypto Co-processor that works identical to the processes of the data encryption standard. Additionally, the developers aim to implement the DES algorithm such

that it can also work with 8-bit systems given that it is originally a 64-bit block cipher. Furthermore, the project to be developed has the goal of improving some aspects of the common DES algorithm implementation. As the algorithm is to be implemented with the SAP architecture, memory utilization is made to be more adept as compared to a standalone implementation. Moreover, it is also expected that there would be a significant improvement in terms of overall efficiency given that the common implementation would have all 16 iterations of the key generation cycle requiring for a fetch instruction which would essentially slow down the performance of the system; the project, however, would only require a single fetch instruction to perform the entire key generation process.

Figure 1 shows the architecture of the proposed SAP-1 computer with DES Co-Processor. It is immediately apparent that most of the components are directly derived from the improved SAP-1 architecture. The DES algorithm will be implemented as a separate unit because many of its internal operations are unique to the algorithm and are not usually implemented in general-purpose microprocessors. Several units are to be added to the top-level SAP architecture, namely, the DES Key, DES Processor, DES out, DES Text, Char Pointer, Block Pointer, and the Output Block Pointer.

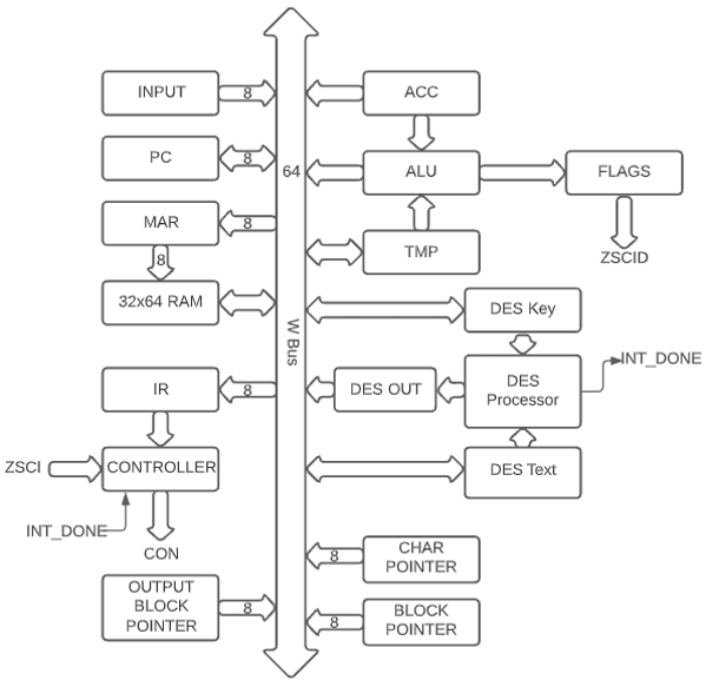

Figure 1: Proposed enhancement to SAP-1 Architecture with DES Co-Processor.

The DES key serves as a register that stores the key that will be utilized for the encryption/decryption process. The DES processor is responsible for running the actual

encryption and decryption process. DES out serves as a buffer that stores the output of the previous process, this was implemented as such because the output will still be updated and iterated, so a temporary storage is necessary before the final output is obtained by the system. The DES text block consists of the block currently being encrypted or decrypted by the processor. The char pointer and the block pointer serve as a guide as to where the next available register can be stored with a character or a text; given this, the block pointer increments by 8 after one text input. The output block pointer serves as a guide pertaining to where or which memory is available to have the final answer stored.

MATERIALS AND METHODS

DES Co-processor implementation is done by modeling the datapath and the controller in VHDL. Xilinx Vivado 2019-2 is used to model and simulate the system. The Instruction Set Architecture of SAP-1 is enhanced by adding instructions specific to the DES operation. Table 1 lists the instruction added to the ISA of SAP-1. After defining the additional instruction set, DES co-processor unit is integrated to the original SAP-1 architecture. As for the initial testing that will be conducted, it is imperative to first make sure that the system resulted in the correct output. This will be done using a C program of a DES algorithm implementation that is found online. So, with this, what will happen is to input a set of parameters to encode or decode into the developed system. This same set of parameters will serve as an input in the C program to verify whether that system developed produced the appropriate output (https://www.techiedelight.com/des-implementation-c/). Now, this can be placed inside a table to have a side-by-side comparison of both results to ensure the correctness of the result.

An SAP-1 assembly language program is written and run in the SAP-1 computer to verify the operation of the co-processor. The timing waveform generated by the system during simulation is the observed and recorded and analyzed to determine correct operation of the DES co-processor. To put some benchmark onto the simulation of the system, the calculation of CPI was performed. Before going into the actual performance, the theoretical CPI was calculated first. This was done by tallying all the instructions and their respective cycles found in the assembly code found in the next section. After this, the actual CPI is computed and is subdivided into 5 parts which includes initialization, block 1, block 2, block 3, and last null check. For the initialization part, the first 7 instructions were checked and with respect to its total cycles. For all the three blocks, they all have the same CPI and to calculate it the

proponents will track the time when the program counter reaches 36 from 12. Lastly, the last null check was easier since the calculation was from the end of block 3 until the part where checking of the termination character was satisfied.

One of the objectives of this study was to develop a specialized SAP processor for performing the DES algorithm. Therefore, it expected to be faster than a software-based implementation. For comparison, the fastest software implementation by (Biham, E, 1997) will be compared to the system. This program had an encryption rate of about 46 Mbps. Since this study's program was tested on a 300MHz Alpha processor machine, it is reasonable that the SAP processor implementation's encryption rate be computed under the assumption that it can be reliably run with a 300MHz clock. The encryption rates will be descriptively compared to assess the performance of the machine.

DES Co-processor design and implementation

Figure 2 shows the finite state machine design for the DES co-processor. It starts when the SAP architecture runs a DESEnc or DESDec instruction. When this opcode is executed, the SAP processor sends a signal to the DES unit *en_des* to start up the encryption/decryption process.

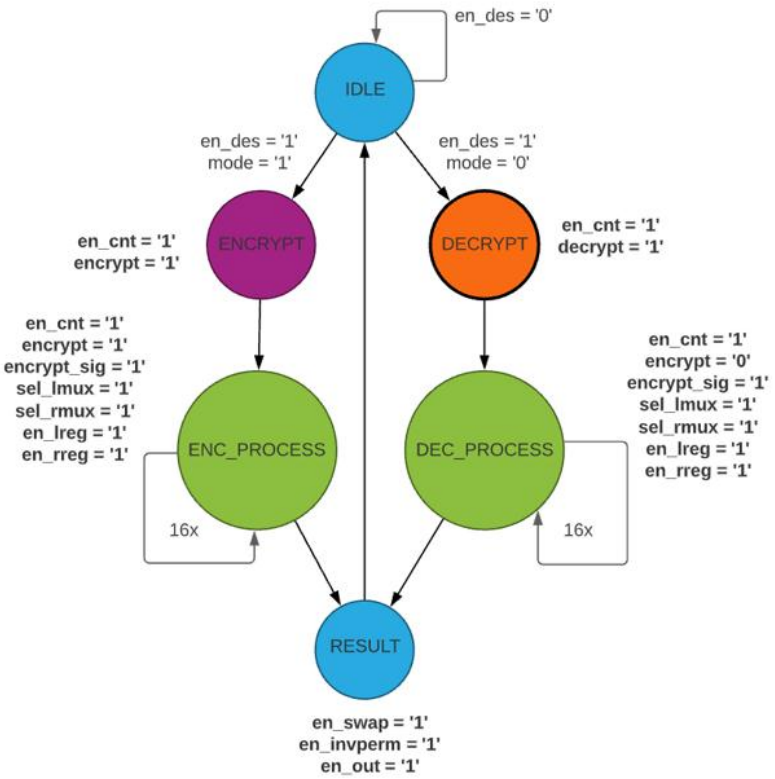

Figure 2: State diagram defining the DES Finite State Machine.

It will also send the *mode* signal to indicate whether the process should do encryption or decryption. After it has determined which operation it should do, it will activate every datapath component (except the swap and inverse permutation modules) to begin the computation. After 16 cycles, it tells the SAP processor that the encryption/decryption is complete. The DES processor has several components internally. Components without control lines are non-sequential. They are:

- **Initial Permutation** - Shuffles the original bits of the input text in a specific configuration.
- **Multiplexers** (CONTROLLED by *sel_lmux* and *sel_rmux*)- Allows the circuit to choose between initial inputs (from initial permutation) or the results of the previous round
- **DES Registers** *(*CONTROLLED by *en_lreg* and *en_rreg*) - Stores the results of the previous round (left for bits 63-32 and right for bits 31-0)
- **S-Box** - Nonlinear transformation of the bits (explained in the previous section)
- **Key Generation** ***(***controlled by an internal 4-bit counter) - Unit responsible for generating all 16 keys from the main key
- **Data swapping** (controlled by *en_swap*) - Switch the contents of the L and R registers
- **Inverse permutation** (controlled by *en_inv_swap*)- Shuffles the bits from data swapping in the reverse order of the initial permutation.

RESULTS AND DISCUSSION

The co-processor was modeled and simulated in VHDL using the Vivado tools. Figure 3 shows the synthesized hardware of the DES co-processor.

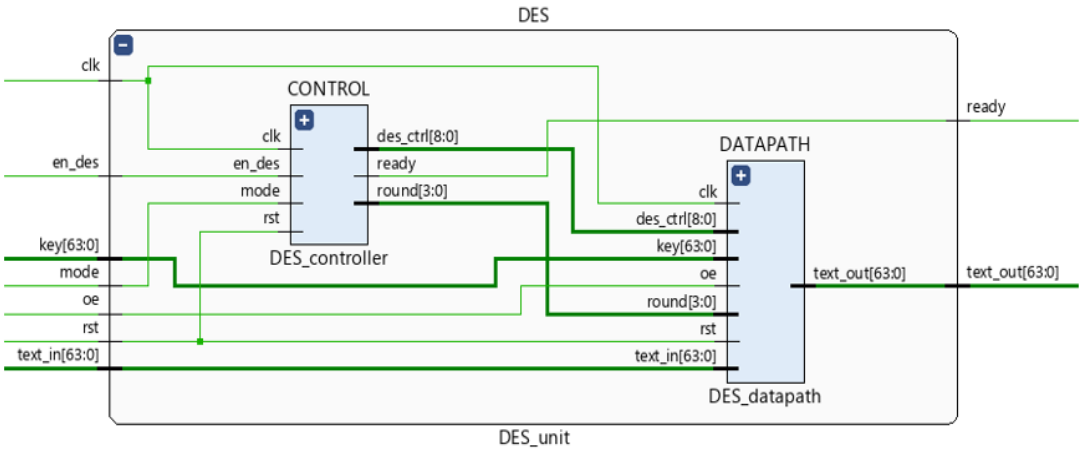

Figure 3: Synthesized DES controller and its datapath.

SAP-1 Assembly Language test program written for SAP-1 architecture is listed in Table 2. This application-specific SAP-1 processor is designed to encrypt/decrypt an arbitrary number of bytes using the DES algorithm. The system will be responsible for grouping every 64 bits as a single block, sending it to the DES co-processor and waiting for the next bytes to arrive during the encryption/decryption of the current block. This naturally means that there needs to be a format for sending encryption/decryption requests to the machine. For this purpose, the format shown in the figure above will be used. The mode byte tells the machine whether an encryption or decryption should be done. The null character at the end signals that the message has ended.

Computer Performance testing

Table 3 shows the comparison of the expected output computed using a C program (provided in the Glossary section) and the outputs for each block. It successfully encrypted all the blocks even though they were sent in quick succession. Using the key 1122112211221122, the machine encrypted the string: 1817161514131211100f0e0d0c0b0a0908070604030201 as 19037f74ca752bb4cdb7b865437c48101a451abcf5881415.

The computation for the theoretical and actual Clock Per Instruction is listed in Table 4 and Table 5, respectively. These values are taken from the simulation run as reported by the simulation tool. Moreover, the tables put into detail the instruction count with each unique clock cycle ranging from 4 to 9. As can be observed, the actual CPI computation had a significantly larger instruction count and total cycle as compared to the theoretical computation, this is due to having the system loop throughout the same instructions to complete the process while the other simply considers a linear flow of the assembly instructions.

Calculating for the time it will take for the DES algorithm to perform encryption using the C language implementation results to an encryption rate of 46 Mbps. The algorithm runs on a computer with a clock rate of 300 MHz. In the proposed system, encryption rate is 52.85 Mbps with the system clocked at the same clock rate. This means that the proposed system is 14.89% faster than the software implementation by (Daemen, J. and Rijmen, V,2013). This is shown in Table 6.

Table 1. Enhanced Instruction Set of SAP-1 that include DES specific instructions.

Mnemonic	Opcode	Description
LDA *address*	10h	Loads the data of a specific address from the memory to the accumulator
STAByte *address*	17h	Stores the least significant byte from the accumulator to the specified address in memory (byte-addressing)
STA CHP	19h	Stores the least significant byte from the accumulator to the current address in the character pointer register and increments the character pointer
MVI A	20h	Loads immediate to accumulator
MVI B	21h	Loads immediate to B register
ANI	38h	Utilizes AND operation of 8-bit immediate data to the accumulator's content
INC B	49h	Increments the data of B register by 1
JMP	D0h	Jump to a specific address location in the memory
JNZ	D7h	Jump to a specific address location of the memory if the zero flag is 1. Otherwise, it doesn't jump
DESEnc	F0h	Encrypts the plain text in the TEXT register using the key from the KEY register
DESDec	F1h	Decrypts the cipher text in the TEXT register using the key from the KEY register
MOV key, A	F2h	Moves the value in the accumulator to the key register
LDBLK *address*	F3h	Loads the 64-bit word (in the address specified by the block pointer) into the text register and increments the pointer by 8 (since a block is 8 bytes long)
STDESOut	F4h	Stores the output of the DES processor to the address in the OUTPUT BLOCK register
HLT	FFh	Stop all processes

Table 2. Assembly language program listing for SAP-1 architecture to test the operation of the DES co-processor.

Address	Instruction	Comment	Content (Hex)
0	LDA		10
1	$key_address		38
2	MOV key, A	Move to key register	F2

3	IN		EE
4	ANI		38
5	0x01		01
6	JNZ		D7
7	**encrypt**		0C
8	MVI A		20
9	F1	DESDec	F1
10	JMP		D0
11	14		0E
12	MVI A	**encrypt**	20
13	F0	DESEnc	F0
14	STA byte		17
15	[address1]		1F
16	MVI B		21
17	0x00		00
18	IN		EE
19	ANI	zero check	38
20	0xFF		FF
21	JZ		D6
22	**operation**		1E
23	STA CHP		19
24	INC B		49
25	MOV A, B		24
26	ANI 0x07		38
27	0x07		07
28	JNZ		D7
29	18		12
30	LDBLK	**operation**	F3
31	[address 1]	*encrypt/decrypt*	00
32	MOV A,B		24
33	ANI	zero check	38
34	0xFF		FF
35	F4		F4
36	JNZ		D7

37	16		10
38	HLT		FF

Table 3: Comparison of the expected output computed output using a C program to estalish the accuracy of the proposed system.

	Block	**Expected Result**	**Output**
1	1817161514131211	19037f74ca752bb4	19037f74ca752bb4
2	100f0e0d0c0b0a09	cdb7b865437c4810	cdb7b865437c4810
3	08070604030201	1a451abcf5881415	1a451abcf5881415
	Accuracy	100%	

Table 4: Theoretical CPI Computation.

Cycles	**Instruction Count**	**Total Cycle**
1	5	20
2	9	45
3	3	18
4	6	42
Total	**23**	**125**
	CPI	5.434782609

Table 5: Actual CPI Computation.

Cycles	**Instruction Count**	**Total Cycle**
1	5	20
2	9	45
3	3	18
4	6	42
Total	**23**	**125**
	CPI	5.434782609

Table 6: Actual CPI Computation.

	Encryption Rate (300MHz)
Software Implementation [5]	46 Mbps
SAP with DES implementation	52.85 Mbps
%Difference	14.89%

CONCLUSIONS

In this study, a hardware implementation of the DES algorithm was caried out. Simulation results indicate that the co-processor was successfully synthesized. The goal of the study to develop a DES crypto co-processor with the use of the modified SAP architecture serving as the structural framework for its operations was successfully carried out. Given the incompatibility of the 8-bit data bus structure of the SAP-1 architecture, and the 64-bit structure of the DES co-processor, the proponents were able to successfully interface the co-processor to the 8-bit processor. Having the goal of having the algorithm in a separate processor meant that there would be a problem upon integration of both since the original SAP was only 8-bits. This means that steps must be taken to load each byte unto the processor if we do it as is. This is a potential bottleneck for the system. One of the more straightforward ways to do this is using 64-bit data and addresses instead.

REFERENCES

1. Stallings, W. Cryptography and Network Security Principles and Practice, 5th ed. Prentice Hall, 2011.
2. Vaudenay, S. A Classical Introduction to Cryptography: Applications for Communications Security. Springer Science & Business Media.
3. Kelly, S. (2006), Security implications of using the data encryption standard (DES). [Online]. https://tools.ietf.org/html/rfc4772
4. Daemen, J. and Rijmen, V. (2013), The design of Rijndael: AES-the advanced encryption standard. New York, USA" Springer Science & Business Media.
5. Biham, E., "A fast new DES implementation in software," Fast Software Encryption, 1997; 260–272. doi: 10.1007/bfb0052352.