UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE CIENCIAS E INGENIERIA

# Power and Area Improvements on an Integrated Circuit by Using Internal Counters on the Finite State Machine of a Reduced MD5 Encryption Chip

**Artículo Académico**

.

**Juan José Jiménez Villalba**

**Ingeniería Electrónica**

Trabajo de titulación presentado como requisito
para la obtención del título de
Ingeniero Electrónico

Quito, 10 de mayo de 2019

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE CIENCIAS E INGENIERIA

**HOJA DE CALIFICACIÓN**
**DE TRABAJO DE TITULACIÓN**

**Power and Area Improvements on an Integrated Circuit by Using Internal Counters on the Finite State Machine of a Reduced MD5 Encryption Chip**

**Juan José Jiménez Villalba**

Calificación:

Nombre del profesor, Título académico: Luis Miguel Prócel , PhD

Firma del profesor: _____

Quito, 10 de mayo de 2019

# Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Juan José Jiménez Villalba

Código: 124737

Cédula de Identidad: 1721783007

Lugar y fecha: Quito, 10 de mayo de 2019

# RESUMEN

**Debido al crecimiento en la utilización en circuitos integrados como aceleradores de hardware y la importancia de la encriptación de la información, se ha decidido unirlos en el diseño de un chip que realiza una versión reducida del algoritmo de reducción criptográfica MD5. Se probaron dos tipos de control basados en las máquinas de estados finitos de Moore. Uno de ellos utiliza registros como contadores que permitirán decodificar las salidas para los 139 estados necesarios para procesar una entrada de 16 bits a una salida de 8 bits. Considerando que el algoritmo requiere una constante iteración de las señales de control en distintos módulos del chip, se espera que el control que utilizo contadores sea más eficiente ya que estos podrán servir para módulos demutiplexores internos.**

**El chip se diseño con el método "Top Down" comenzando con un código en System Verilog hasta terminar con un plano del circuito integrado que utiliza una tecnología de 500nm utilizando el software de Synopsys. Después de que el chip haya validado las etapas de síntesis y compilación del flujo de diseño, se pudieron observar diferencias significativas entre los controles con distintas máquinas de estados finitos. El que utilizó contadores mostro una mejora del 22.17% en su consumo de energía y una reducción de 13.19% en utilización de área, principalmente debido a una reducción de 8.8% en el número de celdas requeridas para este chip.**

**Palabras clave: Circuito Integrado, System Verilog, Synopsys, Diseño Top-Down, Optimización de energía, Reducción de área, Algoritmo de reducción criptográfica MD5**

# ABSTRACT

**Due to the constant growth of integrated circuits as hardware accelerators and the importance of data encryption, we decided to join them on the design of a chip that performs a reduction of the MD5 Message Digest Algorithm. We tested two control types based on a Moore Finite State Machine. One of them had registers as counters that will help encode the output for each of the 139 states necessary to process the 16-bit input into a 8-bit output. Considering that this algorithm required constant iterations of control signals on different modules of the chip, we expected that the control that employed counters would be more efficient since these internal counters might serve on internal demuxing modules.**

**The chip went through a top down design approach where we started from the System Verilog code and ended up with the floorplan of an integrated circuit with a 500 nm technology by using Synopsys software. After the chip went through synthesis and compilation phases of the design flow, we saw significant differences between the chips with different finite state machines. The one that employed registers as internal counters showed a 22.17% improvement in power consumption and a 13.19% on area utilization mainly due to an 8.8% decrease in the number of cells required for the encryption chip.**

***Key words*: Integrated Circuit, System Verilog, Synopsys, Top-Down Design, Power Optimization, Area Reduction, MD5 Message Digest Algorithm.**

# TABLA DE CONTENIDO

# Power and Area Improvements on an Integrated Circuit by Using Internal Counters on the Finite State Machine of a Reduced MD5 Encryption Chip

Juan José Jiménez
Colegio de Ciencias e Ingeniería
Universidad San Francisco de Quito
Campus Cumbayá, PO-Box 17-1200-841
Quito, Ecuador
jjimenezv@estud.usfq.edu.ec

Luis-Miguel Prócel
Instituto de Micro/Nanoelectrónica
Universidad San Francisco de Quito
Campus Cumbayá, PO-Box 17-1200-841
Quito, Ecuador
lprocel@usfq.edu.ec

Lionel Trojman
Instituto de Micro/Nanoelectrónica
Universidad San Francisco de Quito
Campus Cumbayá,PO-Box 17-1200-841
Quito, Ecuador
ltrojman@usfq.edu.ec

***Abstract*—Due to the constant growth of integrated circuits as hardware accelerators and the importance of data encryption, we decided to join them on the design of a chip that performs a reduction of the MD5 Message Digest Algorithm. We tested two control types based on a Moore Finite State Machine. One of them had registers as counters that will help encode the outputs for each of the 139 states necessary to process the 16-bit input into a 8-bit output. Considering that this algorithm required constant iterations of control signals on different modules of the chip, we expected that the control that employed counters would be more efficient since these internal counters might serve on internal demuxing modules.**

**The chip went through a top down design approach where we started from the System Verilog code and ended up with the floorplan of an integrated circuit with a 500 nm technology by using Synopsys software. After the chip went through synthesis and compilation phases of the design flow, we saw significant differences between the chips with different finite state machines. The one that employed registers as internal counters showed a 22.17% improvement in power consumption and a 13.19% on area utilization mainly due to a 8.8% decrease in the number of cells required for the encryption chip.**

KEYWORDS

Integrated Circuit, System Verilog, Synopsys, Top-Down Design, Power Optimization, Area Reduction, MD5 Message Digest Algorithm

## I. INTRODUCTION

In an era were personal information is constantly shared online, concerns about our privacy and the security of our personal information is a pressing matter. Malicious activities like eavesdropping users, gathering data from unsecure storage or even obtaining information via monitoring an individuals network traffic increase the necessity to somehow protect our information. Storing or sending sensitive information, like usernames or passwords, as plain text should be avoided [1]. One way to protect our information is to encrypt our information in some way, like with the MD5 Message-Digest Algorithm.

Created in 1992 by Ronal Rivest, the MD5 Message-Digest Algorithm proved to be an improvement over its predecessor, the MD4 Message-Digest Algorithm. Its objective was to create a 128-bit output that would serve as a digital fingerprint for an input of an arbitrary length. It was expected to be used for digital signature applications since it would be computationally infeasible to recover the initial input by only having the output [2]. Even though the MD5 algorithm has been discredited for certain applications due to fact that it can suffer from collision or dictionary attacks, it is still an improvement over storing information as plain text.

Nevertheless, the MD5 is a protocol that was specified for use in the Internet Protocol Security (IPSEC) as the basis for the HMAC, which is used to produce MAC addresses. Considering that the MD5 Message-Digest Algorithm is currently being used, there have been attempts to produce cryptographic accelerators for IPSEC applications by implementing this algorithm on FPGAs by a top down design describing the operations in VHDL [3].

A top down design of an integrated circuit starts from the description of the processes that the hardware is going to perform via a Hardware Description Language (HDL) like Verilog or System Verilog [4]. The next step is the logical synthesis in which via software, the hardware described in the previous step is converted into a logic circuit with logical cells. In this step, the circuit is also optimized to use less amount of cells possible or include additional elements, like buffers, to insert delays to ensure a correct operation of the hardware based on constrains that are introduced at this stage [5]. The optimization on this stage usually becomes a tradeoff between power, area and timing depending on the requirements of the chip. The designer must choose the most optimal solution for its application. Simulations to ensure that the chip would work as expected are also performed in this stage.

The next step on the top down design is the physical compilation. In this stage, the logical cells are replaced with

real standard cells that will be placed on the floorplan of a chip. By placing and routing real cells on the floorplan, the simulations becomes more realistic and problems congestion, antennas created by wires connecting cells or delays in the clock might arise. Once these problems are addressed, and optimization of placement and routing of the cells is complete, the result will be the final design of a chip that accomplishes the task it was designed for [6].

For the compilation stage, the technology employed was a 500nm library that contains characterized standard cells designed by the Universidad de San Martin in Buenos Aires, Argentina [7].

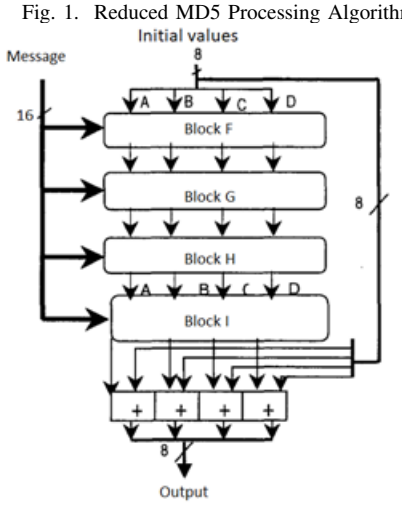
## II. Methodology

To test different types of controls and evaluate its improvements in area, timing and power usage on a chip dedicated for encryption, the first stage will be to create the chip under the specifications we desire. Since the original MD5 algorithm is lengthy and requires numerous iterations, we have considered making a reduction to the algorithm to avoid problems that might come with having a large number of logical cells. The algorithm will be reduced to take a 16-bit message as an input and give an 8-bit digest as an output, but it will maintain the original functions and methods the original MD5 method employs. This will be enough to compare the different control types to evaluate the improvements they might have. The steps taken to evaluate the different controls are the following:

### A. MATLAB Code

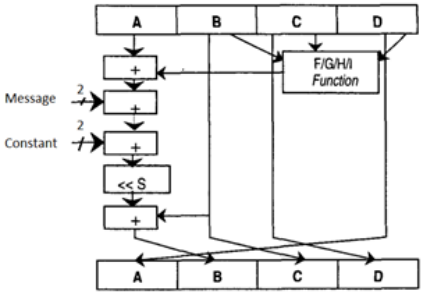
Since the MD5 algorithm was reduced to take a message of 16 bits as an input and 8 bits as an output, the first stage was to modify the MD5 algorithm but maintaining the original processing blocks. The flow of the algorithm is presented in Figure 1.

Fig. 1. Reduced MD5 Processing Algorithm




In each block presented above, the output is the result of various operations done to the message, initial values and constants. The operations include: adding the first two bits of the input with two bits from the message (and discarding the carry out), adding that result to a 2 bit constant, then adding it to the result of a logical function that is different in each block, shifting the order of the first two bits, adding it to the third and forth bits of the input and finally rearranging the bits in order to obtain an 8 bit output. A diagram which explains this process is presented in Figure 2.

Fig. 2. Reduced MD5 Processing Algorithm




The output of each step is reused as the input of the step. The F Block is used 8 times to use the full 16 bits of the message but only 2 at the same time. We will use 4 constants which are used on a cyclical order. The constants are determined by the following formula:

$$C(i) = floor(2^2 * |sin(i+1)|) \tag{1}$$

And the values for them in bits are:

$$C(1) = 11$$

$$C(2) = 11$$

$$C(3) = 00$$

$$C(4) = 11$$

In the first iteration of the Block F, initial values are required for the processing of the information. These initial values are the following:

$$A_0 = 00$$

$$B_0 = 01$$

$$C_0 = 10$$

$$D_0 = 11$$

The functions that are used in each block come from the original MD5 algorithm and they are the following:

$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D) \tag{2}$$

$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D) \tag{3}$$

$$H(B,C,D) = B \oplus C \oplus D \tag{4}$$

$$I(B,C,D) = C \oplus (B \vee \neg D) \tag{5}$$

At the end, the encrypted message is added to the initial values with a full 8-bit adder that discards the carry out to finally get the message digest.

For this method to be an effective MD5 reduction, its output should accomplish 2 objectives. The first one is that there is not a pattern between the outputs that can help identify the input. The second one is that between all the outputs, they should have a similar frequency to show that it does not prefer one output over another one. These objectives are shown on figures 3 and 4.

Fig. 3. Frequency of all Possible Outputs

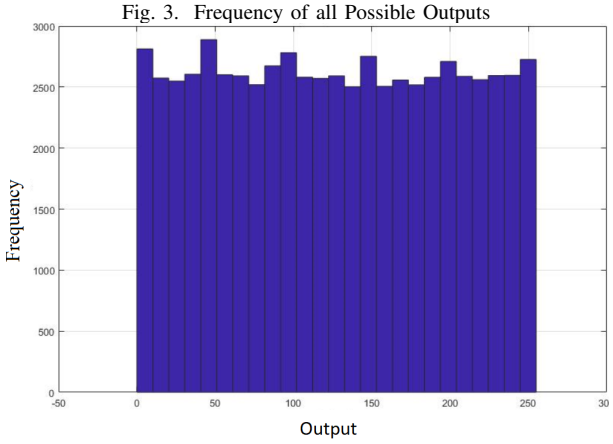
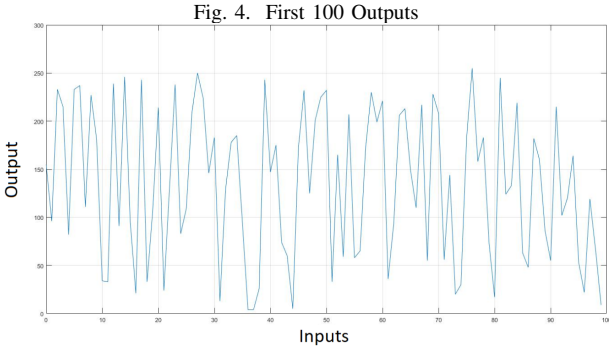



Fig. 4. First 100 Outputs



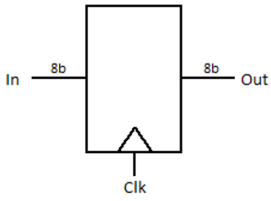

Like figure 3 shows, the frequency of the outputs from all the 65536 possible inputs (created from a 16-bit input) have a pretty similar frequency. This means that this reduction does not significantly favors any output over another one. We can also see that the outputs dont follow a pattern, based on figure 4 which shows the first 100 outputs.

### *B. System Verilog Implementation*

The next stage was to create the description of a hardware in System Verilog that would implement the reduced MD5 that we previously implemented using MATLAB. To produce the same outputs, some basic modules were needed to be created. These modules are the following:

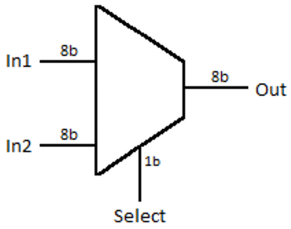
*1) Register with enable:* To create an 8 bit register with an enable, we first created a register that would work on a positive transition of the clock. The schematic for this register is shown on figure 5.

Fig. 5. Register Schematic



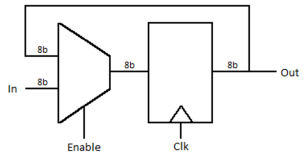

We also required a 16 to 8 bit multiplexor with 1 bit for the control. The schematic for this module is shown on figure 6.

Fig. 6. 18 to 8 Mux Schematic



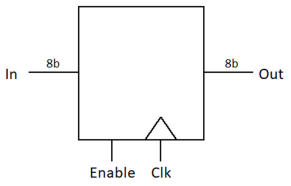

By placing this two modules side by side in the following way:

Fig. 7. Register with Enable Placement



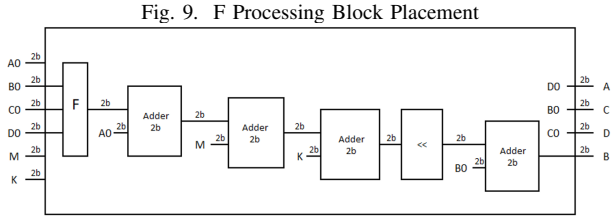

We can create a register that will work normally if the enable pin is set to high, but will retain the output in case that it is set to low. The final schematic for this module is shown on figure 8.

Fig. 8. Register with Enable Schematic



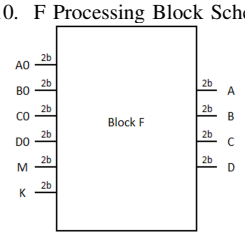

*2) Blocks F, G, H and I:* To create the blocks that perform the combinational logic of the functions that the MD5 algorithm employs, modules that perform a 1 bit shift, a 2 bit full adder that discards the carry out and modules that performs the logical functions from Equations 2 through 5 need to be created. Once created, they must be joined into a module to perform operation required. Figure 9 shows the placement of these modules to create Block F. Notice that it has an internal module that performs function F, described in Eq 2.

Fig. 9. F Processing Block Placement




The blocks for the function G, H and I are similar to this one, but they implement their own function by using their own module. Figure 10 shows the final schematic of these modules. In this case, Block F is used as an example.

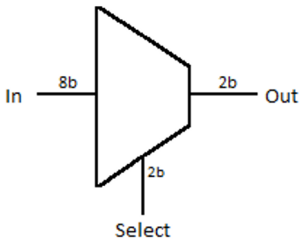
Fig. 10. F Processing Block Schematic




In these modules, the M represents the 2 bit section of the message that is being used while the K represents 2 bits constant.

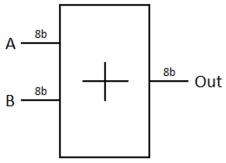
*3) Other Modules:* For the encryption of the message according to the MD5 algorithm, other smaller modules were required. One of them was an 8 to 2 Multiplexor that has 2 bits of control. Its schematic is presented in figure 11.

Fig. 11. 8 to 2 Mux Schematic




Another module necessary was an 8 bit full adder that discards the carry out. Its schematic is presented in figure 12.

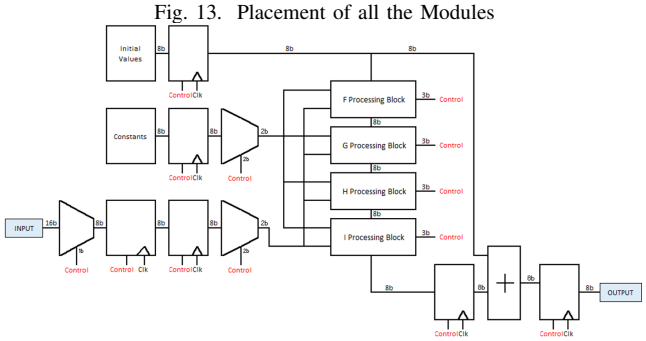
Fig. 12. 8-bit Full Adder Schematic




These are all the modules that are going to be required to implement the reduced MD5, however, there is still a control required to make sure that the encryption works according to plan.

The complete schematic that shows the placement of all the modules to finally get the full encryption module is presented in figure 12.

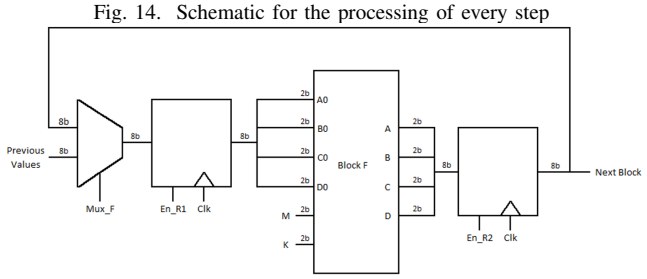
Fig. 13. Placement of all the Modules




All the signals label control will come from the control module explained on the next section. The Processing blocks will be explained on the next section as well.

*4) Control:* It is important to understand that the control module works as a finite state machine which changes its current state on every positive transition of the clock. Therefore, in each state it should be sending the correct control signals to the multiplexors and enables from the registers to perform the operation that is necessary for the flow of the data. To understand the logic behind the control module, it will be helpful to understand how the processing of each step inside each block works. The schematic for the processing of each step is shown on figure 14.

Fig. 14. Schematic for the processing of every step




As we can see in the image above, this block has 7 inputs. However, excluding the two clock signals, the control only has to focus on 5 signals to process on each step. The processing of each step is done in four states of a finite state machine. These steps are the following:

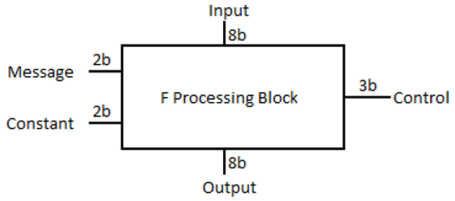
1) In this stage, the values for the constants and for the message are changed by changing their own multiplexors so that the correct value enters the Block F. The signal control of Mux_F is set to 0 so that it gathers the previous values and the enable of the first register, En_R1 is set to 1 so that it acquires the value of the Initial Values. The enable of the second register is set to 0 since the information has not been processed yet.
2) In this stage the enable of the first register changes from 1 to 0 so that it saves the current value that will enter the Block F. It is important to know that since the

moment that the previous values, the message and the constant are propagated to the Block F, this block will start to process this information with its combinational logic. Therefore the value that outputs the Block F might already have changed before this stage. But in this state we only want to block the first register.

3) In this stage, the enable from the second register is set to 1 so that it can acquire the same value that is coming out from the Block F.
4) On the forth state, we change the enable from the second register again to 0 so that it holds the value that was changed on the previous state. This is important because this value cannot change until the processing of the next step is done. At this point one step is finally completed.

Another fact that we should take into consideration is that these 4 stages in which data are processed will repeat with minor changes 8 times for each block, before it passes to the next block. Then they will repeat for the next 3 blocks. This constant repetition will give us a change to optimize the control and eventually the chip overall. The processing modules presented on figure 14 will be reduced to the schematic presented on figure 15.

Fig. 15. Reduced schematic for the processing of every step




Notice that these schematics were previously used on figure 13.

For the creation of both controls we require the use of 139 states. The first 8 will have all the enables on 1 so that the information can properly propagate through all the cells to avoid problems by not allowing enough settling time. Then we have 128 states for the processing of every step since we need 4 states for each step, 8 steps for each block and there are 4 blocks in total. And finally, 3 more states so that the final addition can take place and be saved on a final register. At the end, a final signal named Ready will turn to high when the data is finally processed.

The control will have to input signals which are the clock and an asynchronous reset that can be activated at any time to start the processing from the start. It will have 20 outputs connected to the enables of all the different registers and the multiplexors of the encryption module, as well as an extra control signal that will turn to high when the encryption is ready. With these ideas in mind we proceeded to create two different control modules to see what the advantages were of using one over the other.

*5) Control Module 1: Moore FSM without registers as counters:* The first control we will evaluate will be a Moore finite state machine. Since the Moore machines output can only depend only on the current state and we have 139 different states to encode, we need 8 bits to be able to encode this machine. In each state a combinational logic will be needed so that all the output signals take the values that are necessary for that particular state. Each time there is a posedge clock transition, the machine will go to the next state until it reaches the last one. In the last state, the output named ready will change to high so that the user can know the processing is finally done. The machine will stay in this state forever unless it receives a signal from the reset input, which will start this process all over again.

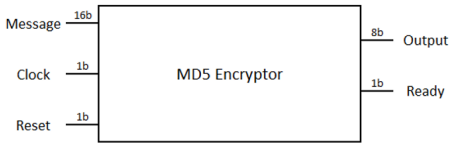
*6) Control Module 2: Moore FSM with registers as counters:* Similarly, we will need to encode 139 different states for this finite state machine. However, in this machine we will encode only 4 states since only 4 are necessary to perform every step. However, we will use registers as counters for different applications to hopefully reduce the number of cells required. Our hypothesis is that the same 4 stages are required to do the processing for most of the steps with few exceptions, therefore, the combinational logic behind each state can be the same one. By using external counters, they will eventually become signals for demultiplexers so that these outputs can go to the correct output without creating a combinational logic for each one of the states.

To describe this idea in System Verilog, we ended up using 2 bits to encode the actual state, but also two 3-bit counters and two 1-bit flags, ending up in a total of 10 bits to encode the states. This was done by nesting up to 3 case statements to decode the output for each of the 139 states.

After we tested both the control modules and determined that they gave the same output signals, we joined this module to the encryption module and went on the process to the synthetization.

Regardless of the control we use, by joining it to the encryption module the final schematic of the complete chip will be shown in figure 16.

Fig. 16. Schematic of the complete chip




## C. Synthesis

In the synthesis phase we employed the tools the program Design Compiler from Synopsys to go from our System Verilog files to an optimized mapped netlist with standard cells. We set up a time constrain which was a clock period of 15ns since it was the lowest speed possible at which both of our designs could operate successfully. With these constraints in mind, the compilation process determined which cells will be the most adequate depending on the location of the cell. For example, on a non-critical path, having fast cells that consume a lot of power might not be effective since time is not so

important in that path. Changing it to a slower cell that does not consume much power might be more effective. Once the synthesis was done, these results were used for the compilation for the actual floorplaning of the chip.

### *D. Compilation*

In the compilation phase, the software IC Compiler gathers the logical cells from the synthesis and places them in a floorplan to create a real chip. On this stage, the software makes power rings to supply energy to the cells. Then it makes an initial placement of the cells which then is optimized to avoid congestion problems. Afterwards, the chip is evaluated under the clock tree synthesis (CTS) flow to make sure that the clock signal gets to all the cells without a small enough delay that might make the chip process the data with mistakes. To achieve this, some places buffers might be inserted so that the delay can as similar as possible in every place this signal is required while still optimizing the area of the chip. Then the chip undergoes routing were the most efficient path is searched. All these steps are done in order to optimize timing, area and power to finally get the floorplan of the chip. As stated before, we will use a library of standard cells of a 500nm technology for this stage.

## III. Results

In both the synthesis phase and the compilation phase, we gathered reports that allow us to be able to compare both control methods implemented.

### *A. Synthesis Results:*

Tables 1 through 4 show the information obtained in the reports after the synthesis was done.

| | 2nd Control With counters [mW] | 1st Control Without counters [mW] | Improvement % |
|---|---|---|---|
| **Cell Internal Power** | 5.97 | 7.95 | 24.91 |
| **Net Switching Power** | 0.74 | 1.00 | 26.00 |
| **Total Dynamic Power** | 6.71 | 8.95 | 25.03 |
| **Cell Leakage Power** | 0.00011 | 0.00014 | 21.43 |
| **Combinational Power** | 22.65% | 22.84% | 0.83 |

Table 1:
Synthesis Power Consumption

| | 2nd Control With counters [nm^2] | 1st Control Without counters [nm^2] | Improvement % |
|---|---|---|---|
| **Total Area** | 196207.20 | 232560.00 | 15.63 |
| **Combinational Area** | 131482.08 | 147296.16 | 10.74 |
| **Non-Comb Area** | 64725.12 | 85263.84 | 24.09 |

Table 2:
Synthesis Area Utilization

| | 2nd Control With counters | 1st Control Without counters | Improvement % |
|---|---|---|---|
| **Total Cell Count** | 1009 | 1138 | 11.34 |
| **Combinational Cells** | 875 | 970 | 9.79 |
| **Sequential Cells** | 134 | 168 | 20.24 |

Table 3:
Synthesis Cell Utilization

| | 2nd Control With counters | 1st Control Without counters | Improvement % | Units |
|---|---|---|---|---|
| **Critical Path Length** | 12.94 | 13.26 | 2.41 | nm |
| **Critical Path Slack** | 0.01 | 0.01 | 0.00 | ns |

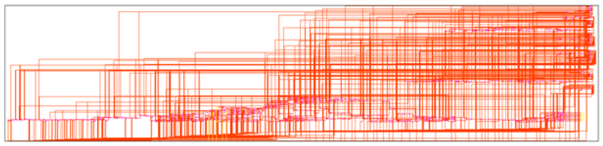
Table 4:
Synthesis Critical Path Results

As we can see, there were some significant results in power, area and timing in the utilization of the control that has internal counters. However, these results are preliminary since once we go through the compilation phase and other optimization stages take place, like congestion analysis or clock tree synthesis, small changes might arise as the simulation becomes more accurate. The final result of the synthesis is the logical circuit that performs the processes described in the System Verilog files. The final schematic is presented in figure 17.

Fig. 17. Schematic of the complete chip

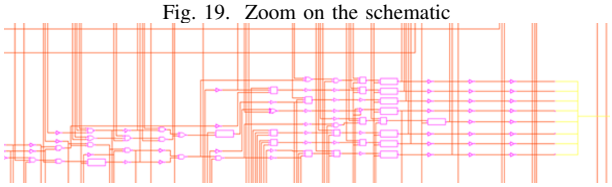

By opening the schematic, we are able to see the logical cells necessary for the implementation of the chip. They are shown figures 18 through 20.

Fig. 18. All logical cells and their connections



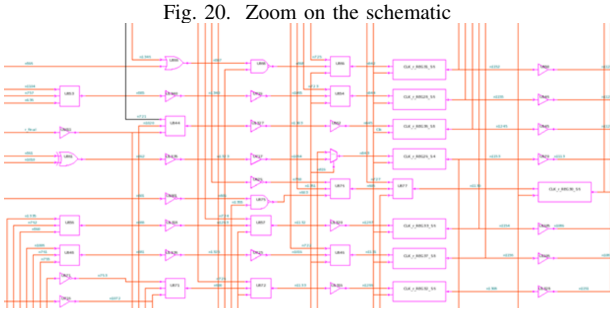
By zooming the previous image, we can see the logical cells implemented:

Fig. 19. Zoom on the schematic



As we can see on figure 20, each cell has an identifier. These names will also show up on the layout after compilation.

Fig. 20. Zoom on the schematic



## B. Compilation Results:

Once all the different optimizations that are performed on the compilation phase are done, we can gather the results from the reports that are generated. These results are summarized on tables 5 through 9.

| | 2nd Control With counters | 1st Control Without counters | Improvement |
|---|---|---|---|
| | [mW] | [mW] | % |
| **Cell Internal Power** | 6.93 | 9.09 | 23.76 |
| **Net Switching Power** | 2.83 | 3.45 | 17.97 |
| **Total Dynamic Power** | 9.76 | 12.54 | 22.17 |
| **Cell Leakage Power** | 0.00012 | 0.00014 | 14.29 |
| **Combinational Power** | 23.12% | 21.40% | -1.72 |

Table 5:
Compilation Power Consumption

| | 2nd Control With counters | 1st Control Without counters | Improvement |
|---|---|---|---|
| | [nm^2] | [nm^2] | % |
| **Total Area** | 223380.00 | 257309.28 | 13.19 |
| **Combinational Area** | 158361.12 | 171898.56 | 7.88 |
| **Non-Comb Area** | 65018.88 | 85410.72 | 23.88 |
| **Area of CT Buffers** | 4112.64 | 5091.84 | 19.23 |

Table 6:
Compilation Area Utilization

| | 2nd Control With counters | 1st Control Without counters | Improvement |
|---|---|---|---|
| | | | % |
| **Total Cell Count** | 1262 | 1385 | 8.88 |
| **Combinational Cells** | 1128 | 1217 | 7.31 |
| **Sequential Cells** | 134 | 168 | 20.24 |
| **CT Buffers** | 21 | 26 | 19.23 |

Table 7:
Compilation Cell Utilization

| | 2nd Control With counters | 1st Control Without counters | Improvement | |
|---|---|---|---|---|
| | | | % | Units |
| **Critical Path Length** | 14.28 | 14.84 | 3.77 | nm |
| **Critical Path Slack** | 0.05 | -0.01 | 600.00 | ns |

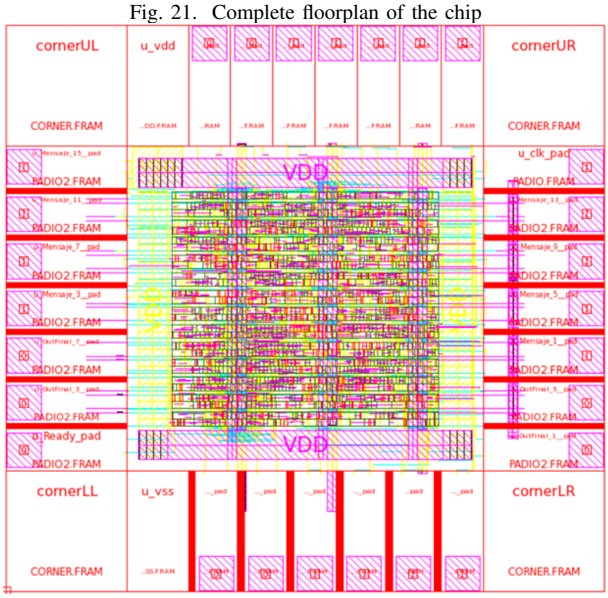
Table 8:
Compilation Critical Path Results

| | 2nd Control With counters | 1st Control Without counters | Improvement |
|---|---|---|---|
| | [ns] | [ns] | % |
| **Max Global Skew** | 0.345 | 0.488 | 29.30 |
| **Longest Path Delay** | 4.204 | 4.409 | 4.65 |
| **Shortest Path Delay** | 3.859 | 3.921 | 1.58 |

Table 9:
Compilation Timing Results

Finally, the last result of the compilation is a finished floorplan chip where all the cells are placed and routed on a chip. This floorplan is presented on figure 21.

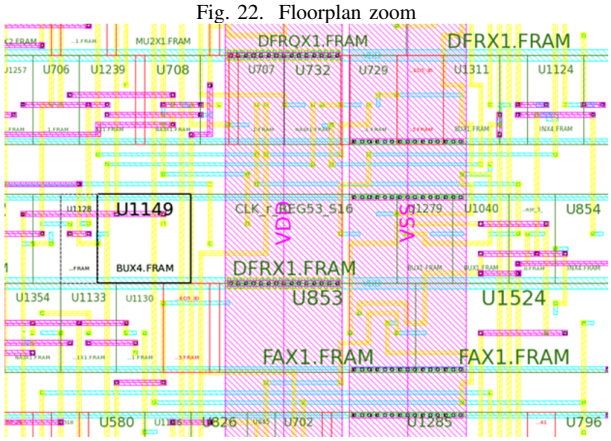
Fig. 21. Complete floorplan of the chip




On this image we can see the layout of the chips. On the exterior we can see the input/output paths that the chip will use. Each I/O path has its own name that corresponds to every input and output. For signals that have more than 1 bit there is a number identifier to which bit it corresponds to. Like the ones on the bottom right corner which are named OutFinal_5 and OutFinal_1.

By zooming on the previous image, shown in figure 22, we can see the individual cells with their names that correspond to the schematic presented on the results of the synthesis. We can also see the power strips from VDD and VSS and the metal connections that occur between the logical cells.

Fig. 22. Floorplan zoom




## IV. Conclusions

The objective of this investigation was to determine if having a finite state machine with internal counters was more efficient than another one that did not employ them in the control module. It was interesting to see that the one which employ them ended up using 10 bits to decode the outputs of every signal on each state, while the one which did not required 8 bits. From the synthesis phase we could see that there were considerable improvements over the control module which employed counters over the other one. However, since the compilation phase simulates the system with more accurate data by using physical libraries, we will focus on that data for our results.

Looking to through the compilation results, we could clearly see that there was an improvement from the control method that employed counters on the control over the ones that did not. The total power consumption decreased on 22.17%. However, we can see that the power used on combinational circuits increased by 1.72% by using this method, which we did not expect, considering that the combinational cells decreased in 7.88%. We can speculate that the compilation could have optimized some other parameter like timing or congestion that by using cells that require more power.

Like we expected, we saw an 8.88% reduction in the number of cells that they were necessary for the chip. This also led to a decrease in overall area usage of the chip which decreased in a 13.19%. What also was interesting was that there was a considerable timing improvement with the control that used counters over the other one. The one that used counters ended up achieving the constrain stated on the synthesis phase which stated that the clock will have a period of 15ns by 0.05ns, while the initial one ended up not being able to achieve timing by 0.01ns. That is why the improvement percentage ended up being 600%. The shortest path delay and longest path delay also showed a small improvement as well. These results show an improvement on the control that employed internal counters on its FSM over the one that did not for this application.

## References


[1] X. Zheng and J. Jin, "Research for the application and safety of MD5 algorithm in password authentication," *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery,* Sichuan, 2012, pp. 2216-2219. doi: 10.1109/FSKD.2012.6234010

[2] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992. DOI 10.17487/RFC1321.

[3] J. Deepakumara, H. M. Heys and R. Venkatesan, "FPGA implementation of MD5 hash algorithm," Canadian Conference on Electrical and Computer Engineering 2001. Conference Proceedings (Cat. No.01TH8555), Toronto, Ontario, Canada, 2001, pp. 919-924 vol.2. doi: 10.1109/CCECE.2001.933564.

[4] Digital IC Design Flow. Datasheet. Assistants. SYNOPSYS. Accessed on: Apr. 10, 2018, Available: https://www.synopsys.com.

[5] Custom Compiler. Datasheet. Assistants. SYNOPSYS. Accessed on: Apr. 10, 2018, Available: https://www.synopsys.com.

[6] IC Compiler. Datasheet. Assistants. SYNOPSYS. Accessed on: Apr. 10, 2018, Available: https://www.synopsys.com.

[7] Y. Kuo, L. J. Arana, L. Seva, C. Marchese and L. Tozzi, "Educational design kit for synopsys tools with a set of characterized standard cell library," *2018 IEEE 9th Latin American Symposium on Circuits Systems (LASCAS),* Puerto Vallarta, 2018, pp. 1-4. doi: 10.1109/LASCAS.2018.8399907