Design of an ASIC CORE for data encryption and decryption Using the NIST Advanced Encryption Standard

Víctor Hugo Cordero Calle, Carlos Silva Cárdenas
Email cordero.vh@pucp.edu.pe, csilva@pucp.edu.pe
Microelectronics Group – Pontificia Universidad Católica del Perú
Telf: (511) 4602870 anexo 304 Ext. 209.

ABSTRACT

The 'National Institute of Standard and Technologies' (NIST) defined on the 'Federal Information Processing Standards' (FIPS) Publication #197 the current 'Advanced Encryption Standard' (AES) to be used for commercial applications. This work focuses on the 'Application Specific Integrated Circuit' (ASIC) implementation, starting in a fully unrolled Verilog description, optimized up to gate level of its critical modules in order to obtain maximum performance while keeping the architecture usable for both of the processes (encryption and decryption processes). This work also covers a full custom design analysis of the encrypt / decrypt module in order to get the expected performance in the 0.12 um CMOS process technology.

RESUMEN

El Instituto Nacional de Estándares y Tecnología de los EE.UU. (NIST) definió en la publicación federal de estándares de manejo de información #197 (FIPS 197) el actual Estándar de encriptación Avanzado (AES) para ser usado en aplicaciones comerciales. El presente trabajo se centra en la implementación de este estándar en un diseño ASIC (Circuito Integrado de Aplicación Especifica), empezando con la descripción desenvuelta del algoritmo en Verilog, y optimizada en código a nivel de puestas lógicas y en paralelismo para obtener así un máximo rendimiento mientras se mantiene un uso bajo de puertas dentro de una arquitectura utilizable para realizar el proceso de encriptación y desencriptación en un solo modulo. En este documento también se añade un análisis de implementación en full custom para el proceso mas critico (el modulo de encriptación y desencriptación) para obtener su performance estimada en tecnología de proceso CMOS de 0.12 um.

Design of an ASIC CORE for data encryption and decryption Using the NIST Advanced Encryption Standard

Victor Hugo Cordero Calle, Carlos Silva Cárdenas
Email cordero.vh@pucp.edu.pe, csilva@pucp.edu.pe
Microelectronics Group – Pontificia Universidad Católica del Perú
Telf: (511) 4370828 anexo 304 Ext. 209.

ABSTRACT

The 'National Institute of Standard and Technologies' (NIST) defined on the 'Federal Information Processing Standards' (FIPS) Publication #197 the current 'Advanced Encryption Standard' (AES) to be used for commercial applications. This work focuses on the 'Application Specific Integrated Circuit' (ASIC) implementation, starting in a fully unrolled Verilog description, optimized up to gate level of its critical modules in order to obtain maximum performance while keeping the architecture usable for both of the processes (encryption and decryption processes). This work also covers a full custom design analysis of the encrypt / decrypt module in order to get the expected performance in the 0.12 um CMOS process technology.

1. INTRODUCTION

On recent years there has been an increasing interest on the encryption technologies, as a result several encryption algorithms have been developed to satisfy client needs, but the proliferation of them made the National Institute of Technology ask for proposals for the Advanced Encryption Standard. The winner of them was the Rijndael algorithm due to its high performance, security, and minimum area usage when implemented in an ASIC or in a FPGA.
Up to this moment, most of the high performance AES implementations have been developed for FPGA. As a result, there have been few papers written where an ASIC implementation of the AES is discussed.
The purpose of this work is to discuss an architecture specifically thought for using the best of the capabilities that the ASIC design allows, where a minimum gate count is highly appreciated from the very beginning at the HDL design. The parallelism and pipelining of the modules have been taken into account too, in order to reach the highest performance possible.
The HDL design has been written entirely on Verilog, using as a compiler and simulator the Synapticad Verilogger[12] where the entire functionality of this architecture has been tested and simulated; the appropriate order of the outputs and signals from the different modules have been tested using this tool too.
For the precise timing, given in the last part of this paper, each of the modules have been designed in accordance to the Verilog description in a full custom ASIC tool (Microwind [11]) setting it for the 0.12um CMOS process technology.
This ASIC core can be used into broadband, wireless or multimedia systems. The application of an ASIC AES core could be Video phones, PDA, point-to-point wireless, satellite communications, surveillance systems, network appliances, virtual private networks (VPN), voice services, etc, and everywhere an embedded encryption procedure could prove to be useful.
The AES algorithm is fully explained in the FIPS 197 [1], the mathematics about Galois Field (GF 2^8) is explained also in [1], therefore, its lecture is a requirement in order to understand the architecture. Since we have to be brief in this paper, no explanations about how the standard work will be made here. Further information about the AES algorithm can be found in [2], [7] and [9].
This work will focus on the fastest possible implementation of this algorithm for an ASIC CORE keeping the gate count reasonable. The following core is focused on the 128-bit-key encryption, but the same architecture can be used for the 192-bit encryption.
There exists four security schemes for data encryption [4] which are cipher block chaining (CBC) mode, cipher feedback (CFB), output feedback (OFB) and the straight forward electronic code book (ECB). In the first three, the output data has to be feedback for the next encryption, therefore, there it wont be possible to use any pipelining (section 2.11). This AES Core will not use round pipelining, as a result, it could be modified to work in the CBC, CFB, and OFB modes if required.

2. ARCHITECTURE

On the following sub-sections it will be explained in details this high performance architecture, which has been written in Verilog for a soft core solution (where is up to the synthesis tools, Cadence for example, to do the transformations into CMOS transistors)
Section 3 shows the full custom implementation of the critical module (the encrypt/decrypt module), in order to obtain a mixed solution (semi-custom design). Since there exists a layout for the full custom implementation (based on the verilog thought architecture) performance results will also be presented.
The possible bit-through-output improvements to this architecture will be highlighted on section 2.11, which are basically of pipelining,

2.1. Encrypt Decrypt Core

The selection of the system current work state is controlled through the following lines:
<Data/Key> (t selects whether the input is Data or the Cipher Key
<WR> Write line, it has to be positive edge activated after the data or cipher key are on the input bus;
<RD> Read Line, it is used to retrieve a 32-bit word from the core.
<Encrypt/Decrypt> Chooses the operation mode of the chip, it can be set to encrypt or to decrypt.
<Clr> This line is used if it is desired a full reset of the current data being encrypted or decrypted.
<Clk> Clock is the master clock for the whole AES CORE.
<KE_ready> This line indicates to the external hardware that the key expansion is complete.
<Data_ready> This line indicates that the encryption or decryption of a 128-bit work have been completed.

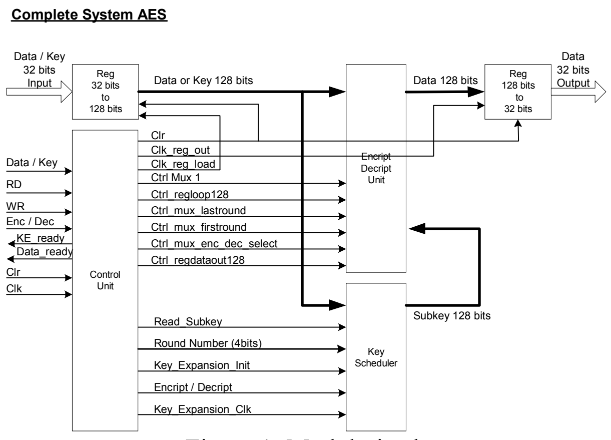


Figure 1: Modularized Architecture of the AES CORE

As shown in (fig 1), the architecture consists of four major units, which are:

Control Unit It handles the main state machine for the AES core, this unit controls the internal lines of the other units in order to complete the encryption, decryption or key expansion processes
Encrypt / Decrypt Unit This unit does the encrypt and decrypt rounds in the same module, utilizing to its best its internal multiplexers and registers.
Key Scheduler This unit will receive the cipher key at the beginning of the encrypt / decrypt process, the key is expanded only once and the resultant sub-keys are stored in a 11x128 bits memory. After the expansion, this module behaves as a memory, providing the 128-bit sub-key required for each of the rounds of the encrypt or the decrypt processes.
Input / Output registries The input registry is controlled by the rising edge of the WR line, after 4 edges, a 128-bit word is passed from the register to the internal 128-bit bus of the AES CORE. The output registry is controlled by the RD line. Right after a complete encryption or decryption process is finished the full 128-bit word will be released to the output bus in 32-bit words synchronized by four RD rising edges.

It is important to highlight that these independent set of input/output registers by sub-words of 32 bits wont have a penalty in the overall performance of the core. The loading and unloading of data from the external bus is done in parallel to the encryption / decryption process (the pipeline is formed by the input registry, the output registry and the enc/dec module), making the critical time, the time it takes to do the full 10 rounds in the Enc / Dec module. This is shown in (table 1).

T0	T1	T2	T3
Load 32->128			
	Enc/Dec Process		
		Unload 128->32	
	Load 32->128		
		Enc/Dec Process	
		Load 32->128	
			Unload 128->32
			Enc/Dec Process
			Load 32->128

Table 1: Pipelining in the AES CORE

To summarize the operation of the AES CORE (fig 1), it starts when the 128-cipher key is expanded in the Key scheduler and stored in its memory. Then, the core is ready to receive any amount of 128 bits blocks of data (in groups of 32 bits) to encrypt or decrypt. In order to write new data, the data line should be set, then, after each edge of WR, 32 bits will be loaded in the input registry. When a block of 128 is ready, the encryption process begins. The Control Unit will set or clear the appropriate lines for the 10 rounds process in the encrypt module; the required sub-key for each round, will be given by the Key Scheduler.
Right after the 10th round process inside the enc/dec unit is finished, the 128-bit crypted / decrypted word is handled

to the Output Registry (128 -> 32 bits). By this time, the enc/dec unit is ready to process another 128-bit word.

2.2. **Encrypt / Decrypt Module**

The encrypt/decrypt module (fig. 2) is the critical module of the system, because the through-output of the CORE relies on how fast this block can handle the data. Therefore, every component inside is combinatorial and capable to process a 128-bit word. There is only one registry in the loop, which is required to hold the data for each round process. One important feature here is that the Decrypt process is arranged in a way that it can use the very same 128-bit internal buses, multiplexers, registries and add-sub-key (128-bit xor stage) modules. Doing so, significant silicon area is saved, in comparison of a scheme with separate encrypt and decrypt units. This is possible because the equivalent inverse cipher[1] was used, so that the same arithmetic structure and sequence is used for the encryption and the decryption processes. The extra cost is the addition of an inverse-mix-column block in the key-scheduler module. The multiplexers chosen here are 128x2, which have only a 3-gate delay per couple of lines to be multiplexed.

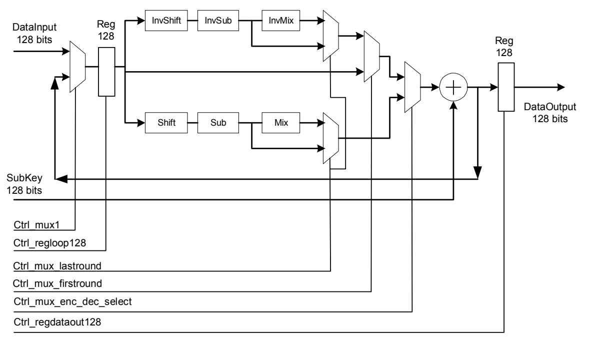


Figure 2: Encrypt / Decrypt Module

A couple of 128-bit buses are taken right after the SubBytes and InverseSubBytes modules, due to the last round process, which doesn't require the MixColumn nor the InverseMixColumn. Another 128-bit bus is taken right after the 128-bit loop register, because it is required for the first round process in the AES.

With this architecture, it is only required one of each module (InvShift, InvSub, InvMix, Shift, Sub, MixColum and Addroundkey) for the encrypt/decrypt processes. The penalty is just the delay and the area of the muxes, that is just of 3 gates delay per mux, significantly less costly in area than to implement any of the modules again (specially the Look-up-Tables)

2.3. **Parallel process in the encryption**

In this architecture, all of the processes are performed with an internal bus of 128-bit. The ShiftRow, SubBytes and MixColumn processes shown in (fig. 2) as blocks are detailed in (fig. 3).

The Input to the process is a 128-bit word, the matrix array proposed in the AES standard is unrolled as a linear vector of 8-bit elements, with all of the bytes being processed in parallel.

The first of the operations (ShiftRow) is realized by the rearrangement of the input bus (In a CMOS implementation there won't be transistors used, in consequence its delay will be almost zero).

The second operation is the SubBytes operation, which is a non-linear byte substitution, the optimal way to do this transformation is to pre-calculate the results for each input byte, and store the results of this transformation in a fast hard-wired Look up Table. This 8-bit input, 8-bit output table is named SBOX.

The third process is the Mix Column, which has been optimized for minimum gate delay, and optimums cmos implementation (this will be detailed in section 2.5).

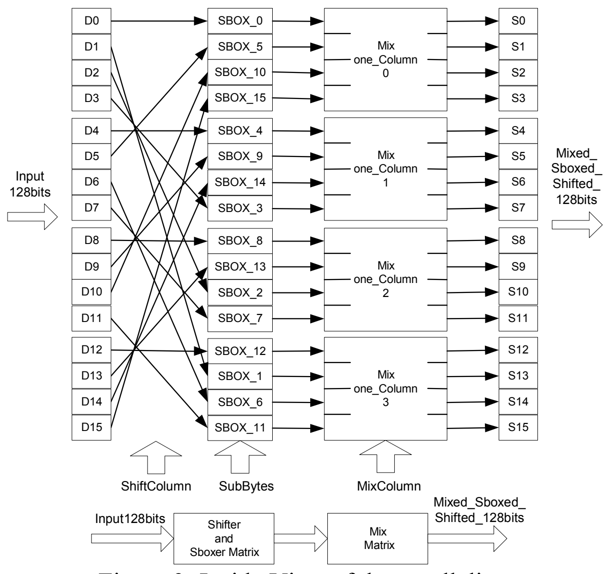


Figure 3: Inside View of the parallelism in the encrypt process

2.4. **SubBytes Look-up-Table (S-BOX)**

The SubByte is a hard-wired Look-up-Table that has been written in Verilog as a decoder of 256 positions pointing to 8-bit elements.

There are 16 S-BOX tables in the encrypt module. These blocks occupy most of the area in the CORE. Nevertheless, this is the best way to get the fastest encryption / decryption possible speed. The other solution would have been to perform the slow complex matrix transformation.

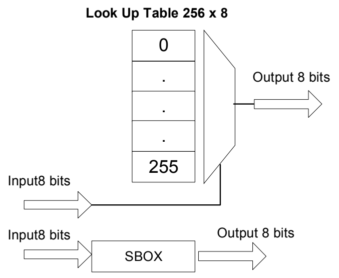


Figure 4: Sub Bytes module (S-BOX) For the encrypt process.

2.5. **Mix Column Module**

The mix column process (fig. 5) has been carefully developed to have the minimum possible delay. The full mix column process has just the delay of one X2 galois field (GF 2^8) multiplication (the math of Galois Field GF 2^8 is explained in [1]), a two input xor, and a four input xor which can be designed as a complex CMOS function of multiple inputs (instead of the usual AND, OR array implementation). The mix column gate delays are explained in section 3.2 .

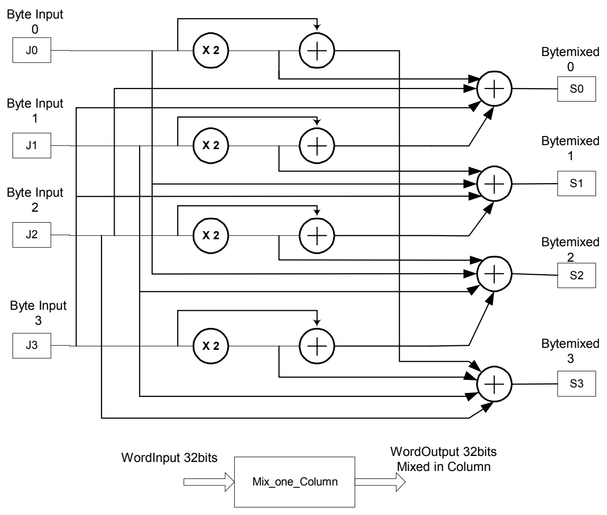


Figure 5: Mix-Column module For the encryption Process

2.6. **Multiplier per 2 in Galois Field 2^8**

The multiplier per two in the Galois field with module 'B00011011 (fig. 6) is the distinctive operation in the AES [1], all of the multiplications can be arranged to be a group of continuous X2 multiplications. On this design, the X2 delay is equal to a XOR gate delay.

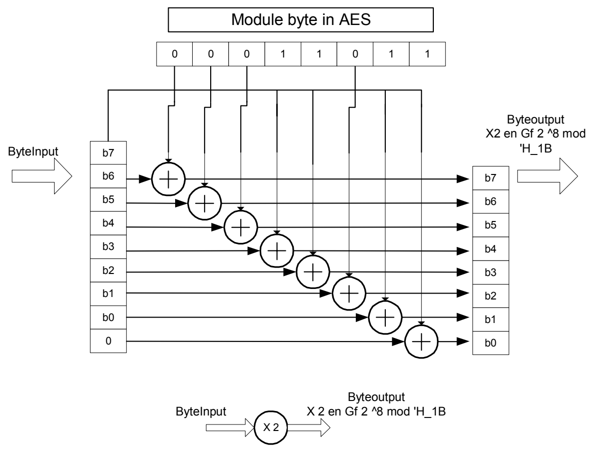


Figure 6: Multiplier per 2 module 'B0001_1011 In the Galois Field 2^8

2.7. **Parallel process in the decryption**

The equivalent unrolled version of the inverse cipher (fig. 7) does the decrypt process very similar to the encrypt process. As with the encrypt process the hard-wired Inverse shift stage so it wont give us significant delays, next, 16 Inverse Sboxes receive the sixteen 8-bit words, finally, four inverse column modules receive four 32-bit words.

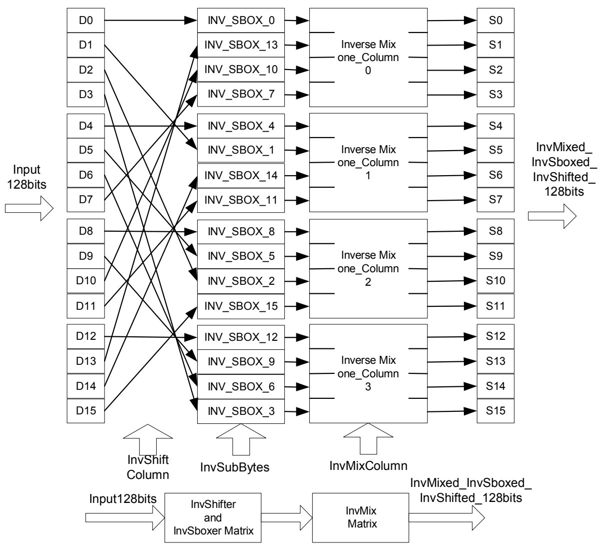


Figure 7: Inside View of the parallelism in the decrypt process

2.8. **Inverse Sbox for the decrypt process**

The AES defines a inverse sub-byte transformation which can be rearranged as a look-up-table (named inverse-SBOX in (fig.8)) with the exact same considerations as the SBOX but with different coefficients stored inside.

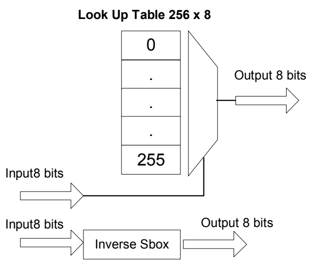


Figure 8: Inverse Sub Bytes module (Inv-S-BOX)
For the decrypt process.

2.9. **Inverse Mix Column**

The main difference of the decrypt process from the encrypt process is the inverse mix column. The inverse mix column (fig. 9) will have a longer path to cover, because the process requires multiplications with higher coefficients than the mix column process. To minimize this effect, and to reduce the overall gate count, a multiproduct module was developed, this way, for the 16 galois field products required in the invMixColumn, only 4 of this kind of multipliers are utilized.

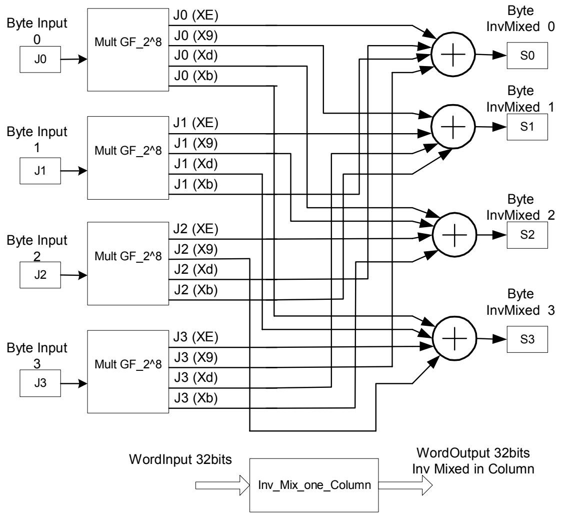


Figure 9: Inverse-Mix-Column module
For the decryption Process

2.10. **Multiproduct Multiplier in Galois Field 2^8**

This module (fig. 10) was optimized to have the closest performance to its equivalent of the encrypt process, the product operation has been rearranged to be three X2 product operation, that recursively use its internal results in order to get all of the multiplications required for the inv mix column. By doing so, the only extra delay added was of just two X2 delays, which translates to an additional two xor gates delay. This way the decryption process will be almost as fast as the encryption process.

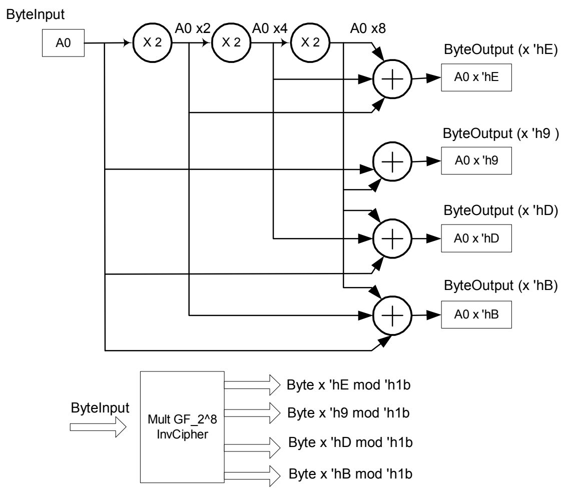


Figure 10: Multipurpose multiplier module
in the Galois Field 2^8 for the decrypt process

In sum, by focusing only on the particulars GF 2^8 products required in the AES process for the Inv-mix-column, a lower gate delay than the one in [10] was reached. Our Inv-mix-column module's delay is just of 5 xor gate delay.

2.11 **Some Improvements**

The possible improvements [3] on round-based encryption algorithms are inner-round pipelining (fig 11_a) and outer-round pipelining (fig 11_b).

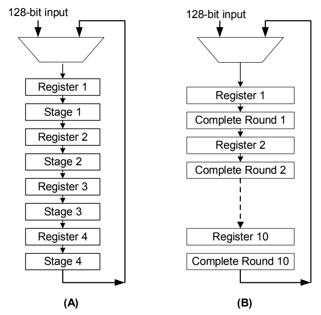


Figure 11: Pipelining Schemes

In the inner round pipelining (fig11-a) the purpose is to divide the round process in separate stages, with a register between each stage. This way if we could be able to separate the one round in four stages of approximately the same delay, we would be able to encrypt four 128-bit data block in a 10+3 rounds because of the pipelining. The main trouble would be that it would require an architecture where each of the processes were performed exactly one after the other, so, it would be necessary to implement extra modules to perform the first and the last round which performs differently than the 2nd through 9th round (16 S-Box and 16 inv-S-Box more). Inner pipelining would improve bit-through-output performance by at least a factor of four.

The other possible improvement would be to serialize multiple complete round (outer round pipelining), putting register between them. This way if we had the ten complete rounds serialized the performance would increase by ten. The problem here would be the excessive area used and the massive amount of lines the control unit would have to handle.

Note that the pipelined version will only work for the Electronic Code Book security scheme, were an input 128-bit word will always transform in the same 128-bit ciphered word.

The USA National Security Agency implemented de AES in ASIC with the outer round pipelining architecture, their performance results can be read in [6]. Their area usage was huge (471,996,329 um; 7,130,697 transistors for their 5337.78 mbps version).

3. RESULTS

All of the architecture modules have been written in verilog, compiled and simulated thoroughly in Synapticad Verilogger. This tool helped to check if the processes were being performed in the appropriate order and to check if whether the modules passed their data on time.

The results presented here will come from the full custom design tool Insa Toulouse Microwind v2.5. On this tool was analyzed the critical path of each separate module of the encrypt/decrypt process (they just differ on the mix-column process). Because of the pipelining, the critical module that will determine the theoretical performance of the AES CORE will be the encrypt/decrypt module. The key expansion was not analyzed because this process is done only once in the whole operation of the AES CORE, After the key is expanded, is stored in a memory and read directly from there, which has about the same delay as a SBOX-table look up. The performance of the control unit wasn't analyzed either, that is because, this unit is composed of a large state machine which clearly works faster than the enc/dec module.

All of the full custom modules analyses were designed for a CMOS 0.12 um technology.

3.1. Look up table delay

The full-custom look up table(fig. 13) is implemented as two decoders of 4 inputs, one of them selects the column and the other the row of the table. Once selected a cell in the table, eight couples of C-switches releases a hard-wired 8-bit word. This implementation gives a delay (fig 12) of 0.160 ns with a minimum period of the input word of 0.5ns. This can be seen in figure 11, where the fetched word passes from ‘h00 to ‘h11, giving the appropriate output accordingly to AES[1] (‘h63 and ‘h82).

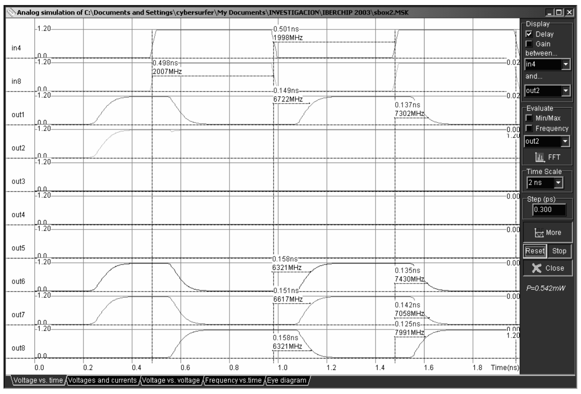
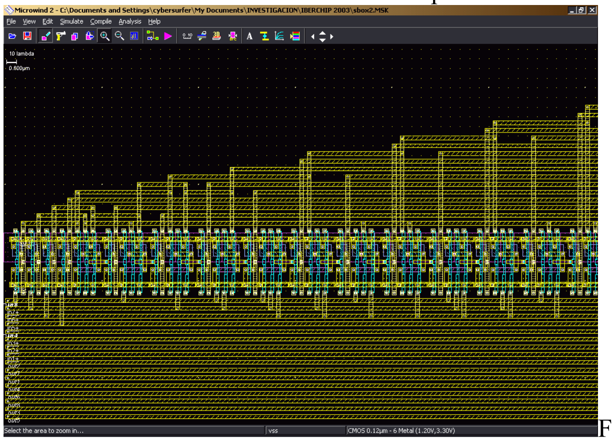


Figure 12: Look up table time diagram, passing From ‘H00 to ‘H11 in its input.


Microwind 2 - C:\Documents and Settings\cybersurfer\My Documents\INVESTIGACION\IBERCHIP 2003\sbox2.MSK
File
View
Edit
Simulate
Compile
Analysis
Help
10 lambda
0.600µm
Select the area to zoom in...
vss
CMOS 0.12µm - 6 Metal (1.20V,3.30V)

F
igure 13: Part of the look up table cmos layout in Microwind

3.2. Mix Column operation delay

As shown in (fig 5), the mix column critical path will be the delay of a X2 operation followed by a xor (each input of 8 bits xored bit by bit) and a four input xor (each input of 8 bits). In the worst case scenario for the 4 input xor, its delay (fig.14) is of 0.125ns, and for the 2 input xor is of 0.50 ns. When all of the blocks were put together, the delay grew up to 0.520ns, because a automatic synthesis (fig.15) tool was used in order to be able to manage all of the data lines if a mixcolumn module. Because of the horizontal expansion of the cmos transistors of this tool the delay incremented higher than the adition of the simple modules' delay.

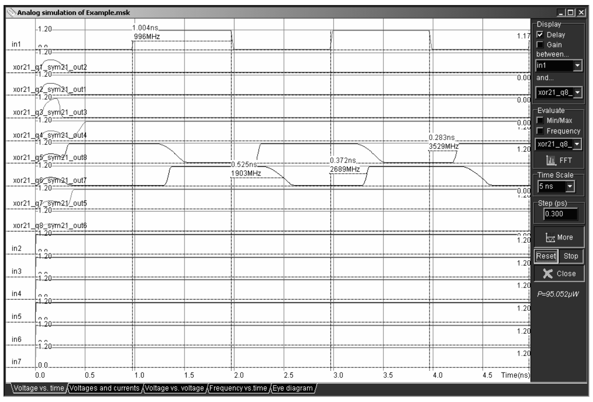


Figure 14: Maximum delay for the mix column

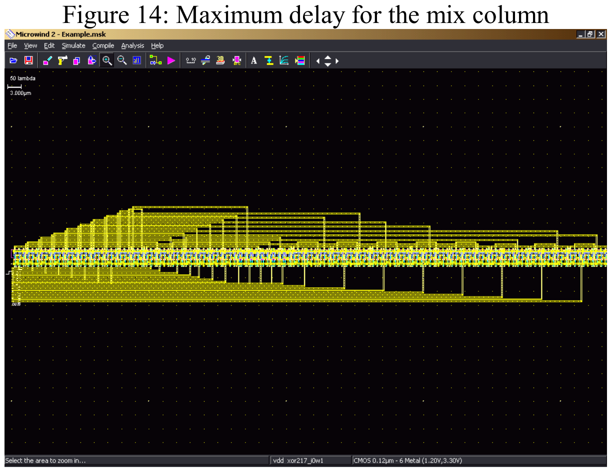


Figure 15: Cmos layout of the mix Column by Automatic cmos synthesis

3.3. Multiplexers 128x2 delay

The multiplexers chosen for this AES CORE, were 128x2 muxes, this is because its minimum gate delay in comparison to use a 128x8 mux. By arranging a set of 128x2 the extra expenditure will be only of metal path.

A multiplexer 128x2 can be seen as a set of 128 parallel couples of c-switches controlled by the same selector (fig.17) In consequence it is only required to analyze one of this. The delay (fig.16) of this kind of mux is only 0.017ns.

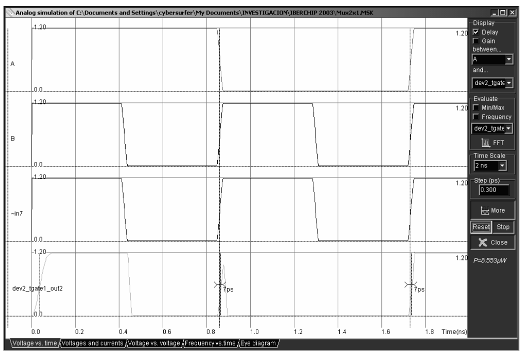


Figure 16: Maximum delay of the multiplexer

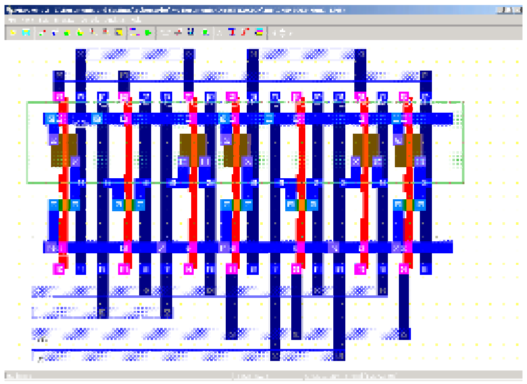


Figure 17: Layout of the basic build cell of a mux by c-switches

3.4. Register delay

Similarly to the multiplexer the 128 bit register is composed of 128 parallel d-latches (fig.19) with the same clock and with the same delay (fig.18). The computed delay is of 0.06ns

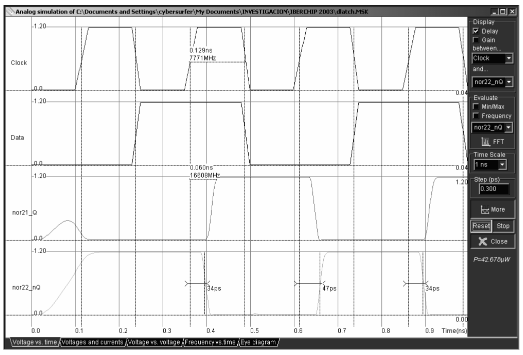


Figure 18: Delay of the fully customized cmos latch

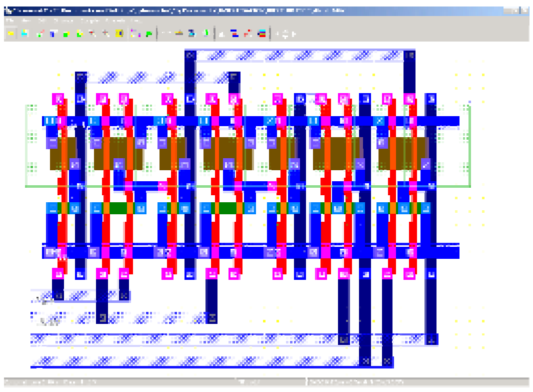


Figure 19: Cmos layout of the basic Latch for the 128-bit registry

3.5. Conclusions

From the numbers thrown above, the theoretical maximum speed for this architecture comes from the addition of the propagation delays, multiplied by a factor due to extra metal paths to connect the modules (1.5) and another to ensure that the data is stable enough (2).

Total numerical delay for a simple round process: 1.291ns
Max through output considering 10 rounds for 128-cipher key and 0.12 um cmos technology:

$$\frac{1}{1.291ns*10rounds*1.5*2}*128bits=3304megabits/s$$

These results are in accordance with the through output reached by the microelectronics group of the University of California [4] & [4_A], who reached a data transfer rate of 2290megabits/s on a 0.18 um cmos technology integrated circuit using 173k gates.
More comparative results can be found in [5] where an un-pipelined implementation on 0.35um CMOS was made, their through-output was of 1.95gbps using 613k gates.

4. FUTURE WORK

The present project is the thesis work advances for obtaining the electronic degree title of the author, the next stage of this work will the to make the whole encrypt / decrypt module in a full custom layout. As it was shown here, this critical process has to be carefully regarded.
The other modules like the control unit, and the key scheduler will be synthesized in the Cadence software for its cmos layout, the actual code written in Verilog will be optimized and tested thoroughly in Synapticad and once it reaches its best performance it will be passed to Cadence.
This project has also been presented for financing to the Academic Research Bureau of the Catholic University of Peru.
In a second version of this architecture, it is planned to use inner and outer pipelining, this way we will get an AES core that works only in the ECB mode but with a really high through-output.

5. REFERENCES

[1] Specification for the Advanced Encryption Standard (AES) Federal Information Processing Standards Publication 197 http://csrc.nist.gov/encryption/aes/frn-fips197.pdf (nov - 2001)
[2] NIST. Advanced Encryption Standard (AES). NIST homepage about AES *http://csrc.nist.gov/encryption/aes*
[3] Gaj, Kris e Chodowiec, Vincent "Comparison of the hardware performance of the AES candidates using reconfigurable hardware".
[4] Patrick. Schaumont, Henry Kuo, Ingrid Verbauwhede. Unlocking the Design Secrets if a 2.29 Gb/s Rijndael Processor
[4_A] Patrick. Schaumont, Henry Kuo, Ingrid Verbauwhede. A 2.29 Gbits/sec, 56mw, Non-pipelined Rijndael Aes Encryption IC in a 1.8v, 0.18 um CMOS technology
[5] T. Ichikawa, T. Kasuya, M. Matsui, "Hardware Evaluationof the AES Finalists," in AES3: the third Advanced Encryption Standard Candidate conference, New-York,April 13-14, 2000.
[6] Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms http://csrc.nist.gov/encryption/aes/round2/NSA-AESfinalreport.pdf
[7] Rijndael Homepage by: J. Daemen and V. Rijmen *http://www.esat.kuleuven.ac.be/~rijmen/rijndael/index.html*
[8] The rijndael fan page: implementations: *http://rijndael.com/implementations.html*
[9] The advance Rijndael algorithm by John Savard *http://home.ecn.ab.ca/~jsavard/crypto/co040801.htm*
[10] M.H. Jing, Y.H. Chen, Y.T. Chang, C.H. Hsu "The Design of a Fast Inverse Module in AES" paper IEEE
[11] MicroWind2 cmos layout design tool from l'Institut National des Sciences Appliquées de Toulouse http://intrage.insa-tlse.fr/~etienne/Microwind
[12] Synapticad Verilogger (Verilog simulation environment) from SYNAPTICAD INC www.syncad.com/