Maximum Operating Frequency Self-Tuning System on FPGAs Using Dynamic Reconfiguration

Carlos, Fernández-García
Instituto de Microelectrónica de Sevilla
Seville, Spain
carlos@imse-cnm.csic.es

Aleksan Hakobyan
Universidad de Sevilla
Seville, Spain
alehak@us.es

Carlos J. Jiménez-Fernández
Insituto de Microelectrónica de Sevilla
Universidad de Sevilla - Escuela Politécnica Superior
Seville, Spain
cjesus@imse-cnm.csic.es

***Abstract*—This paper proposes the use of a dynamic clock frequency reconfiguration technique to optimize the performance of a circuit by changing its clock frequency to achieve the maximum operating frequency. The proposed technique uses an FPGA reconfigurable c lock g enerator c ircuit t hat changes the generated clock frequency for a circuit in real time. By systematically increasing the clock frequency and monitoring the response of the circuit, the real maximum operating frequency can be determined. The effectiveness of the proposed technique is demonstrated through simulation and experimental results with the development of an experimental system. The results show that it can accurately determine the maximum operating frequency of a circuit while maintaining its reliability and integrity.**

***Index Terms*—Maximum operating frequency, Clock Managers, FPGA, timing analysis**

I. Introduction

The operating frequency of a circuit fundamentally affects two things: performance and power consumption. Dynamically modifying the operating frequency of a circuit can be a performance optimization tool, by raising the frequency to the maximum operating frequency when increased performance is required and lowering it when the circuit is not operating or high performance is not required, thus saving power consumption.

The timing analysis of the maximum operating frequency has been a long-standing issue that has been addressed by various proposed solutions for several years [1]. However, despite the common use of timing analysis tools such as static analysis and temporal simulation tools, the model's accuracy is not too precise and often leads to conservative outcomes. As such, there have been recent proposals to enhance and refine t he c urrent a nalysis m ethods [2] [3].

In nanometer technologies, the timing analysis problem becomes even more complicated, as the delay calculation of logic gates and bistables are compounded by the increasing influence o f i nterconnect d elays a nd a ll t he p roblems associated with signal noise such as crosstalk [4]. The timing analysis tools included in commercial microelectronic design tools do not usually include precise methods to determine the impact of these effects in these latest nanometer technologies, providing very conservative results at maximum operating frequencies in a way that guarantees performance. Thus being an experimental approach a good alternative to detect the real maximum operating frequency.

FPGA devices incorporate elements for the internal generation of clock signals whose frequency, lower or higher than the input clock frequency, is set when the circuit is being designed. However, the latest FPGA device families incorporate mechanisms for dynamic control of the output clock frequency.

In this paper, we present an automatic and embedded self-tuning mechanism to get a circuit implemented in an FPGA to work at the maximum operating frequency. The mechanism is based on the use of the Mixed-Mode Clock Manager (MMCM) of a Xilinx 7-series FPGA [5] and on the dynamic configuration of the frequency through the DRP (Dynamic Reconfiguration Port) [6]. The tests have been carried out on a Xilinx Artix-7 family device, so its use can be extended to any device of this family.

This paper is organized as follows. Section II discusses the main characteristics of the Xilinx 7-series MMCM and the proposed mechanism to work at maximum frequency. Section III presents a circuit developed to test the proposed mechanism, Section IV explains how the circuit presented in III works and Section V presents and compares the reports from the main temporal analysis tools and experimental results. Finally in Section VI, some conclusions are drawn.

II. Xilinx 7 Series FPGA Clocking Resources

FPGA devices have specific resources to handle clock signals. The fundamental ones are the global lines and the MMCM.

Global clock lines allow the clock signal to reach many flip-flops with low jitter and low skew. MMCMs allow up to 6 clock signals to be generated from one clock, at both lower and higher frequencies. To synthesize these frequencies, the MMCM uses the basic blocks that make up a PLL as well as clock dividers that allow the designer to select the frequency.

There are three dividers (D, O, and M), which are used to generate the multiplication factor of the input MMCM clock frequency, thus achieving a wide range of output frequencies. The expression 1 shows the output frequency of the MMCM as a function of the divisors already discussed.

$$f_{out} = f_{in} \frac{M}{D \cdot O} \tag{1}$$

As mentioned in the previous section, the value of these dividers can be configured dynamically using the DRP, the configuration ports are highlighted in red in Figure 1 which shows the block of an MMCM of the FPGA family 7 of the manufacturer Xilinx in which only the main signals of the module and the ports necessary for the dynamic configuration are shown.

The DADDR signal consists of a 7-bit address bus that allows access to the registers of the different configuration groups of the MMCM. The input data bus consists of a 16-bit bus that allows the writing of the control registers of the different groups, provided that WE (Write Enable) signal indicates this with a logic '1'. The DEN (Data Enable) input is an enable input that must remain at '1' during write or read operations. DCLK is the clock signal used for writing or reading the DRP. There are also two output signals DO and DRDY, which are used as a 16-bit output data bus for reading the register indicated by the DADDR input and to indicate the validity of this output data respectively.

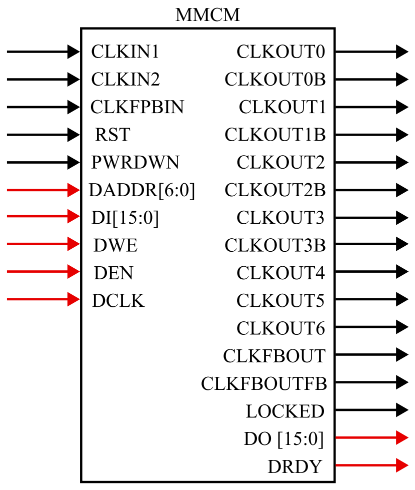

Fig. 1. MMCM Block.

The DRP allows access to 6 configuration groups for selecting the frequency of each of the outputs. These groups are divided into 16-bit registers, grouped into Divider groups used to select the dividers value, Phase groups for phase configuration, and Fractional groups used to configure the division using non-integer values. In addition, there are 3 independent groups called Lock group, Filter group, and Power group. It should be noted that the Divider group is available for both the input clock divider and the output clock divider.

III. Dynamic Reconfiguration Technique

In order to determine the maximum operating frequency of a digital circuit during operation, it is necessary to use, in addition to the necessary MMCM modules, a control logic that is responsible for the configuration of these modules, as well as the detection of a correct/incorrect operation of the digital circuit on which the frequency variation is applied. The block diagram of the proposed scheme is shown in figure 2.

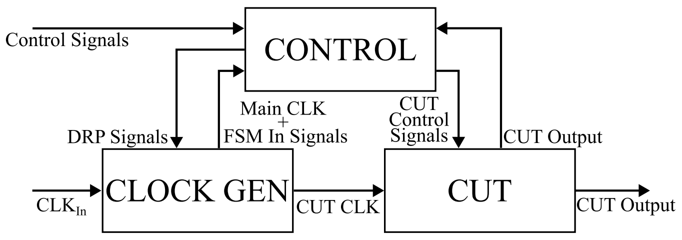

Fig. 2. System Block Diagram.

A. Clock Generation Block

Two MMCMs are used for clock generation as shown in Figure 3. The first one is used to generate a main clock which will be used by the controller block in Figure 2, while the second MMCM will be used to vary the frequency of the digital circuit by means of the dynamic reconfiguration ports that will be stimulated by the controller.

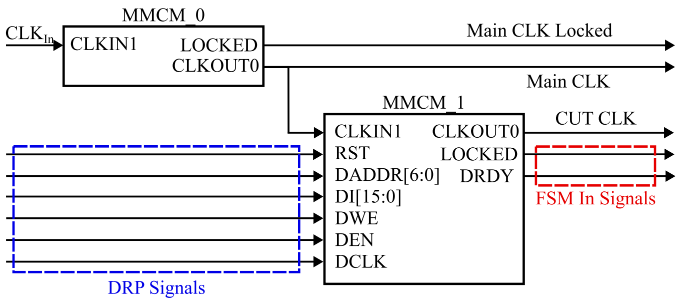

Fig. 3. Clock Generation Scheme.

B. Controller Block

As mentioned above, the controller is in charge of frequency configuration and control of the digital circuit.

The logic of the controller consists of a state machine that first of all takes care of the desired frequency configuration of the digital circuit by writing the corresponding data in the registers of the output clock divider group as well as the input clock divider and feedback. Once the desired frequency has been set, it is necessary to wait for the frequency to stabilize, which is indicated by the LOCKED signal of the second MMCM of the clock generation block, before the digital circuit begins operation.

Once a clock frequency has been dynamically set, it will perform a test run of the circuit under test (CUT). In case the circuit works correctly, the clock frequency will be reconfigured to be slightly increased. The process will continue until a malfunction is detected in the CUT. In that case, the frequency will be reconfigured to the last frequency at which it had been operating correctly.

C. *Circuit Under Test*

The CUT shall have two operating modes: a normal mode and a clock setup mode. In normal mode, the CUT receives as input signals the signals coming from the system in which it is inserted, but in clock setup mode, the CUT's input signals shall come from the control block. Only in this operating mode will the CUT checks if the outputs are as expected and decides whether to change the clock frequency or not.

IV. Dynamic Reconfiguration Technique Application Example

In order to test the validity of the proposed mechanism, a test design was carried out. In this application example, a hardware implementation of the AES (Advanced Encryption Standard) symmetric encryption standard using a 128-bit key has been selected as CUT. The implemented design performs each round of encryption in one clock cycle, so, counting the load and output generation cycles, 11 clock cycles are required to generate a valid output. The logical operations to be performed in each round (SubBytes, ShitftRows, MixColumns, and AddRoundKey) are combinational operations and will predictably be the operations that limit the maximum clock frequency. The control block of the developed example dynamically configures the AES clock frequency and performs the encryption of a default plaintext using also a default key. It checks the result of the data encrypted by the AES with the expected output also stored in the control circuit itself.

Figure 4 shows a schematic of the operations performed by the controller to correctly adjust the operating frequency of the circuit.

The first necessary step is to configure the power group using the dynamic reconfiguration ports. This group must always be configured with the same value (0xFFFF) to ensure the correct operation of the MMCM. Once this port has been configured, wait for the ready indication (DRDY) of the MMCM to continue with the frequency configuration. Note that every time you write through the DRP in some of the corresponding registers, you must wait for the DRDY signal to be asserted high before continuing.

After the configuration of the power group, the output signal and the feedback signal frequencies are set by writing the divider group. This configuration requires the writing of the two registers that make up the divider group, the first or less significant one is used for the configuration of the divisor value, while the second one is used, in this case, mainly so that the generated signal has a duty cycle of 50%. In the case of the divider group configuration of the input clock, only the first and only available register is modified, which allows the configuration of both the division and the duty cycle.

Note that the values with which the corresponding registers are updated are stored in a ROM, and provide a 10 % frequency increase in each iteration from an initial frequency of 100 MHz. This is mainly because not all combinations of the divisors D, O, and M are possible and it is easier to store these values than to implement their calculation within the control block.

Once the desired frequency is set, the MMCM waits for the frequency to be generated correctly (indicated by the LOCKED signal as depicted in the previous section), after which the CUT operation is started.

V. Results Analysis and Comparison

To evaluate the behavior of the developed application example, the whole system has been implemented on a Xilinx FPGA of the Artix-7 family, specifically on the xc7a100tcsg324-1 device using the Xilinx ISE software and the VHDL hardware description language to describe the auto-tuning system as well as the AES used as the circuit under test. In order to know the maximum operating frequency, the ROM memory address containing the programming coefficients of the output clock frequency has been set as the output of the experimental system. This address value is only updated in the case of a successful operation. In this way, the maximum frequency obtained can be easily checked and compared with the estimates of the time analysis tools.

Before performing any experimental analysis, the system has been simulated to check its correct operation, and timing analyses have been carried out to estimate the maximum operating frequency. In order to obtain an estimated maximum operating frequency we have performed post-route simulations as a dynamic timing analysis tool, and we have revised the static timing analysis reports for the static timing evaluation. We then implemented our FPGA-based maximum operating frequency detection system and check the maximum operating frequency.

A. *Simulation and Timing Analysis Results*

The first analysis performed is that of the correct dynamic configuration of the CUT clock frequency. In this case, a functional simulation is appropriate. Table I shows a screenshot of

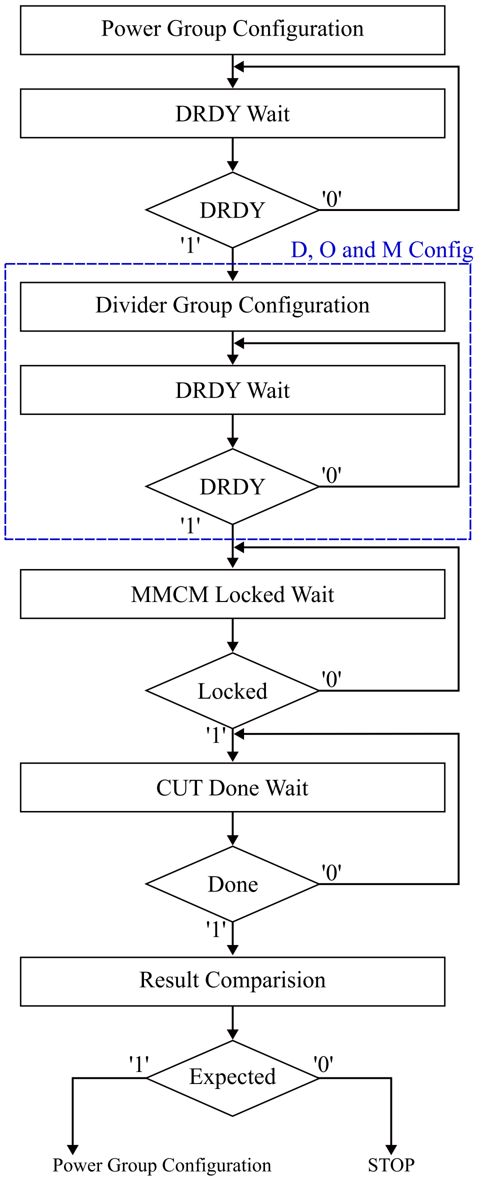

Fig. 4. Controller FSM.

the different frequencies synthesized by the MMCM for all the ROM addresses where the values of the different dividers are stored.

The second analysis performed is a post-route simulation in which the CUT clock frequency is increased until a failure in the AES operation is observed. Figure 5 shows a snapshot of the simulation waveform. The o_freqcnt signal is the one that indicates the address of the ROM of dividers that has been programmed in the MMCM and the o_fail signal is the one that indicates if there has been a failure in the operation of the CUT. As can be seen in the snapshot, the failure occurs for an address of ”0010” which means a frequency of 120 MHz. The static timing analysis reports have also been reviewed, examining the paths with the greatest delays within the AES.

TABLE I
FREQUENCIES SYNTHESIZED BY THE MMCM DEPENDING ON THE ROM ADDRESSED BY THE CONTROL BLOCK

ROM Memory Address	Frequency	Generated Clock Waveform
0000	100 MHz	
0001	110 MHz	
0010	120 MHz	
0011	130 MHz	
0100	140 MHz	
0101	150 MHz	
0110	160 MHz	
0111	170 MHz	
1000	180 MHz	
1001	190 MHz	

For the AES core used as CUT, the static analysis tool reported a minimum period of 8.868 ns which means a maximum frequency of 112.765 MHz. It can be seen that the maximum operating frequency results provided by the post-route simulation and by the static analysis tool are similar and consistent. In the post-route simulation, the correct operation is verified at 110 MHz but not at 120 MHz, and in the static analysis the estimated maximum frequency is 112 MHz.

B. Experimental Results

The complete system has been implemented in Digilent's Nexys-4 DDR (Figure 6) development board. In order to observe the maximum operating frequency, the outputs o_freqcnt have been mapped to the four least significant LEDs on the board. In this way, the maximum operating frequency of the CUT can be observed in a very simple way.

The experimental results show a binary value of ”0111” at the o_freqcnt output, which corresponds to a maximum operating frequency of 170 MHz. This maximum operating frequency value is significantly higher than that estimated by the temporal analysis. This discrepancy of values is not strange, because as already commented in the introduction of the article, the time analysis tools are not very accurate and thus tend to give values quite conservative and thus guarantee the correct operation at those frequencies.

VI. CONCLUSIONS

In this paper, we have presented a clock frequency dynamic reconfiguration self-tuning technique for detecting the maximum operating frequency of a circuit under test. The developed system has been implemented in a Xilinx Series-7 FPGA using MMCM blocks in order to increase the CUT clock frequency until a failure occurs, which in turn allows us to identify the maximum operating frequency. The developed system has, in addition to an MMCM to generate the clock signal and the CUT, a control block that is dedicated to perform the dynamic reconfiguration of the clock frequency

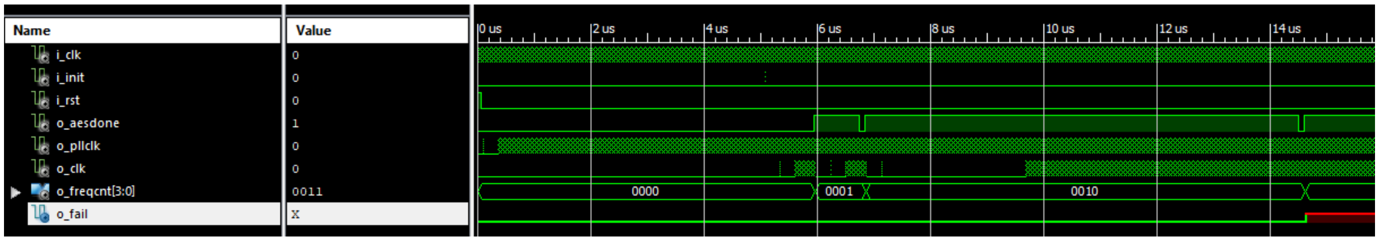

Fig. 5. Timing Violation in Post-Route Simulation Snapshot

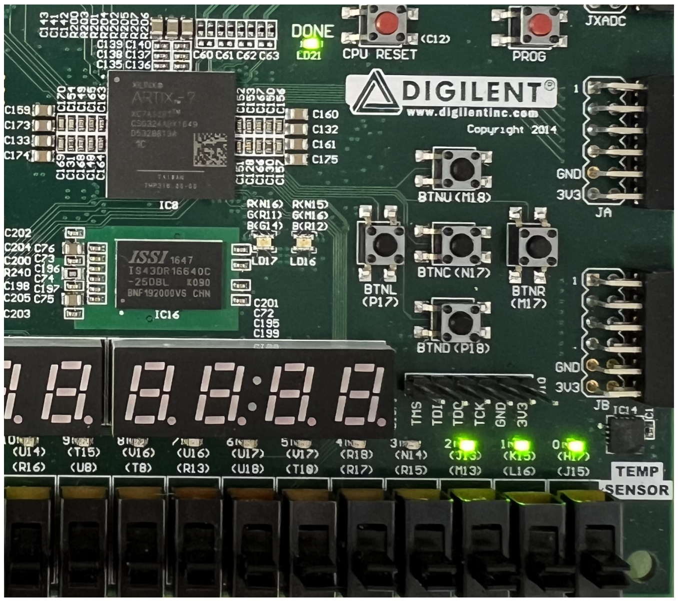

Fig. 6. Picture of the demo running in the Nexys4-DDR board used for the experimental test

and to analyze if the CUT operation is correct or not. The developed system uses a hardware implementation of the AES symmetric cipher as CUT. This circuit is a complex enough to detect differences between the results given by the timing analysis tools and the actual implementation of the circuit in an FPGA.

Static timing analysis indicates a maximum operating frequency of 110 MHz, while the experimental maximum operating frequency obtained is 170 MHz. These results show that the maximum frequency of the FPGA implementation is higher than that predicted by simulation and static timing analysis tools. This highlights the fact that the performance of a circuit's silicon implementation may actually be superior to the performance estimated by the different synthesis and place & route tools, and therefore this self-tuning technique could be useful to reach the implemented circuit's maximum operating frequency in order to achieve the best possible performance of the circuit.

Therefore changing the clock frequency by dynamic reconfiguration in FPGA can be a powerful tool to optimize circuit performance.

Acknowledgment

This work has been funded by SPIRS (Secure Platform for ICT Systems Rooted at the Silicon Manufacturing Process) Project with Grant Agreement No. 952622 under the European Union's Horizon 2020 research and innovation programme. Authors want to thank Programa Operativo FEDER 2014-2020 and Consejería de Economía, Conocimiento, Empresas y Universidad de la Junta de Andalucía under Project US-1380823, and project Grant PID2020-116664RB-I00 funded by MCIN/AEI/10.13039/501100011033.

References

[1] R. B. Hitchcock, "Timing verification and the timing analysis program," in *Papers on Twenty-Five Years of Electronic Design Automation*, 1988, p. 446–456.
[2] D. Garyfallou, I. Tsiokanos, N. Evmorfopoulos, G. Stamoulis, and G. Karakonstantis, "Accurate estimation of dynamic timing slacks using event-driven simulation," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, 2020, pp. 225–230.
[3] M. Fathi, T. Martin, G. Grewal, and S. Areibi, "Using machine learning to predict operating frequency during placement in fpga designs," in *2021 International Conference on Microelectronics (ICM)*, 2021, pp. 53–56.
[4] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*, 1st ed. Springer Publishing Company, Incorporated, 2009.
[5] "7 Series FPGAs Clocking Resources User Guide," https://docs.xilinx.com/v/u/en-US/ug472_7Series_Clocking, Jun 2018, accessed may 2023.
[6] "MMCM and PLL Dynamic Reconfiguration Application Note," https://docs.xilinx.com/v/u/en-US/xapp888_7Series_DynamicRecon, Aug 2019, accessed may 2023.