FPGA design example for maximum operating frequency measurements

Carlos Jiménez Fernández, Pilar Parra Fernández,
Carmen Baena Oliva, Manuel Valencia Barrero
Dpto. Tecnología Electrónica
Universidad de Sevilla
Instituto Microlectrónica de Sevilla
IMSE-CNM CSIC/US
Sevilla, España
{cjesus, parra, baena, manolov}@imse-cnm.csic.es

F. Eugenio Potestad Ordóñez
Instituto Microlectrónica de Sevilla
IMSE-CNM CSIC/US
Sevilla, España
potestad@imse-cnm.csic.es

***Abstract*— The best way to learn how to design digital systems at the RT level is to use practical examples. In addition, from a teaching point of view, the more practical they are, the more attractive to students. But for a design to be attractive, even if it is presented with a low complexity, it is not possible to do it in a single practice session. This paper presents, as a demonstrator, the design at RT level and its implementation in FPGA of a digital system that uses the Trivium flow cipher and on which measurements of maximum operating frequency are made. This circuit is designed in three laboratory sessions of about two hours each.**

Keywords— VHDL, digital systems design, FPGA, Chipscope.

I. INTRODUCTION

Digital design subjects in advanced electronic degree courses can use hardware description languages for the description of circuits and FPGA devices as a technology to experimentally test the behaviour of designed circuits. This methodology has many advantages. One of them is the use of a single CAD environment for the design, verification and programming of devices. In addition, this CAD environment is offered free of charge for educational use by FPGA manufacturers. Another advantage is the availability of development boards that include, in addition to the FPGA, display and interface elements that allow the input values to be entered and the output values to be viewed. For all these reasons, teaching digital design with the VHDL-FPGA tandem is a low-cost option that, above all, allows experimental testing of the designs, which is highly attractive to students.

This alternative is being applied in the course "Advanced Digital Design"[1], an elective in the fourth year of the Degree in Industrial Electronics at the Polytechnic School of the University of Seville. The aim of this course is for students to acquire the most important skills in digital design. The only previous knowledge on which this subject is built is the subjects "Industrial Electronics" and "Digital Electronics", both of which are in the second year.

The subject "Industrial Electronics" is a subject of the common training block of the industrial branch, taught in the first four-month period of the second year. It is the first subject that students have in this degree related to electronics. The contents of this subject [2] are divided into two blocks: an analog block and a digital block. In the analog block, the amplification and filtering operations are studied using operational amplifiers. In the digital block, the basic concepts of digital electronics are introduced, from switching algebra to the design of state machines, through the concepts of logic and bistable gates.

The subject "Digital Electronics" is a compulsory subject in the degree curriculum. It is taught in the second term of the second year. Its contents [3] develop are based on those in Industrial Electronics. They include real logic gate and bistable gate features, analysis and design of digital circuits (both combinational and sequential) and combinational and sequential subsystems. The latest topics in the course introduce concepts related to design at RT level (circuit structure based on control unit and data unit), ASM charts and basic microprocessor principles. Although some laboratory sessions are done using FPGA devices, designs based on schema capture environment and no hardware description languages are used.

With this previous knowledge on the part of the students, in the course "Advanced Digital Design" the students are taught the description of digital systems using VHDL hardware description language and the way to implement them on FPGA devices (in our case Xilinx). The subject is presented in a very practical way, so that in a two-hour session a theory part is combined with a laboratory part, in which the students begin by designing small circuits. As the subject progresses, the time dedicated to the theory part is reduced and the laboratories increase in complexity, but always taking into account that the complexity is not too great to give the student time to complete the practice in the time assigned to the laboratories.

The two-hour limit seems to require that no minimally complex systems be developed or relegated to a project-based learning methodology [4][5]. However, one solution to this problem is the reuse of previous designs or the realization of a design in several sessions. For this solution to be interesting, it must have several characteristics: on the one hand, each part must be self-contained (it must have a design objective that can be achieved in a laboratory session) and, on the other hand, it must involve a gradual construction of the functionality to be achieved. The different laboratory sessions should gradually build up the functionality of the entire system. The design of

978-1-5386-0928-6/18/$31.00 ©2018 IEEE

the final system can be achieved both by breaking it down into simpler parts (divide and conquer methodology) and by building from little to much (bottom-up methodology).

In this paper we present as a demonstrator a set of three laboratory sessions whose aim is the temporal analysis and measurement of the maximum frequency of operation of a moderately complex digital system, combining CAD tools and experimental measurements. The implementation of these three sessions entails, in addition, the handling of clock signal generator blocks of FPGA devices (to obtain high frequency clocks) and also the handling of the Chipscope tool, which is a built-in logic analyzer, whose internal signals can be displayed during the operation of a circuit.

The structure of this communication is as follows: the second section briefly explains the design to be carried out. Section III details the contents and the design objectives of each of the sessions into which the design has been divided and the results to be obtained. Finally, some conclusions are drawn.

II. DESIGN TO BE DEVELOPED

This set of laboratory sessions is based on the Trivium [6] flow cipher. Flow ciphers are private key cryptographic circuits used for bit-by-bit data encryption. Its function is to generate a pseudo-random sequence using a (secret) key and an initialization vector (IV) that can be public. The message is encrypted by making the XOR operation between the unencrypted message and the pseudo-random sequence generated. In the receiver, the message is decrypted in the same way, by means of an XOR operation between the encrypted message and the same pseudo-random sequence.

For this reason the process of encryption and decryption is relatively simple and the same for the encryption and decryption process. The same circuit is used in the encryption and decryption process. The main difficulty when using this type of encryption is that both, emitter and receiver must synchronize their operation very well. The variation in a single clock cycle between encryption and decryption causes the decryption process to be incorrect. It should also be noted that every time the encryption is restarted, a new initialization vector must be used, as the use of the same initialization vector creates a significant security problem.

The circuit structure of flow ciphers is usually based on shift registers with non-linear feedback. The Trivium cipher is one of the finalist flow ciphers in the European eSTREAM Project [7], which aimed to select new, more secure cipher proposals. The Trivium encryption is optimized for hardware implementation. It consists of three registers, which add up to a total of 288 bits, and some feedback (linear and non-linear). The register with the 288 bits is usually called a state register. The cipher requires an 80-bit key, which must be secret and known only by both the flow cipher and decipher, and an initialization vector (IV), also 80-bit, which can be public. The circuit structure is shown in *"Fig. 1"*. Initially the state register is loaded with the key and the IV and 1152 clock cycles are allowed to pass before a valid *key_stream* can be generated. In data encryption, the plain text supporting the data is combined (with an XOR operation) with the Trivium *key_stream* output signal to generate the encrypted text stream. In decryption, this encrypted text is combined (also with an XOR operation) with the *key_stream* output signal from the Trivium to retrieve the original plain text sequence.

III. DEVELOPMENT OF THE LABORATORY SESSIONS

The three laboratory sessions to be developed have the following partial objectives:

- Session 1: Design of the Trivium flow cipher: Functional verification, definition of time constraints and static time analysis.
- Session 2: Creation of a state machine that controls the loading of the key and the IV and the operation of the Trivium. Introduction of a DCM (Digital Clock Manager) module to generate a clock signal with different frequencies. Verification of maximum operating frequency with post-route simulations.
- Session 3: Implementation of the design on a Nexys4 DDR board and use of Chipscope to experimentally measure the maximum operating frequency.

The Xilinx ISE tool, version 14.7, and the Digilent Nexys-4 DDR board are used to carry out the sessios.

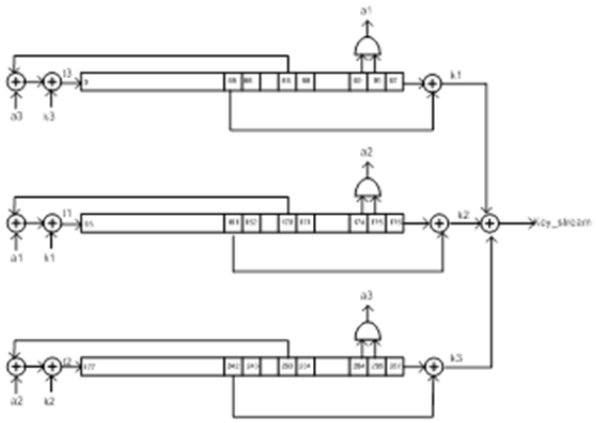


Fig. 1. Trivium flow cipher circuit structure.

A. First Session: Trivium flow cipher Design

As it can be seen in *"Fig. 1"*, the VHDL design of the Trivium is quite simple. It has as inputs the clock (*clk*), an asynchronous reset signal active at low (*reset*) and two control inputs: *ce* that works as an enabling input for operation and *ld* that controls the loading of the key and the initialization vector. To simplify the design, the *key* and the initialization vector (*IV*) are defined as constants within the cipher. As output it has the *key_stream* signal, through which the pseudo-random sequence generated by the cipher is obtained.

The code will contain, among others:

- The declaration of a *state* signal containing the state register (288 bits) and the declaration of the *key* and *IV* constants with the included values in the laboratory memory:

```
key = X"0F62B5085BAE0154A7FA"
iv  = X"F67A079428CF0AF960C7"
```

- The generation, in a combinational and concurrent way, of intermediate signals (a1, a2, a3, k1, k2, k3, t1, t2, t3) and of the *key_stream* (see *"Fig. 1"*):

```
k1 <= state(65) XOR state(92);
k2 <= state(161) XOR state(176);
k3 <= state(242) XOR state(287);

a1 <= state(90) AND state(91);
a2 <= state(174) AND state(175);
a3 <= state(285) AND state(286);

t1 <= k1 XOR a1 XOR state(170);
t2 <= k2 XOR a2 XOR state(263);
t3 <= k3 XOR a3 XOR state(68);

keystream <= k1 XOR k2 XOR k3;
```

- A synchronous process, triggered by the rising edge of the clock and with an asynchronous reset active on high. When resetting is active, the entire state register must be reset to zero. When *ce* is set to zero, the cipher must maintain its state. When *ce* is at one, if *ld* is at one it must load the *key* and *IV*, and if *ld* is at zero it must operate making the shifts.

The *key* and *IV* are loaded according to the Trivium cipher specifications [6].

In addition to the cipher design, the student must create a testbench for functional verification. The testbench, in addition to generating the signals oriented to load the key and the IV, must make the cipher work for 1000 clock cycles, deactivate the *ce* signal for 10 cycles and then make it work indefinitely.

Once the Trivium cipher is designed and the testbench generated, it is necessary to perform simulations and measurements:

- Carrying out a functional simulation.

From the functional simulation, it is requested to note the most significant bits (in hexadecimal) of the state register (*state* signal) in time at the moment when after 1000 clock cycles the cipher is stopped. This value will be used to know the maximum operating frequency.

- Performing an implementation.

Analysis of the reports offered by the Xilinx tool looking for: consumed resources (slices registers, luts used as logic, % of FPGA occupation), as well as the maximum operating frequency and the delays between the inputs and the clock edge. The delay between the clock edge and the output is also noted.

- Carrying out post-route simulations.

The objective is to know the minimum clock period of the circuit. For this purpose, a first post-route simulation with a clock of 10 ns of period is requested. First, it is necessary to check if the behavior is correct by comparing the values in the state register with those noted for the functional simulation. Then, the clock period is shortened until the contents of the state register do not match with the values obtained after functional simulation. The minimum value of the period for which the system is working correctly is noted and it will be used in future sessions.

- Temporary restrictions imposed.

A time restriction is imposed on the clock signal of 5 ns of period. A new implementation is made and the reports are analyzed to see if it has been possible to comply with this restriction. A post-route simulation is performed again and the minimum period of time for which the operation is correct is checked.

At the end of this practice, not only has the design and functional verification of the Trivium cipher been carried out, but temporary data have also been obtained with post-route simulations and static temporal analysis.

B. Second session: Use of clock generators (DCM)

This second laboratory session aims to create a state machine that generates input signals for the Trivium encryption, as well as generating the Trivium clock signal via an MMCM. However, to simplify the development of the laboratory, students are provided with the state machine design.

The state machine provided to them has the following functionality: it loads the key and the IV, runs the Trivium and waits for 1023 clock cycles to pass, sets a special output to one during a clock cycle, and keeps the Trivium encryption running continuously until a reset signal is activated. This special output will serve as a reference to check if the generated key_stream is correct or not.

The procedure to follow in this session is as follows:

- Carrying out a functional simulation.

The student must create a testbench and perform a functional simulation checking the correct functioning of the Trivium (loading, operation and generation of the special signal).

- Enter a MMCM to generate the clock.

It is requested to create a new design that will have the same behaviour and the same inputs and outputs as the circuit with the state machine but in which a MMCM (Mixed-Mode Clock Manager) clock signal generation block is placed. Therefore, the design code of the state machine is copied into this design.

The MMCM is inserted by creating a new IP-type source, and configuring it so that it has a 100 MHz input clock and a 200 MHz output clock (*clkfx*). Once generated, it is incorporated as a component into the circuit with the state machine, making both the Trivium clock and the state machine clock the output clock of the MMCM block (*clkfx*).

- Create a testbench and perform a functional simulation.

A new testbench is created and the simulation is performed. An interesting point is to check that the *clkfx* clock has a frequency of 200 MHz.

- Perform a post-route simulation.

In order to analyze the maximum operating frequency from the simulation point of view, a design implementation and a post-route simulation are performed. But in this case the clock frequency is not changed in the testbench, but in the MMCM module, so that although the input clock frequency remains at 100 MHz, the frequency of the *clkfx* clock signal is changed. With these simulations, a maximum operating frequency of the circuit is obtained again.

- Maximum external frequency check.

Finally, a final analysis is proposed. This analysis is experimental and consists of displaying the FPGA output signals on the oscilloscope. To do this, the circuit is implemented on the Nexys-4 DDR board and the key_stream output of the encryption device is connected to an oscilloscope. By changing the frequency of the *clkfx* clock and programming the board you can see the waveform and whether it is visible or not.

The results of this practice are various: on the one hand, the MMCMs has been used to internally generate clock signals with both lower and higher frequencies than the input clock frequency. The maximum operating frequency has been measured again with post-route simulations and it has been possible to see how the digital outputs degrade when the operating frequency is increased.

"Fig. 2" shows screenshots of functional and post-route simulations for a frequency of 200 MHz. These are the intended results of this practice. It can be checked how the output at both frequencies is different, which implies a malfunctioning of the encryption at 200 MHz.

C. Third session: Experimental verification of maximum operating frequency

The purpose of this laboratory session is to analyze the internal signals of the circuit in the board to verify the maximum operating frequency of the Trivium cipher. The Xilinx Chipscope tool will be used for this purpose.

Chipscope is a built-in logic analyzer. It allows the inclusion of test points within the FPGA to capture the values of sampled signals with a sampling signal. The sampled data are stored in a memory block of the FPGA device. They are then transferred to the computer and displayed on the screen.

This session is based on the latest design from a previous work. This design includes an MMCM module to change the frequency of the clock signal.

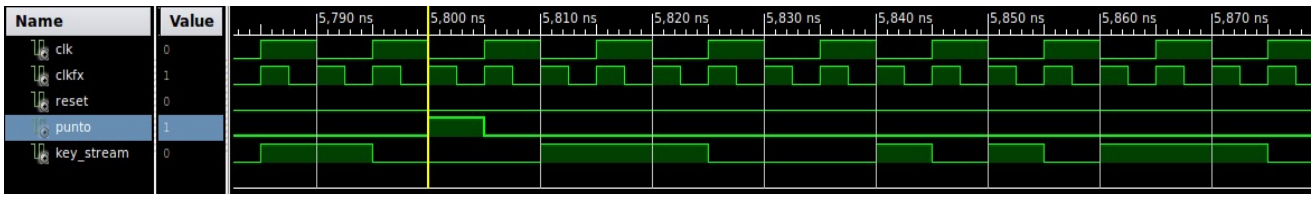
A Chipscope module must be added to the design. A new source of the type "ChipScope Definition and Connection File" is added for this purpose. The configuration of this module consists basically of selecting the signals to be sampled, the sampling signal and the number of samples to be stored. All this configuration is done through a graphical environment.

Once the Chipscope module is included, the steps to follow are:

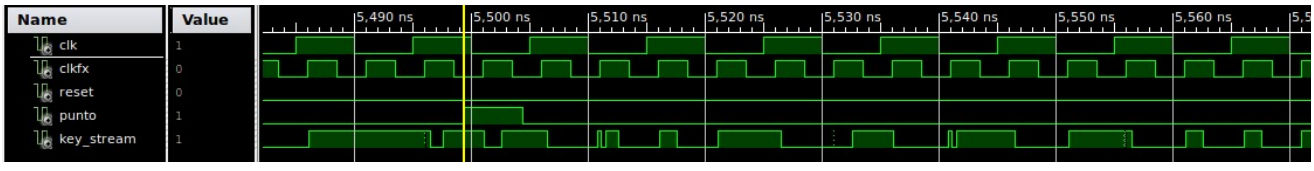
- Programming and verification with Chipscope

Programming is done within the same Xilinx ISE tool using the "Analyze Design Using Chipscope" option. This program opens a new graphical interface in which you can configure the signals to be displayed and the trigger mode.

The trigger is set to occur when the point output is set to'1'. The function verification is done by comparing the results obtained in functional simulation with those obtained by means of Chipscope. To analyze the behavior at different frequencies, the MCMM module changes the frequency of the output clock and programs the FPGA. In Chipscope, the output is analyzed from the point signal set to '1'. By comparing the outputs observed in the functional simulation, it is possible to know if



(a)



(b)

Fig. 2. Captures of performance (a) functional, (b) Post-route at 200MHz.

the operation is correct or not.

"Fig. 3" shows the results of the functional simulation and the outputs captured by Chipscope at 500 MHz and 600 MHz. It can be seen that at 500 MHz the output is equal to that generated by functional simulation, while at 600 MHz the output is completely different. This lead to the conclusion that the maximum operating frequency is between 500 and 600 MHz.

An important point is the comparison between the experimental results with those obtained by simulation. Experimentally, the maximum operating frequencies are much higher than those obtained through post-route simulation.

IV. Conclusions

In this article we have proposed a set of three laboratory sessions that use VHDL designs implemented in FPGA to teach students aspects related to the generation of clocks in an FPGA and the experimental calculation of the maximum operating frequency.

The circuit chosen is simple enough to operate at a higher frequency than the input clock, so an internal clock signal generation block is used to generate a clock frequency higher than the input signal frequency.

The results are very satisfactory and impressive for the students, as they prove that externally the signals can only reach a very low frequency compared to the maximum frequency at which they operate internally.

Acknowledgements

This work has been partially supported by the CESAR (TEC2013-45523-R), INTERVALO (TEC2016-80549-R) and LACRE (CSIC 201550E039) projects.

References

[1] The teaching project can be consulted at the link: http://www.us.es/estudios/grados/plan_201/asignatura_2010034 (accessed in February 2018).

[2] The teaching project can be consulted at the link: http://www.us.es/estudios/grados/plan_201/asignatura_2010011 (accessed in February 2018).

[3] The teaching project can be consulted at the link: http://www.us.es/estudios/grados/plan_201/asignatura_2010018 (accessed in February 2018).

[4] Julio Pastor Mendoza, José Manuel Villadangos Carrizo, Francisco Javier Rodríguez Sánchez, "Experiencias en el aprendizaje basado en proyectos del diseño de sistemas empotrados", TAEE 2016, Sevilla 22-24 de junio.

[5] Alfredo Rosado Muñoz, Manuel Bataller Mompeán y Juan Fco. Guerrero Martínez, "Aprendizaje por Proyectos: Una Aproximación Docente al Diseño Digital Basado en VHDL", Revista Iberoamericana de Tecnologías del Aprendizaje (IEEE-RITA), Vol. 3, Nº 2, Noviembre 2008, pp. 87-95.

[6] C. De Cannière, "Trivium: A stream cipher construction inspired by block cipher design principles," in Proc. Int. Conf. Inf. Secur., 2006, pp. 171–186, doi: 10.1007/11836810_13.

[7] eSTREAM: The ECRYPT Stream Cipher Project. Accessed: Jul. 2017.[Online]. Available: http://www.ecrypt.eu.org/stream/ (accedido en febrero 2018).

Fig. 3. Performance screenshots: (a) functional, (b) correct operation observed with Chipscope at 500 MHz, (c) incorrect operation observed with Chipscope at 600 MHz