International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 4 Issue 4, April-2015, pp: (296-304), Impact Factor: 1.252, Available online at: www.erpublications.com

Design of Globally Asynchronous Locally Synchronous (GALS) System using FPGA

Rashmi Jain[1], Dinesh V. Padole[2], Mithun B. Kulkarni[3], Abhijeet[4], Amit D. Singhal[5]

[1,2,3,4,5]B.E. Department of Electronics Engineering, DYPIET, Pimpri, Pune, Maharashtra, India

Abstract: Globally Asynchronous Locally Synchronous (GALS) is a relatively new VLSI system design methodology that promises to combine the advantages of both synchronous and asynchronous designs. Through the proposed design, our aim is to bring forward the advantages of asynchronous design as much as possible. To draw comparisons, a general purpose 8-bit synchronous microprocessor was first designed and then converted into GALS processor. Both the models were implemented in VHDL using Xilinx ISE 13.3 software and simulated using ISim tool. The synthesis results show that under the same power consumption and a small area overhead, GALS processor outperformed the synchronous processor in terms of operating frequency which is approximately double the frequency of its synchronous version. Synchronous or clocked design is still by far the most popular digital system design methodology, well understood and supported by the well-known EDA tools.

Keywords: GALS, SOC, microprocessor, asynchronous processor, synchronous processor, plausible.

I. INTRODUCTION

GALS systems were first described by chapiro in 1984 who formulated a strategy to prevent metstability by stretching confined clocks. Modern digital systems are implemented as system-on-chips (SOCs), where different functional blocks have different clock requirements and operating frequencies. distributing a high frequency global clock to the entire system with low skew, is a challenging task demanding a lot of design effort, die area and power. As a result system integration has become a crucial problem. so, pure synchronous methodology doesn't look much promising to handle all design requirements.

Asynchronous methodology is shown to be a good candidate solution which can solve nearly all the problems related to clock. While asynchronous circuits potentially can have better performance, lower power consumption and more robustness, they can be larger due to handshaking overheads and they also suffer from design complexity and lack of automated cad tools. GALS methodology is an intermediate solution that combines the benefits of both synchronous and asynchronous methodologies. A GALS system consists of locally clocked synchronous modules i.e. each module can perform operations with its own clock and those modules are usually developed using standard synchronous cad tools. The communication between different synchronous modules can be achieved via asynchronous channels i.e. using handshaking approach.

II. RELATED WORK

Several researchers focus their work mainly to pausible clock based GALS system [4],[5] as it has been proven a promising approach to SoCs and Network-on-chips (NoCs). Currently, GALS applications mainly target the area of NoCs [6], [7], [8], multiprocessor systems [9] and integration of highly complex SoCs [10], [11]. Field-programmable gate array (FPGA) implementation of GALS is discussed in [12],[13]. Not many publications are available regarding design of GALS processors except [14] where the authors have attempted to build a GALS model of available synchronous superscalar processor architecture with a different partitioning strategy. Apart from this, [15] and [16] presents asynchronous design of 8051 microcontroller. The synchronous microprocessor we attempted to design is a simple general purpose processor and its GALS version is created by applying different partitioning strategy on the synchronous architecture.

III. THEORY

A. Synchronous Design:

Till date microprocessors available in market are all built using synchronous technology. In synchronous systems there is a centralized clock-pulse generator. Every part of the processor has to be connected to the same clock-signal.

This behaviour has some disadvantages.

1) Slowest component limits the clock rate: For example there is an operation which needs four gates and an operation which needs six gates the clock-signal is not allowed to be faster than the necessary time to pass six gates.
2) Energy consumption: All parts of a processor need energy even if there is nothing to do. At least the clock tree switches each clock.
3) High-frequency radiation: Higher clock rates also mean more high-frequency radiation. The synchronous design leads to a strengthening of the radiation.
4) Necessary chip space: Much chip space is needed by a clock generator. For higher frequencies the clock generator needs a quartz piezo-electric oscillator.
5) Long way for signals: The clock generator produces the clock signal, but this signal has to be delivered to each unit of the chip at the same time. Speed of light is limited and this can be a problem. The speed of light in a circuit is around 1 mm in100 ps. If the chip has a size of 30 mm than the needed time to traverse the chip is3 ns .This equals to 10 clocks at 3 GHz.

Despite the no. of disadvantages, designers continue to struggle with the synchronous methodology. The reason being this methodology is well understood and tool vendors (e.g. Cadence, Mentor Graphics, Synopsis etc.) provide a plethora of design environments that aid in design, verification, synthesis and testing.

B. Asynchronous Design:

Asynchronous or self-timed circuits work under the absence of global clock and the system timing is performed by the elements themselves. This has several possible benefits:

1) Lower power: Standard synchronous circuits have to toggle clock lines, and possibly pre-charge and discharge signals, in portions of a circuit unused in the current computation.
2) No clock skew: Clock skew is the difference in arrival times of the clock signal at unlike parts of the circuit. Since asynchronous circuits by classification have no globally distributed clock. In contrast, synchronous systems often slow down their circuits to accommodate the skew. As feature sizes decrease, clock skew becomes a much greater concern.
3) System Integration: With the asynchronous design methodology, the synchronous Intellectual Property (IP) core is completely decoupled from the connect bus. This makes it likely to integrate asynchronous communication with an existing synchronous system.
4) Easing of global timing issues: In systems such as asynchronous processor, the system clock, and thus system performance, is dictated by the slowest (essential) path. Thus, mainly portions of a circuit must be carefully optimized to achieve the highest clock rate, including seldom used portions of the system.
5) Better technology migration potential: Integrated circuits will often be implemented in several different technologies during their lifetime. Early systems may be implemented through gate arrays, while later on production runs may migrate to semi-custom or custom ICs.
6) Automatic adaptation to physical properties: The delay through a circuit can change with variations in manufacture, temperature, and power-supply voltage. Synchronous circuits must assume that the worst possible combination of factors is present and clock the system consequently. Several asynchronous circuits sense calculation completion, and will run as rapidly as the current physical properties allow.
7) Low Electro-magnetic Interference (EMI): The electromagnetic radiation than in synchronous processors because the pulsed electromagnetic radiation of millions of concurrently switching transistors of synchronous processors adds up to a large electromagnetic pulse.

IV. Overview of GALS design

In a GALS design, the clock skew constraints are reduced, since the clocking is no longer a global issue. GALS is therefore a promising technique for simplifying the design of large high performance soc design. In the GALS system design, the main issue is designing reliable GALS interfaces to handle synchronous and asynchronous logic domains.

In Gals design styles are divided into three categories:

1) Pausible or stretchable clock: In this design style a local clock generator is used, able to pause or stretch to prevent metstability during data transfer between different clock domains. Pausible clock is more or less similar to clock gating in synchronous circuits.
2) Asynchronous: This design style involves general case with no consideration of timing relationship between synchronous clocks.
3) Loosely synchronous: This design style assumes a known relationship between clocks, either in clock frequency or in the clock phase.

GALS basically combines synchronous design methodology with the asynchronous design style. The goal is to combine the advantages of the respective design styles while avoiding their shortcomings. GALS architecture (Fig.1) is composed of large synchronous blocks (SBs) which communicate asynchronously. By eliminating the global clock, a major source of power consumption and a design bottleneck is eliminated.

The advantages of GALS systems are:

1) The possibility to reuse the existing synchronous IP cores.
2) Easy to design systems with multiple clock domains.
3) No global clock distribution problem.
4) The employment of the standard synchronous EDA tools to design and verify new IP cores.
5) The ability to run SoC components at different frequencies, which contributes to power savings and also solves system integration problem.
6) System only operates when data is available, hence energy efficient.
7) GALS inherently generates low EMI.

Table: 1 Input Output pin Description of GALS Core

Signals	Width (Bits)	I/O	Name / Function
Input _ mpu	8	I	Input port of the GALS processor Module
Add _ memory	3	I	Address of the memory location where the 8 – bit data is to be written into the register memory
In _ memory	8	I	Input port for providing data to the register memory
CLK _ ctrl_mpu	1	I	Clock for the control unit module
CLK _ dp_mpu	1	I	Clock for mux + acc. Module
Rst	1	I	Active high reset for the GALS processor module
Wr_mpu	1	I	Active high write signal for the register memory module
Output_mpu	8	O	Output port of the GALS processor module

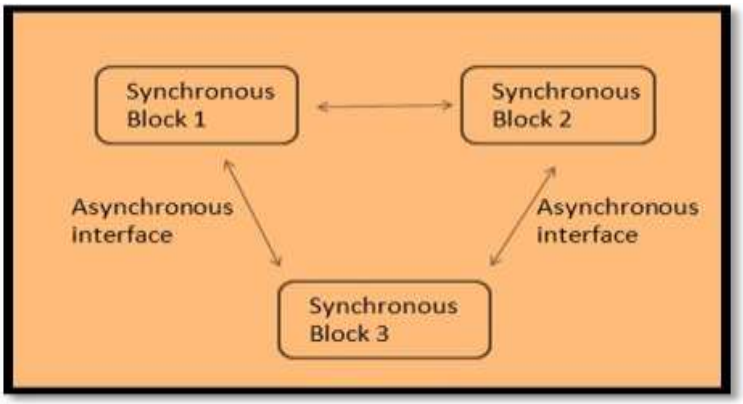

Fig 1. GALS Architecture

V. GALS MICROPROCESSOR DESIGN

Synchronous design is the foundation of the GALS design. Partitioning the synchronous design into locally clocked synchronous modules remains the most critical aspect of GALS. The partitioning has more influence on the performance of the system than all other factors combined. Till date no well-defined methodology has been developed for partitioning. It requires manual intervention to partition the design depending upon the architecture and considering design complexity. By carefully looking at the proposed synchronous processor architecture and considering global clock distribution problem at the processor level (not at the SoC level since processor alone could not constitute a system) our GALS design consists of 4 synchronous modules.

(SMs), operating in 4 different frequencies: SM 1: Control Unit Module

SM 2: Mux + acc. Module

SM 3: Register Memory Module

SM 4: Functional Units (Alu + Shifter) Module

Handshaking between different clocked modules is handled by the simple request and acknowledge signals which are nothing but the control and data signals that mark the validity of the data. Also, the block which is not required in the current execution cycle can be stalled using pausible clock approach. From the above partitioning we can theoretically say that in GALS, the global clock distribution is replaced by local clock distribution due to which clock related problems will now be limited within a particular synchronous module or within a short area and will not be as severe as in the case of fully synchronous design.

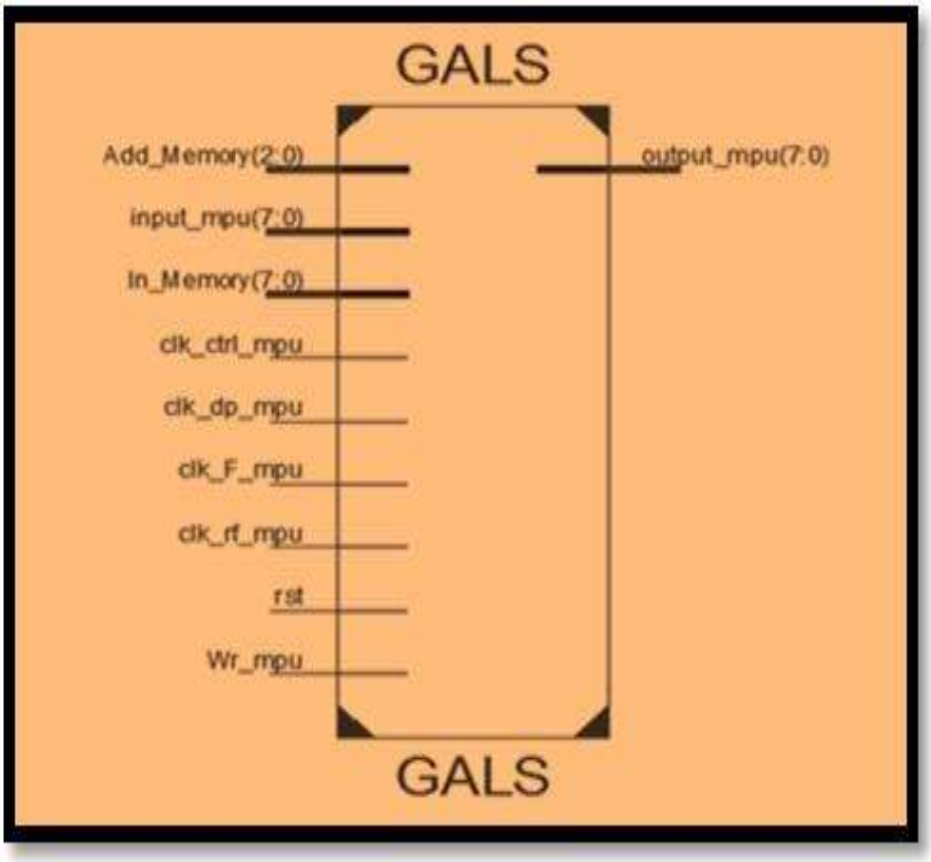

Fig 2. Top level entity of 8-bit GALS processor

VI. DESIGNING STEP

The first design step was to model the processor by programming in VHDL. At this time the model has been refined to a level where a transition to VHDL will be easy. This is the time to choose if the implementation is going to be synchronous or GALS. There is later a possibility to go from synchronous to GALS design and vice versa. Though, the

transition to GALS is easier if the architecture has a good modular design.

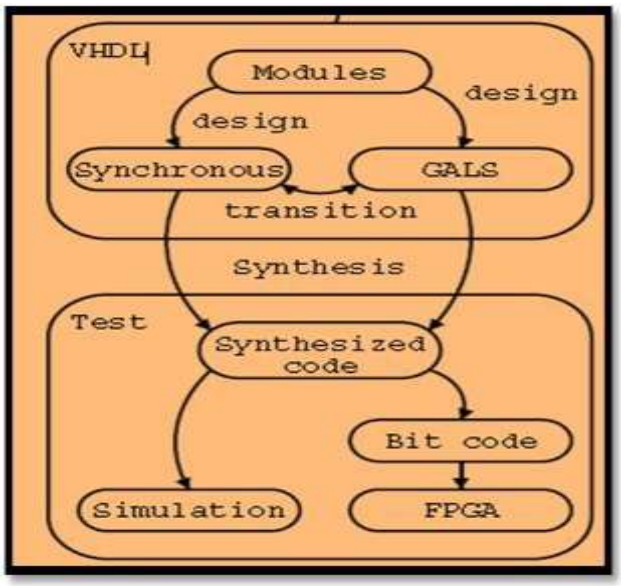

Fig 3. Flow chart

Synthesis of a GALS system is more troublesome than its synchronous counterpart. This problem arises mainly from the lack of support for asynchronous circuits in the synthesis tools. Asynchronous circuits require redundant logic to function correctly and by default the synthesis tools removes the redundancy. By using setting similar to do not modify, the synthesis tool do not change these asynchronous blocks. Testing is here divided into two parts. One part is an FPGA/UTLP test, and the other is a simulation test of the synthesized code.

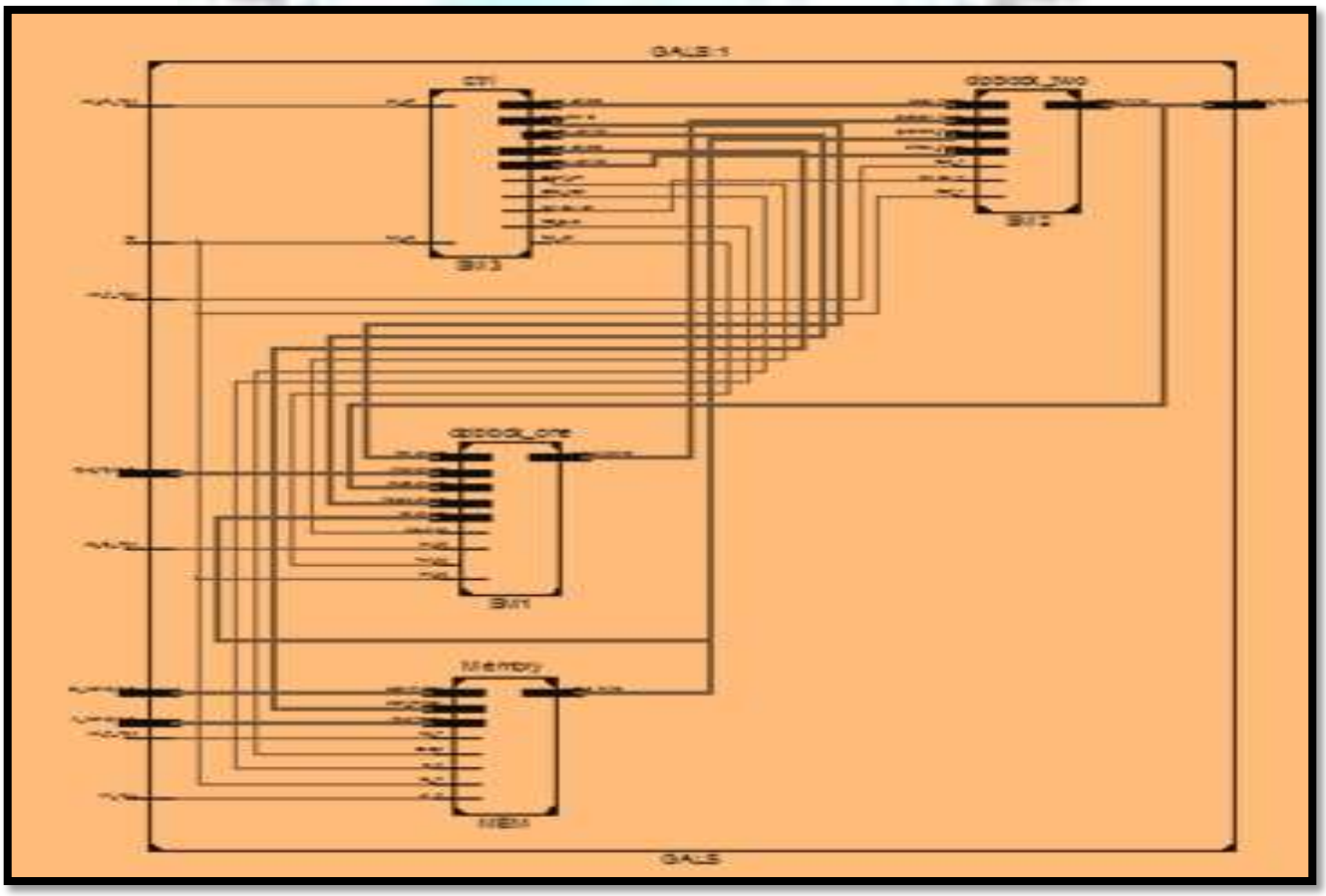

Fig 4 : top level integrated GALS module

VII. RESULTS OF SIMULATION AND SYNTHESIS

The VHDL code for synchronous processor and GALS processor are synthesized using Xilinx Synthesis Tool (XST) and the functionality is verified using ISE simulator (ISim). The VHDL code for synchronous and GALS is synthesized using Xilinx Synthesis Tool (XST) and

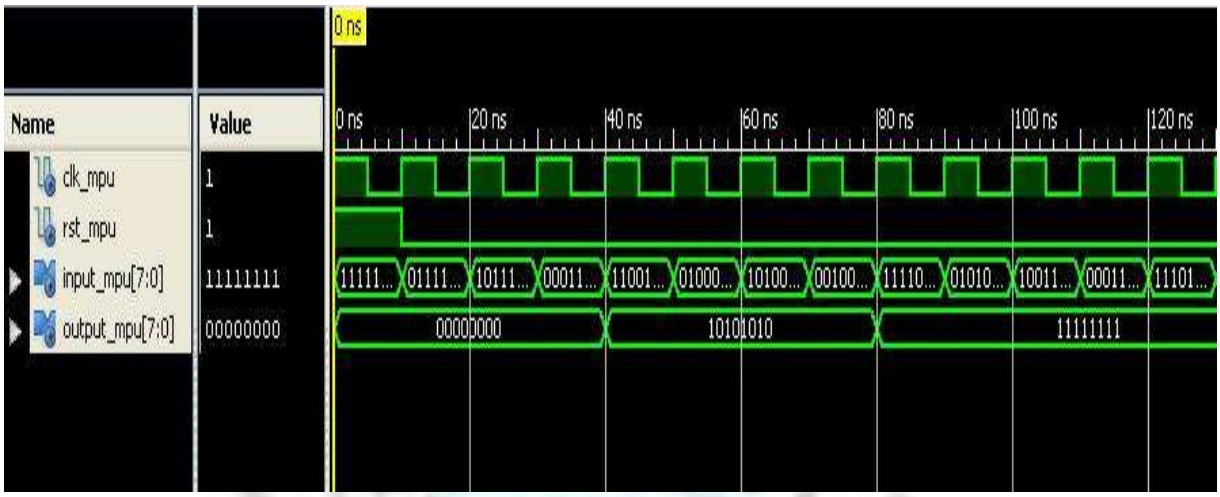

Fig 5. ADD instruction of core

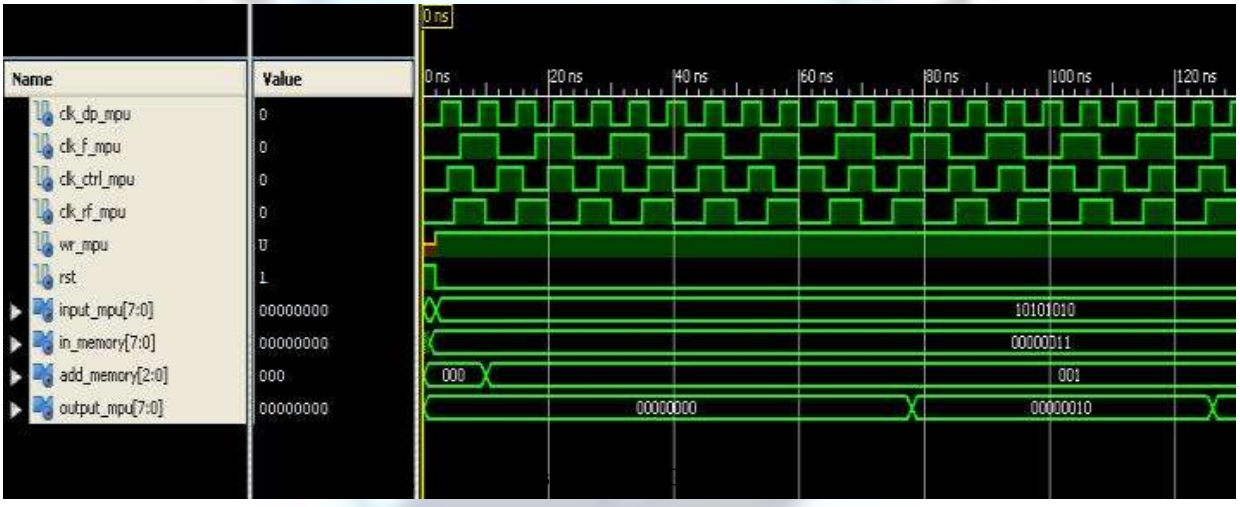

Table: 2 Comparisons of Synchronous Processor & GALS Processor

Parameters	Synchronous Processor	GALS processor
Operating frequency	143.698 MHz	269.233 MHz
Power Consumption	Quiescent power=56mW Dynamic power=3mW Total power=59mW	Quiescent power=56mW Dynamic power=3mW
Area overhead	59 LUTs 31 slices 27 slice flip-flops	67 LUTs 52 slices 69 slice flip-flops

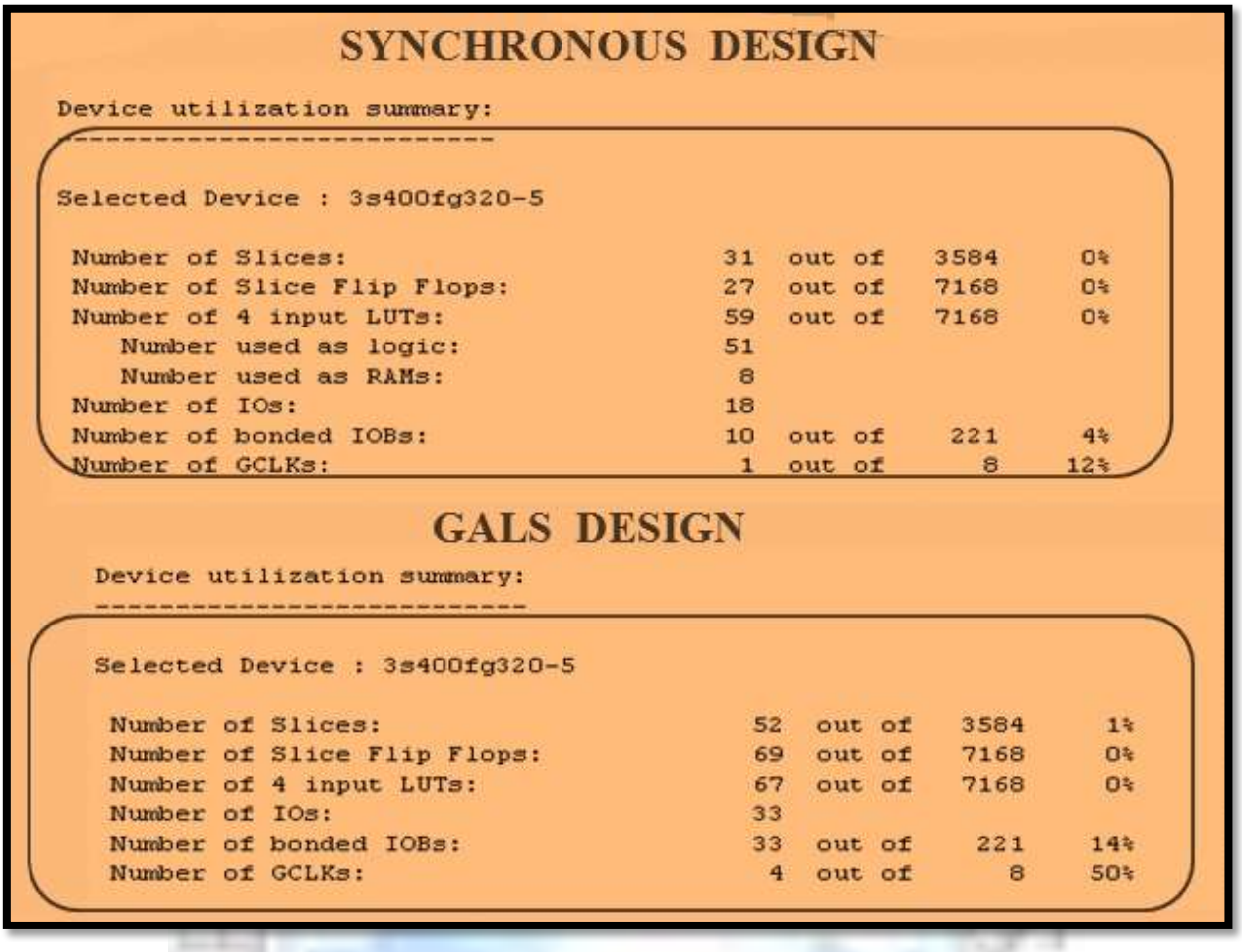

Fig 7. Device utilization summary of Synchronous and GALS Processor

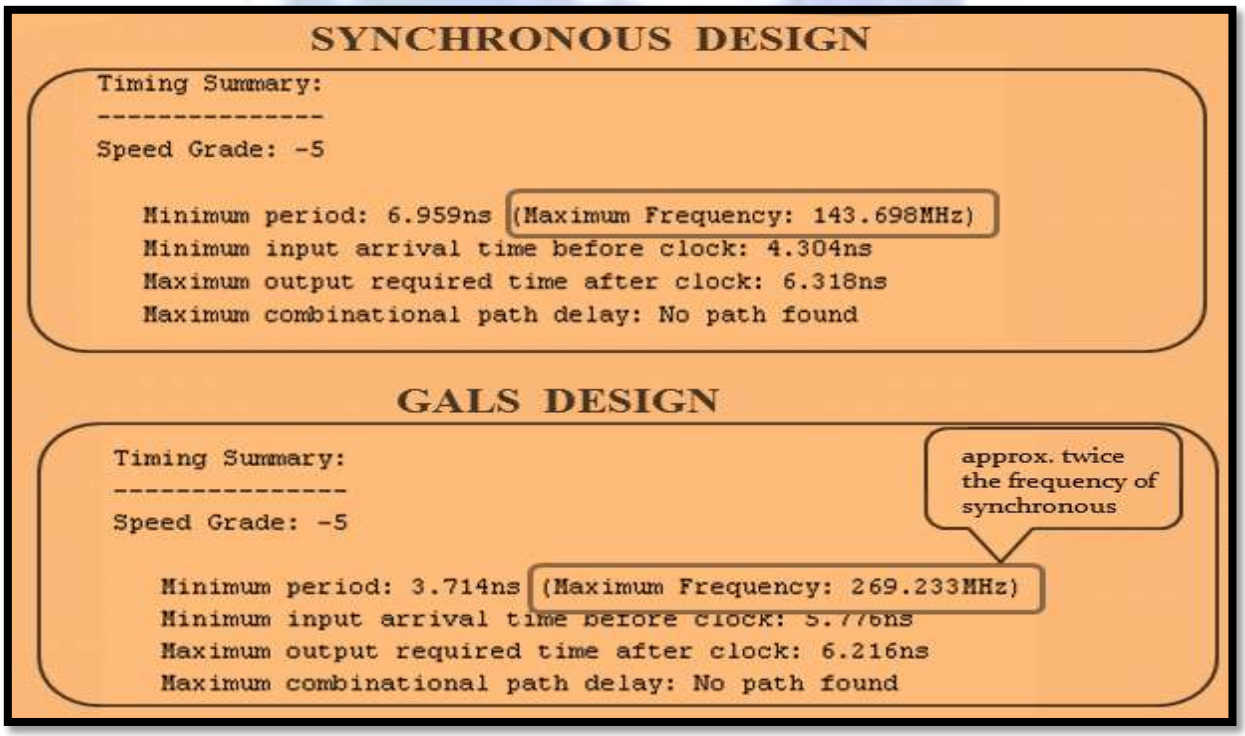

Fig 8. Timing summary of Synchronous and GALS Processor

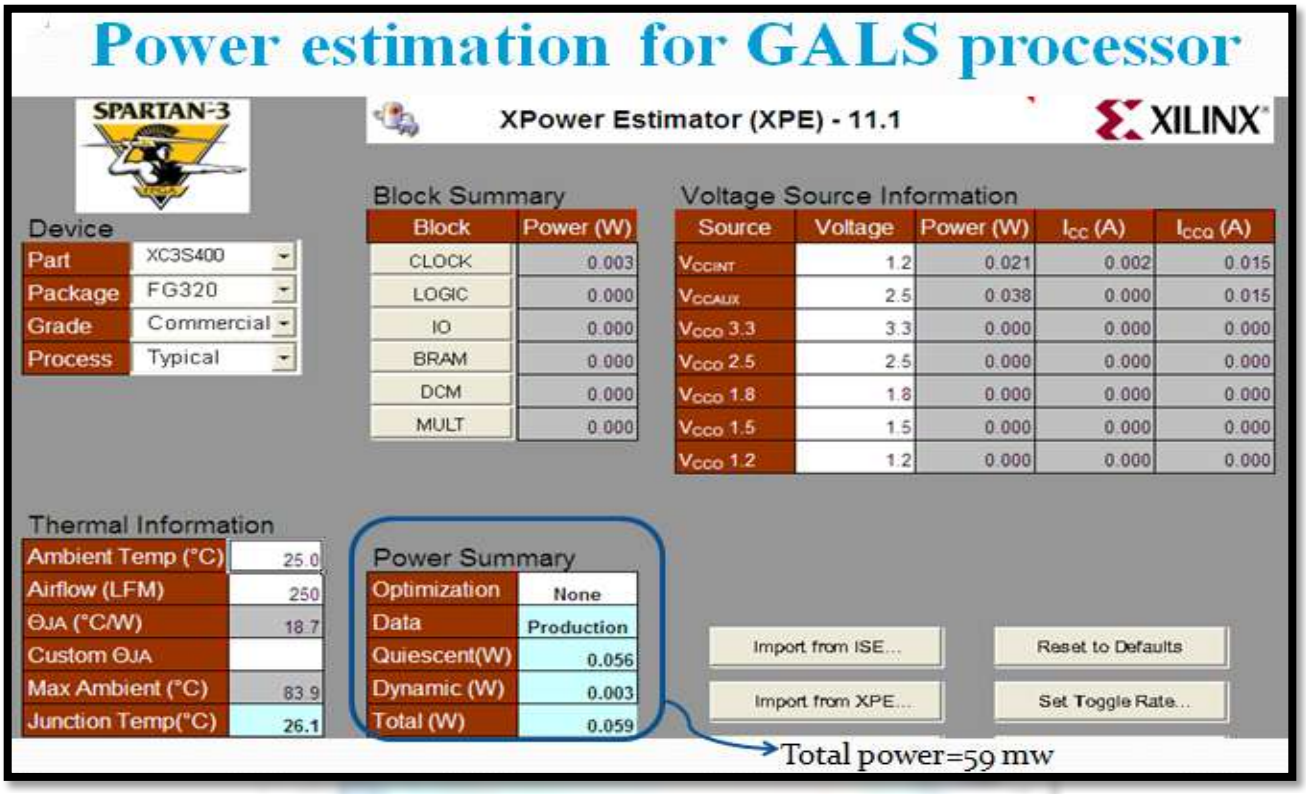

Fig 9. Power estimation of GALS Processor

VIII. FUTURE SCOPE

Technology has seen the development of processor industry right from micro to the latest nano-technology with speed and performance being important criteria, not much attention has been given to the power requirement for these integrated circuits. The presented GALS processor was a simple model of the proposed synchronous processor, designed using pausible GALS clocking style. The extra advantage which we found in our GALS processor was the reduced handshaking latency which is one of the drawbacks of asynchronous design as well as GALS design. Present fully synchronous processors have evolved with a global clock which is supplied throughout the die; this has resulted in unwanted power consumption and dissipation. The lack of global synchronization makes power savings in the clock net possible. Moreover, the independently clocked synchronous blocks opens up great possibilities for substantial power reduction using Dynamic Voltage and Frequency Scaling (DVFS) techniques which can be applied to this work for further reduction in power and also this architecture can be implemented in FPGA

CONCLUSION

A novel methodology for the design of globally-asynchronous, locally-synchronous systems called “synchro-tokens” has been presented. Such systems are deterministic, a property which facilitates validation, debug, and test. This deterministic behavior has been demonstrated with an out-of-order processor core implemented in Verilog. Much work remains before the feasibility of the synchro-tokens architecture can be demonstrated. A larger system with bigger and more SBs will enable studies of the area and performance impact of the synchro-tokens architecture. A schematic implementation will enable a study of the critical paths and a more detailed clock design. Development of validation, debug, and test methodologies which are compatible with existing synchronous tools and testers is also needed.

The presented GALS processor was a simple model of the proposed synchronous processor, designed using pausible GALS clocking style. The extra advantage which we found in our GALS processor was the reduced handshaking latency which is one of the drawbacks of asynchronous design as well as GALS design. Present fully synchronous processors have evolved with a global clock which is supplied throughout the die; this has resulted in unwanted power consumption and dissipation. The lack of global synchronization makes power savings in the clock net possible. Moreover, the independently clocked synchronous blocks opens up great possibilities for substantial power reduction using Dynamic Voltage and Frequency Scaling (DVFS) techniques which can be applied to this work for further reduction in power and also this architecture can be implemented in FPGA.

REFERENCES

[1]. J. Sparso, S. Furber "Principles of Asynchronous Circuit Design- A System Perspective" Kluwer Academic Publishers, 2001
[2]. M. Krstic, E. Grass, F. K. Gurkaynak, P. Vivet, "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook", IEEE Design and Test of Computers, vol 24, pp430-441, sept. 2007.
[3]. P. Teehan, M. Greenstreet, G. Lemieux "A Survey and Taxonomy of GALS Design Styles" IEEE Design and Test of Computers, 2007, pp 418-428
[4]. X. Fan, M. Krstic, E. Grass, "Analysis and Optimization of Pausible Clocking based GALS Design" International conference on Computer Design ICCD, 2009,pp 358-365
[5]. M. A. Rahimian, S. Mohammadi, M. Fattah, "A High- Throughput, Metastability-Free GALS Channel Based on Pausible Clock Method" 2nd Asia Symposium on Quality Electronic Design, 2010 IEEE, pp 294-300
[6]. Wu Ning, Ge Fen, Wu Fei, "Design of a GALS Wrapper for Network on Chip" 2009 World Congress on Computer Science and Information Engineering, pp 592-595
[7]. Shijun Lin, Su Li, Jin Depeg, Zeng Lieguang, "Universal GALS platform and evaluation Methodology for networks on chip" Tsinghua Science and Technology, ISSN 1007-0214, pp176-182, Vol 14, No. 2, April 2009
[8]. Yvain Thonnart, Pascal Vivet, Fabien Clermidy, "A Fully- Asynchronous Low-Power Framework for GALS NoC Integration" Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010, pp 33 – 38
[9]. Z.Yu and B.Baas, "High Performance, Energy Efficiency & Scalability with GALS chip Multiprocessors" IEEE Transactions on VLSI systems 2009, pp 66-79
[10]. Rune Watn, Tormod NjBlstad, Frank Berntsen and Jan Fmde Lonnum, "Independent Clocks for Peripheral Modules inSystem on-Chip Design", SOC Conference, 2003. Proceedings. IEEE International [Systems- on-Chip], pp 25-28
[11]. S.Zhuang, J. Carlsson, L. Wannamal, "A Design Approach for GALS based SoC" Solid-State and Integrated Circuits Technology, 2004. Proceedings. 7th International Conference on, pp 1368 - 1371 vol.2
[12]. René Gagné, Jean Belzile, and Claude Thibeault, "Asynchronous Component Implementation Methodology for GALS Design in FPGAs", Circuits and Systems and TAISA Conference, 2009. NEWCAS-TAISA '09. Joint IEEE North- East Workshop on, pp 1-4
[13]. Esmail Amini, Mehrdad Najibi, Zahra Jeddi and Hossein Pedram, "FPGA Implementation of Gated Clock Based GALS Wrapper Circuits", Signals, Circuits and Systems, 2007. ISSCS 2007. International Symposium on, pp 1-4, vol 1
[14]. A.Iyer, D. Marculescu, "Power and Performance Evaluation of GALS processors "Computer Proceedings. 29th Annual International Symposium on, pp158-168
[15]. M.Kovac, "Asynchronous microcontroller simulation model in vhdl" 2008
[16]. Ryan Mabry, "Asynchronous implementation of 8051 microcontroller" Honour's Thesis
[17]. 17]F.Gurkaynak, T.Villiger, S.Oetiker, N.Felber, H.Kaeslin, W Fichtner, "A Functional Test Methodology for GALS Systems", Asynchronous Proceedings. Eighth International Symposium on, pp 181-189
[18]. E. Hwang, "Microprocessor design Principles and Practices with VHDL" La Sierra University 2004 ISBN: 0-534-46593-5
[19]. http://www.xilinx.com/
[20]. Miguel Prices et all" Chained In-Order/Out-of-Order Double Core Architecture" in 2006
[21]. Francisco Javier Cazorla Almeida "Quality of Service for Simultaneous Multithreading Processors" in 2005.
[22]. O. Mutlu, J. Stark, C. Wilkerson, and Y. N. Patt. Run ahead Execution: An alternative to very large instruction windows for out-of-order processors. In Proc .of the 9th Intl. Symp .on High Performance Computer Architecture, pages 129–140, 2003.
[23]. M. Pericas, A. Cristal, R. Gonzalez, D. A. Jimenez, and M. Valero. A decoupled kilo-instruction processor. In Proc. of the 12th Intl. Symp. on High Performance Computer Architecture, February 2006.
[24]. H. Zhou. Dual-core execution: Building a highly scalable single-thread instruction window. In Proc. of the 14th Intl. Conf .on Parallel Architectures and Compilation Techniques, September 2005.
[25]. H. Akkary, R. Rajwar, and S. T. Srinivasan. Checkpoint processing and recovery: Towards scalable large instruction window processors. 2003.
[26]. R.Kumar, V.Zyuban, and D.M.Tullsen. Interconnection in multi-core architectures: Understanding mechanisms, overheads, an scaling. In Proc of the 32ndIntl.Symp. on Computer Architecture, June 2005.
[27]. Rashmi Jain et al "Globally Asynchronous Locally Synchronous Design Based Heterogeneous Multi-core System"(eds.), ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of CSI - Volume I, Advances in Intelligent Systems and Computing 248, DOI: 10.1007/978-3-319-03107-1_81, © SpringerInternationalPublishingSwitzerland2014.
[28]. Rashmi A Jain 2.Dr. Dinesh Padole "Scalable and Flexible heterogeneous multi-core system" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No. 12, 2013 [
[29]. H. Akkary, R. Raj war, and S. T. Srinivasan. Checkpoint processing and recovery: Towards scalable large instruction window processors. 2003.
[30]. S. Balakrishnan, R. Raj war, M. Upton, and K. Lai. The impact of performance asymmetry in emerging multi core architectures. In Proc. of the Intl. Symp. On Computer Architecture, pages 506–517, June 2005.