Indonesian Journal of Electrical Engineering and Computer Science
Vol. 14, No. 2, May 2019, pp. 637~645
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v14.i2.pp637-645

Manual clock distribution technique in partitioning stage for multi-FPGA prototyping

Salahuddin Savugathali, Muslim Mustapa, Fazrul Faiz Zakaria
School of Computer and Communication Engineering, Univeristy Malaysia Perlis, Malaysia

Article Info

Article history:

Received Aug 27, 2018
Revised Nov 1, 2018
Accepted Jan 20, 2019

Keywords:

Clock distribution
Clock replication
Cut clocks
Cut nets
FPGA
Multi-FPGA prototyping
Routing congestion

ABSTRACT

As the complexity of ASIC/SoC design is increasing along with the number of logic gates, a prototyping process in the verification stage is facing a challenge when the ASIC/SoC design cannot fit into a single FPGA. A solution to prototyping multi-million logic gates of ASIC/SoC circuit into the FPGA platform for verification purpose is by partition the design into multi-FPGA. There are various implementation tools and platform available in the market which automates an FPGA-based prototype phase such as Cadence Protium Rapid Prototyping Platform, Synopsys and S2C. In this paper, Synopsys protocompiler tool will be used to perform the prototyping process of the large 4 core CPU based circuit into the HAPS-80 FPGA platform. This paper will be focusing on the partition requirement needed to successfully prototype the large SoC circuit into the multi-FPGA. The presence of cut clocks in a circuit after partition stage will resulting to the failure in routing stage due to the congestion error. In this paper, two techniques are used, which is automatic clock replication by the Synopsys Protocompiler tool and our proposed technique which is Manual Clock Distribution technique to solve the presence of the cut clock, so that the circuit is able to meet the partition requirement to complete the prototyping process into multi-FPGA. Obtained result from the proposed technique showing that prototyping the large SoC circuit into the multi-FPGA platform has met the specification by eliminating 100% presence of cut clock.

Corresponding Author:

Salahuddin Savugathali,
School of Computer and Communication Engineering,
University Malaysia Perlis, Malaysia.
Email: salahuddin@studentmail.unimap.edu.my

1. INTRODUCTION

Nowadays, more resolution is required for the verification of continuous enhanced integrated design technologies which leads to more complex and high-performance SoC design which is currently become a main electronic design in semiconductor industries[1]. Design verification is an end-stage process of ensuring that everything on the integrated design works as planned [2] and meets the customer requirement.According to International Business Strategic (IBS) [3], cost of developing a full design showing that verification stage is expanding at an aggressive rate and has the highest rate among the overall design cost. Due to tremendous increases in size, complexity, and cost of SoC designs consumed in verification stage, the software developer can no longer wait for the chip to be fabricated for the integration of the hardware/software phase in order to meet the ever-shrinking time-to-market window. Therefore, FPGA-based prototyping technique is used to address these challenges by prototyping a SoC circuit on an FPGA so that, the design can be verified in a pre-silicon stage [4]. Multiple FPGA-based protoyping is an option to for the large design verification due to its high execution speed[5] ,[6] but require additional effort to partition in an optimized way [7].

Figure 1 shows a multi-FPGA prototyping flow using the Synopsys Protocompiler tool. ASIC/SoC design's RTL must be reworked to meet the FPGA based requirement as follow; top-level pads required to be adapted for the FPGA tool flow, gated-clock and complex generated clocks in SoC/ASIC must be transformed in FPGAs and memories required to be handled with FPGA memory resources. Once the circuit is converted into the FPGA-based, a compilation stage will take place before went through the pre-partition, partition, system route and system generate. Once the FPGA-based SoC circuit is successfully partitioned, the synthesis process will take place for each partitioned FPGA which begins with compile, and continue with the pre-map, map, and place and route using a Vivado tool.

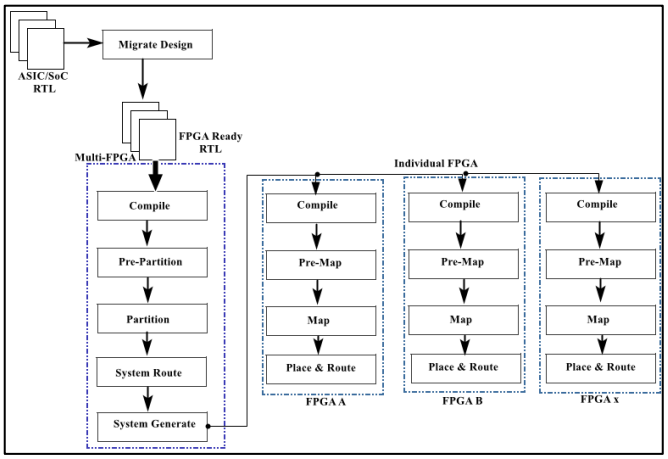


Figure 1. Multi-FPGA prototyping flow

In Multi-FPGA prototyping flow, the higher routing congestion which is more than the maximum congestion level which is set to be at level 4, will fail the FPGA partitioned design [8]. A larger SoC/ASIC design must be partitioned into the multiple FPGA before the routing stage is visited [9]. During the FPGA partitioning stage, there are few partition requirements must be achieved which is; zero unrouted nets, zero cut clocks, low number of feedthrough, low number of multi-hop nets, the minimum number of FPGA interconnect nets, and each of the FPGA utilization must be less than 65% [8]. Therefore, in this paper, work has been devoted to satisfy the partition requirement to avoid the routing congestion problem.

This paper is organized as follow; Section 2 describes an existing technique to solve the challenges in multi-FPGA prototyping flow. While Section 3, discusses the step implemented in this paper from the analyzation of the failed design until the implementation of the approached techniques. Section 4 will summarize the approached techniques with the obtained result.

2. LITERATURE REVIEW

There are plenty of research have been done before to address the multi-FPGA prototyping flow challenges. This paper will focus on satisfying the partition requirements to avoid the routing congestion. In this section, all the previous work devoted to multi-FPGA flow will be discussed further.

Due to the large design which is not fitted in a single FPGA, a multi-FPGA prototyping flow is required to split the design into multi-FPGA. Most of the common problem during multi-FPGA prototyping is faced by the designer in the routing stage [10]. High congestion within inter-FPGA signal caused the design to be un-routable [11].

A study in [10] has identified that due to the inter-FPGA communications, the system frequency of the prototyped design is decreased and also, the number of inter-FPGA signals and critical path delay affected by the approach of partition technique. Therefore, the partitioner tool has been constrained in order to allow the automatic design partition by a tradeoff between criteria that affects the system frequency. An iterative routing algorithm is used to route the inter-FPGA signal and, for exceeding signal cases the multiplexing IPs are implemented in sending and receiving FPGA to transmit the signal through the same physical wire. In [12], a new approach is proposed using routability-driven routing approach which gives a better result than approach used in [10]. In [13], a proposed iterative routing algorithm in [14] is enhanced to route multi-terminal nets in multi-point tracks for routing cut nets in two point track by saving the FPGA

input/output (I/O). While, in [15], author has present partitioning method based on topological ordering and levelization. Result obtained from the the experiment is able to emphasis the performance of the design, however to avoid to focus on this thesis, we are solving the problem which is causing the design are not be partitioned due to the congestion error.

In this paper, an automatic clock replication technique by using Synopsys Protocompiler will be applied in order to avoid the routing congestion by eliminate the cut clocks. Another technique applied in this paper is by manually redistributing the clock to other FPGA through HAPS global clock network manually on partition constraint file (PCF)[8]. Both of these techniques will be compared on the capability to meet the partition requirement to solve the cut nets which is causing the design to be un-routable.

This paper is organized as follow; Section 2 discusses the step implemented in this paper from the analyzation of the failed design until the implementation of the approached techniques. Section 3 will summarize the approached techniques with the obtained result.

3. RESEARCH METHODOLOGY

In this section, all the precaution taken in order to satisfy the partition requirement in multi-FPGA prototyping flow will be discussed. A design with a multi-million logic gates will be used in this paper.

3.1. Implementation Step

Based on the multi-FPGA prototyping flow shown in Figure 1, a large design which is consist of 4 core CPU based circuit named as design_3 will be partitioned into the multiple FPGA before synthesizing it in the individual FPGA stage. Most of the challenges faced in the multi-FPGA prototyping flow are in the partition stage as to meet the partition requirement. Therefore, this research work has been done on the implementation steps to meet the partition requirement. Figure 2 shows the steps taken to partition a design into multi-FPGA.

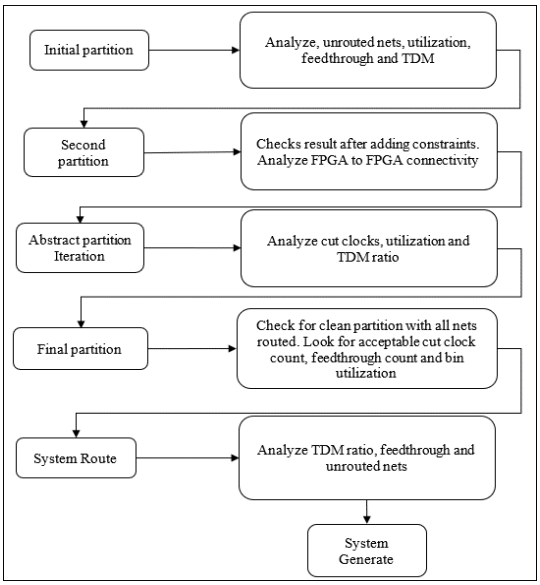


Figure 2. Methodology for partition

Before partitioning a design, 2 input files have been prepared which is TSS file describing the hardware setup and partition constraint file (PCF) describing a partitioning constraint. The Protocompiler tool will then partition a design according to the constraint defined in the PCF. Results of each partition iteration will be analyzed to ensure the partition requirement is met.

In the initial stage, a run has been executed with default setting on the partition requirement where the Protocompiler tool will automate the partition stage before going to the next stage, system route. However, the design is not able to pass the routing stage as high congestion in an FPGA which cause the design is un-routable. The result from the partition report is showing failing partition stage because the partition requirements are not met. Therefore, an iterative process is required to refine the partition result until it is satisfied.

Referring to Figure 2, in the initial partition stage, the cable connectivity among the FPGA will be defined first in the partition constraint file (PCF) before analyzing the nets in the partition report. Global Route Summary section in the partition report has been visited to ensure there is no un-routed net is reported. If the un-routed net is reported then top-level port assignment, clock constraints and cell assignment constraints need to be checked.

Figure 3 shows the first challenge faced in this research where lookup table (LUT) utilization in the FPGA A is high compared to FPGA D. The utilization of LUT for each FPGA should be below than 65% [8]. Figure 4 shows the partition requirement to be satisfied in the Global Route Summary section.

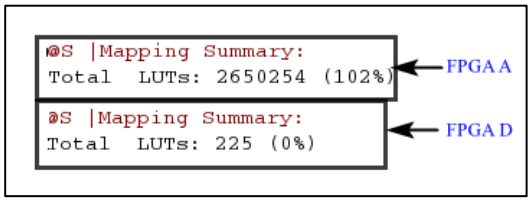


Figure 3. Utilization report for FPGA A and FPGA D

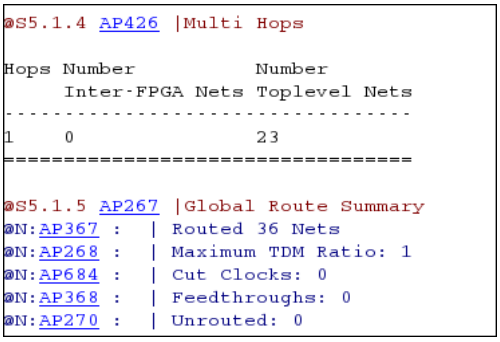


Figure 4. Global route summary

It is shown that all partition requirements is satisfied except the total LUT utilization. Therefore, utilization for all bins has been divided manually by the number of the core in the design_3 through the PCF file. The 4 core CPU based design is divided by 2 cores for each FPGA and the main bus of the design is placed in the FPGA A using a PCF command. After the design is manually repartitioned again according to the new PCF constraint, then the utilization problem is solved, but another problem arises due to the existence of the cut clocks as per shown in Figure 5. Figure 6 shows the detail of existing cut clock where the clock is crossing FPGA boundaries and causing a clock skew problem which will result in low timing performance in the design.

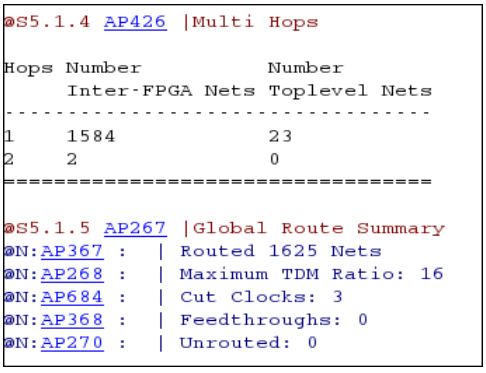


Figure 5. Global route summary after for the second partition

```
@S4.2.3 AP408 |Partitioner Estimate of Clock & Asynchronous Reset Crossings
  Cut Clock: FB1.uA->{FB1.uD} smca.cpu0.ckmclksrccore[3]
  Cut Clock: FB1.uA->{FB1.uD} smca.cpu0.ckatpgcfgshift_core_t1n22[0]
  Cut Clock: FB1.uA->{FB1.uD} smca.cpu0.ckmclksrccore[2]
```

Figure 6. Cut clock detail

Therefore, Protocompiler tool's feature is utilized for this issue by enabling the automatic clock replication on a specific clock tree of the problematic clock through the PCF. This feature able to solve cut clock in the partitioned design. However, this feature introduces an un-routed net between both FPGA as shown in Figure 7. Thus, alternative technique used in this research to avoid the cut clock problem, is by redistributing the clock to other FPGA through HAPS global clock network manually on PCF.

```
@S6.1.5 AP267 |Global Route Summary
@N:AP367 :   | Routed 1622 Nets
@N:AP268 :   | Maximum TDM Ratio: 16
@N:AP684 :   | Cut Clocks: 0
@N:AP368 :   | Feedthroughs: 0
@N:AP270 :   | Unrouted: 3
   FB1.uA -> FB1.uD 3
```

Figure 7. Unrouted nets is reported

Fifure 8 shows an example of illustration for the cut clock occurred in the design during an FPGA partition stage where a clock from FPGA A is crossed boundary and connect with FPGA D. There are two cut clocks is shown in Figure 8 which is CLK1 and Gated-CLK1 has crossed the FPGA boundaries from FPGA A to FPGA D.

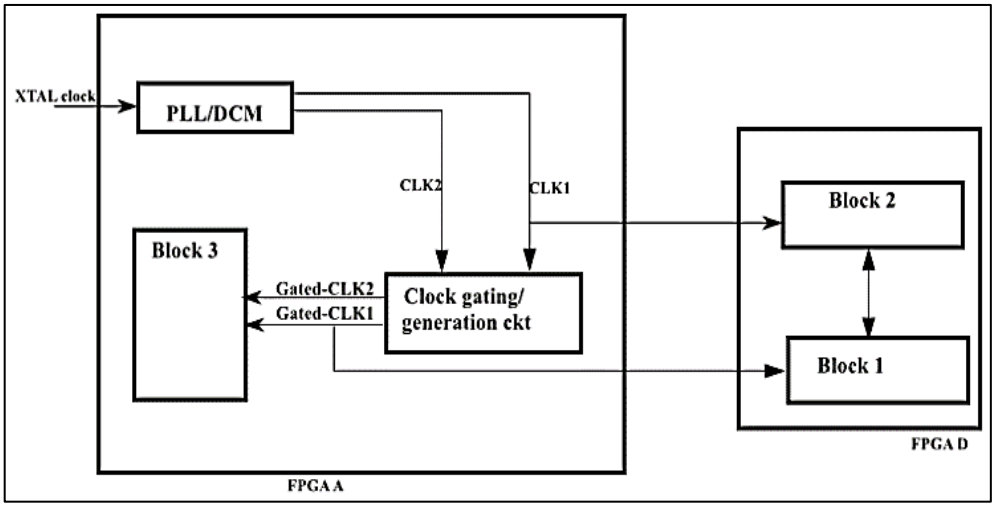


Figure 8. Illustration of the cut clock in a design

Figure 9 represents a clock distribution technique for the cut clock issue in the design which is illustrated in the example on Figure 8. HAPS-80 platform has a global clock net which is connected to every single FPGA. Therefore, a CLK1 which is crossing the FPGA A to FPGA D will be connected to CLKC_SRC[1] before connect back to the FPGA D through the GCLK[7].

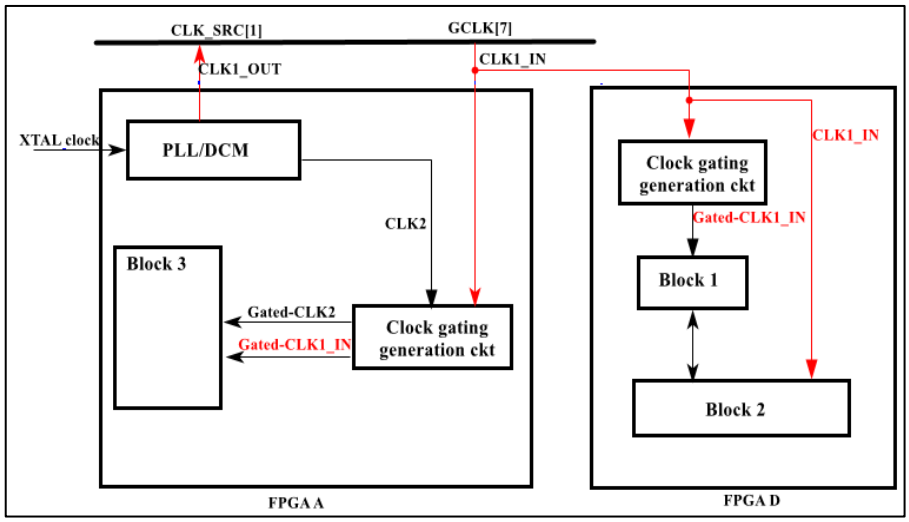


Figure 9. Solution for cut clock

Figure 10 shows a constraint defined in the TSS file to configure the GCLK to be used in the design to be prototyped. TSS is a hardware specification file, where all the nets, clock, and traces to be used is specified to configure the HAPS-80 platform. While Figure 11 and Figure 12 represent a constraint defined in a partition constraint file (PCF). After configuring the HAPS platform, a PCF file must be defined to partition the design according to the specification. Therefore, a problematic clock which is causing a cut clock issues in this design will be connected to the global clock to another FPGA as per defined in a PCF file.

```
#########################################################################
#configure your selected GCLK to source the input from the FPGA
#########################################################################
board_system_configure -clock FB1.GCLK7 FPGA
board_system_configure -clock FB1.GCLK8 FPGA
board_system_configure -clock FB1.GCLK9 FPGA
```

Figure 10. constraint defined for TSS file

```
#############################################################################################
#Define the functional group of internally generated clocks as GCLKs in the pcf
#############################################################################################
net_attribute {smca.cpu0.ckmclksrccore[3]} -function GCLK -diffsingle -is_clock1
net_attribute {smca.cpu0.ckmclksrccore[2]} -function GCLK -diffsingle -is_clock1
net_attribute {smca.cpu0.ckatpgcfgshift_core_t1n22[3]} -function GCLK -diffsingle -is_clock1
```

Figure 11. constraint defined in PCF file (1)

```
#########################################################################
#Assign the internally generated clock to GCLK in the pcf file
#########################################################################
assign_global_net {smca.cpu0.ckmclksrccore[3]} FB1.GCLK7
assign_global_net {smca.cpu0.ckmclksrccore[2]} FB1.GCLK8
assign_global_net {smca.cpu0.ckatpgcfgshift_core_t1n22[3]} FB1.GCLK9
```

Figure 12. constraint defined in PCF file (2)

The result obtained for all implementation step during satisfying the partition requirement will be discussed in next section.

4. RESULTS

For multi-FPGA prototyping, partition requirement is set as a benchmark to be met in order to successfully routing the design before synthesized it in an individual FPGA level. Among the partition requirements to be met are the number of unrouted nets, total cut clocks and feedthrough in a design should be zero. While multi-hop net which is a connection between FPGA-to-FPGA must be below than 3 and each FPGA utilization should not exceed 65%.

As the first run in an experiment, the default run has been executed without any fixes as the protocompiler tool is providing a feature to auto-partition the design. However, this run hit an error at partition stage as the requirement is achieved. Therefore, the generated partition report is revisited to check all the requirement for a successful partition.

4.1. The Result of Auto-Partitioning Using Protocompiler Tool

Table 1 shows the result obtained by using auto partition features using Protocompiler tool. Based on the recorded result in the table, all the requirement is met except for FPGA utilization. More than 100% average of the logic block is partitioned into an FPGA A while FPGA D is not utilized fully. Therefore, all other requirement is met since there is no any clock crossing the FPGA boundaries and not much multi-hop net is required.

Table 1. Result of Partition Requirement Using Auto-Partition

Partition Requirement	Result
Unrouted nets	0
Cut clocks	0
Feedthrough	0
Multi-hop nets	1
FPGA utilization	FPGA A = 2650254(102%) FPGA D = 225(0%)

4.2. The Result of Manual Partition on the First Iteration

Table 2 represents the result of partition requirement by application of manual partition through the constraint which is explained in the previous section. An FPGA utilization using manual partition has achieved less than 65% for each FPGA, however, another problem surfaced which is the cut clock now exist that causing the partition stage to fail.

Table 2. The Result of Partition Requirement Using Manual Partition (1st Iteration)

Partition Requirement	Expected value
Unrouted nets	0
Cut clocks	3
Feedthrough	0
Multi-hop nets	3
FPGA utilization	FPGA A = 1460430 (56%) FPGA D = 1143290 (44%)

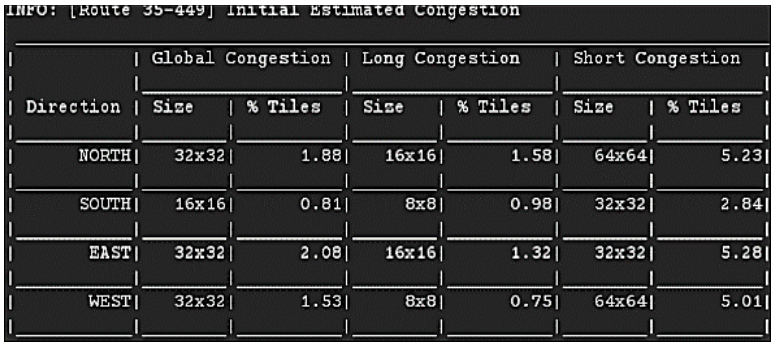
4.3. The Result of Manual Partition on the First Iteration

For the second iteration of partition stage, an automatic clock replication technique by specifying the clock tree has been applied using a protocompiler tool. Table 3 shows the result obtained for the second iteration where the existence of the cut clock has been eliminated but causing a more serious problem in the routing stage where the unrouted nets exist.

Table 3. The Result of Partition Requirement Using Automatic Clock Replication (2nd Iteration)

Partition Requirement	Expected value
Unrouted nets	3
Cut clocks	0
Feedthrough	0
Multi-hop nets	3
FPGA utilization	FPGA A = 1460430 (56%) FPGA D = 1143290 (44%)

Figure 13 shows the congestion area in an FPGA which is the reason of failure in routing stage. The maximum size of the congestion area should not exceed more than level 4. As seen in Figrue 13, two short congestion areas in NORTH and WEST has exceeded the maximum range with the size of level 5 and level 6.



INFO: [Route 35-449] Initial Estimated Congestion

	Global Congestion		Long Congestion		Short Congestion	
Direction	Size	% Tiles	Size	% Tiles	Size	% Tiles
NORTH	32x32	1.88	16x16	1.58	64x64	5.23
SOUTH	16x16	0.81	8x8	0.98	32x32	2.84
EAST	32x32	2.08	16x16	1.32	32x32	5.28
WEST	32x32	1.53	8x8	0.75	64x64	5.01

Figure 13. Routing congestion level

4.4. The Result of Manual Partition on the Final Iterationsub Section 1

Therefore, an automatic clock replication technique which is applied before is reverted back and newly proposed technique is used, which redistributes the clock among the FPGA trough HAPS global clock network. All implementation step for these techniques has been well explained in the Section 1.

Table 4 represent a final iteration of partition stage after using a proposed clock redistribution technique. Apart from achieving the partition requirement, the design also able to pass the routing stage before it is synthesized in individual FPGA level.

Table 4. Result of Partition Requirement Using Clock Distribution Techniques (Final Iteration)

Partition Requirement	Result
Unrouted nets	0
Cut clocks	0
Feedthrough	0
Multi-hop nets	3
FPGA utilization	FPGA A = 1460430 (56%)
	FPGA D = 1143290 (44%)

5. CONCLUSION

In this paper, two different techniques which is an automatic clock replication by the Synopsys Protocompiler tool and our proposed technique Manual Clock Distribution technique have been applied separately to eliminate the presence of the cut clock, so that the circuit is able to meet the partition requirement to complete the prototyping process into multi-FPGA. 4 core CPU based SoC design has been used. An existence of the cut clock has been fixed by using the Manual Clock Distribution technique by 100% elimination compared to the automatic clock replication technique which created an unrouted nets in the circuit. By using this technique, the circuit is able to pass the routing stage and prototyped into multiple FPGA accordingly.

REFERENCES

[1] X. Li, L. Hou, S. Geng, J. Wang, and H. Zhang, "The FPGA prototyping implementation of LEON3 SoC," *Proc. 2012 Int. Conf. Ind. Control Electron. Eng. ICICEE 2012*, pp. 1643–1646, 2012.
[2] A. Aboagye, M. Patel, and N. Vig, "Standing up to the semiconductor verification challenge," pp. 43–48, 2012.
[3] Q. Tang, M. P. Platform, G. Hardware, P. Et, M. Curie, and P. R. À. B. D. E. M. Ulti, "Methodology of Multi-FPGA Prototyping Platform Generation Qingshan Tang," 2016.
[4] Y. Abarbanel, E. Singerman, and M. Y. Vardi, "Validation of SoC Firmware-Hardware Flows," *Proc. 51st Annu. Des. Autom. Conf. Des. Autom. Conf. - DAC '14*, pp. 1–4, 2014.
[5] M. M. Azeem, R. Chotin-Avot, U. Farooq, M. Ravoson, and H. Mehrez, "Multiple FPGAs based prototyping and debugging with complete design flow," *Int. Des. Test Work.*, pp. 171–176, 2017.

[6] O. Melnikova, I. Hahanova, and K. Mostovaya, "Using multi-FPGA systems for ASIC prototyping," *CAD Syst. Microelectron. 2009 CADSM 2009 10th Int. Conf. Exp. Des. Appl.*, pp. 237–239, 2009.
[7] "Globally Optimal Time-Multiplexing In Inter-Fpga Connections For Accelerating Multi-Fpga Systems Masato Inagi Yasuhiro Takashima Yuichi Nakamura Dept . of Computer and Faculty of System IP Core Network Engineering Environmental Engineering Laboratories Hi," *Fpl*, pp. 212–217, 2009.
[8] "HAPS ProtoCompiler User Guide," no. September. Synopsys, Inc., 2016.
[9] Kevin Morris, "A Synthesis & Partitioning Strategy for Effective Multi-FPGA Prototyping – EEJournal," 2009.
[10] M. Turki, H. Mehrez, Z. Marrakchi, and M. Abid, "Partitioning constraints and signal routing approach for multi-FPGA prototyping platform," *Syst. Chip (SoC), 2013 Int. Symp.*, pp. 1–4, 2013.
[11] U. Farooq *et al.*, "Using Timing-Driven Inter-FPGA Routing for Multi-FPGA Prototyping Exploration," *2016 Euromicro Conf. Digit. Syst. Des.*, pp. 641–645, 2016.
[12] M. Inagi, Y. Takashima, and Y. Nakamura, "Globally optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA systems," in *2009 International Conference on Field Programmable Logic and Applications*, 2009, pp. 212–217.
[13] Q. Tang, H. Mehrez, and M. Tuna, "Routing algorithm for multi-FPGA based systems using multi-point physical tracks," *Proc. 2013 Int. Symp. Rapid Syst. Prototyp. Shortening Path from Specif. to Prototype, RSP 2013*, vol. 2, pp. 2–8, 2013.
[14] M. Turki, Z. Marrakchi, H. Mehrez, and M. Abid, "Iterative Routing Algorithm of Inter-FPGA Signals for Multi-FPGA Prototyping Platform TT -," *Lect. notes Comput. Sci. TA -*, no. 7806, pp. 210–217, 2013.
[15] N. Kerkiz, A. Elchouemi, and D. Bouldin, "Multi-FPGA partitioning method based on topological levelization," *J. Electr. Comput. Eng.*, vol. 2010, 2010.

BIOGRAPHIES OF AUTHORS

Salahuddin Savugathali received the B.Eng. degree in Computer Engineering from University Malaysia Perlis, Malaysia, and currently doing Master's degree Computer Engineering at University Malaysia Perlis, Malaysia. His research interest includes FPGA and software development.

Muslim Mustapa received the B.Eng. degree in electrical and electronics engineering from Universiti Teknologi Petronas, Perak, Malaysia, in 2007, the Master's degree in Electrical and Computer Systems from Monash University, Victoria, Australia, and the Ph.D. degree in electrical engineering from the University of Toledo, Toledo, OH, USA, in 2015. He is currently a Senior Lecturer with Universiti Malaysia Perlis. His research interests include hardware security, PUF, FPGA, smart grid, and advanced metering infrastructure.

Fazrul Faiz Zakaria received B.Eng. degree in Communication Engineering from University Malaysia Perlis, Malaysia, the Master's degree in Telecommunication and Electronic Engineering from TELECOM Bretagne, France.He is currently a Lecturer with Universiti Malaysia Perlis.