IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 11, Issue 4, Ver. I (Jul.-Aug .2016), PP 17-24
www.iosrjournals.org

Low Resource Fast Operation Using Register Scheduling In Mesochronous Design

Sukanya. K [1], Dr. G. Laxminarayana [2]
[1](Department of E.C.E, TKR College of Engineering and Technology, Ranga Reddy, Telangana-500097)
[2](Department of E.C.E, Anurag College of Engineering, Ranga Reddy, Telangana-501301)

Abstract: *In the process of making the system operating faster with low resource overhead, Mesochronous design approach has been suggested as an optimal solution. Such design performs a segregation of operational units in consideration to delay parameter. The delay units are considered towards common clocking system, to overcome the propagation delay in digital system operation. However, the delays are considered to be linearized w.r.to register alignment and common clocking method. The operation of a Mesochronous unit is performed via instruction and delay unit alignment. Here, the sequencing of instruction, however doesn't consider the instruction redundancy property. In this paper, a new register re-alignment approach is suggested, which schedules the operation sequence by instruction property to achieve, low resource fast processing approach. The design model suggested, illustrates a higher processing efficiency in coding, as in comparison to conventional pipelined approach.*

Keywords: *Clocking unit, Delay, Instruction redundancy, Mesochronous design, Register alignment*

I. Introduction

The upcoming demand of services and their demanded quality, results in the development of various high speed processing units to achieve the current and future service requirement. New processing units, operating in parallel distributed manner are designed to provide faster computation. The services of high processing efficiency and demand for parallel computing of multiple operations has lead to new challenges in such designs. The current and future designing demands for high level computing such as, multi cores processing, distributed processing or process of pipelining to cope with emerging new processing architectures. With demand for high level of processing, the synchronization problem arises in different sub-components. For the design of such system to maximize the processing efficiency, processing capacity of digital system and power consumption, the operational efficiency is measured in term of speed and accuracy for an execution. To achieve this objective, design of fast processing units using, data and clock signals simultaneously to achieve synchronous operation is carried out, which is called as Mesochronous operations. Mesochronous designs are benefited from the processing of synchronous communication, and synchronous coding ability. The clock period in conventional pipeline scheme is observed to be proportional to the maximum stage delay, whereas in a Mesochronous pipelining it is proportional to the maximum delay difference, which leads to higher clock speeds [1].

In the designing of Mesochronous operation, various formats were suggested. In [2], synchronization in a breadth search approach, a fine grain processing for synchronization issue is developed. The data locality in this system is evaluated. A code let model focusing on the fine grain parallelism of processing in synchronous mode is proposed. The Code let Model more efficiently exploits data locality than the Open MP-like execution models which traditionally focus on coarse-grain parallelism inside loops. In [3], a high level implementation of L1-Cache system based on asynchronous communication oriented design style is proposed. Each of the units of cache architecture is pipelined asynchronously using asynchronous interfaces. The enhanced pipelining increases the area and energy resulting to the increased control circuitry requirement for handling large number of handshakes, resulting in saving of considerable area omitting clock generation, distribution and gating circuitry. The developments on the architectural level were reviewed and the future developments towards the synchronization requirement are presented in [3]. Due to the fact that each cache memory implementation is time consuming and error prone process, a synthesizable and a configurable model proves out to be of immense help as it aids in generating a range of caches in a reproducible and quick fashion. The micro pipelined cache, implemented using C-Elements acts as a distributed message-passing system. The RTL cache model implemented, comprising of data and instruction caches which has a wide array of configurable parameters. In addition to timing robustness this implementation has high average cache throughput and low latency. The implemented architecture comprises of two direct-mapped, write-through caches for data and instruction.

In [4], a low complexity link micro architecture for Mesochronous on-chip communication that enables skew constraint looseness in the clock tree synthesis, frequency speedup, power consumption reduction, and faster back-end turnarounds is proposed. In [5], a regular synchronizer and six multi synchronous synchronizers

are implemented on a programmable logic device and the synchronization is measured. An experiment system and method for measuring synchronizers and metastable flip-flops are described. Two separate settling time constants are shown for a detestable flop. Clocking cross-talk between asynchronous clocks is demonstrated. A regular synchronizer useful for communications between asynchronous clock domains, while the other synchronizers providing higher bandwidth communications between multi-synchronous and Mesochronous domains is presented. In [6], a work on Synchronous Digital Hierarchy is reviewed. A brief review regarding the problems in synchronization of different data rate signals in single clock, and master slave technique overcome the issue of synchronization problem. With respect to the state of the art, the proposed link architecture stands for its low power and low complexity overheads. Moreover, it can be easily integrated into a conventional digital design flow since it is implemented by means of standard cells only. In [7], a short tutorial attempts to present the approach and the criticality of the subject of met stability and synchronizers is presented. The issues with system met stability for false operation is proposed. In [8], for generating sequential code, the concurrency expressed in the synchronous programs is sequential zed. The developed approach was designed to run on single-core processors. An attempt generating multi-threaded code data-flow model is proposed.

To optimize the operational performance, a register optimization process is suggested. The rest of the paper is presented in 5 sections. Where, section 2 outlines the conventional and Mesochronous operation, section 3 present the register optimization technique suggested in synch-Mesochronous design, section 4 outlines the experimental results obtained and section 5 conclude the proposed approach.

II. Mesochronous Design

In a conventional pipeline (CPP) scheme, a digital system is divided into small sub-systems called pipeline stages buffer separated by pipeline registers. In a pipelined system, different pipeline stages operate on different data sets simultaneously and each stage on only one data set at any given time. In the Mesochronous approach, the data is processed to one or more processing unit based on one or more than one dataset simultaneously. In this scheme a delayed clock signal is included in a path as a delay element. Due to higher clock frequency distribution the influence of clock uncertainties is mitigated significantly. This reduction in clock logic reduces the logical overhead, and hence minimizes the power consumption. The number of register is lower in this case, which result in low power consumption. However in such approach it is observed that, for each operation numbers of operational registers were developed to perform pipeline operations with a clock delay input. This register allocation is made based on instruction to execute. Each register is then allocated with a delay unit to provide a clock delay as per the operational instruction. Here it is observed that for each register, delay logic is introduced. In such case, for K-operation registers, K-delay units are used. Each delay unit is latch unit, which is used for buffering the clock cycle and get released when a control signal is applied. Where in to latch, the clock power is dissipated. This power dissipation results in power wastage. It is hence required to optimize the Mesochronous pipeline operation by optimizing the operation of register bank or/and delay unit to minimize the operational overhead. The method outlined in [1], defines a mesochronous pipeline scheme modifies conventional pipeline scheme to achieve performance gains. In the proposed scheme, the system is clocked such that a Since in a typical system, most gates do not switch every pipeline stage is operating on more than one data set clock cycle, the switching frequency can be expressed as a simultaneously.

III. Register Optimization

In the process of Mesochronous operation, the registers are aligned in a sequential manner which leads to higher processing delay, to optimize this issue, a operation register realignment is suggested. In the design process, The IF unit fetches an instruction from the instruction memory at the current program counter (PC). The ID unit reads the values of the source operands from the register file. The third functional unit in the pipeline is EX and it performs arithmetic logic operations, such as add and multiply. In the MEM unit, memory is accessed (read from or written to). Finally, the WB unit writes the new value of the destination register to the register file.

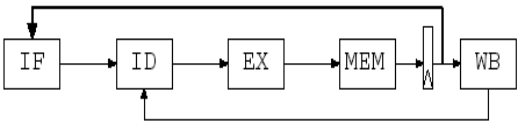

Fig. 1: Conventional Processing Operation

The process in Fig. 1 is unpipelined, because there are no pipeline registers separating the functional units, except for the register between MEM and WB, which is necessary because of using a write-before-read register file. The functional units are represented by rectangular boxes and the wires between them are shown using arrows. The bold line represents the instruction path that carries a new pc back to the fetch unit, when a

branch occurs. To design the system, synchronization is carried out between two pipeline registers segment. The clock period (clock cycle) is measured in real time and it is equal to the longest time taken by a segment to process an instruction. In Fig. 2, the dashed lines show the 3 segments in the unpipelined processing. There is an implicit pipeline register at the beginning of the pipeline, before the first functional unit.

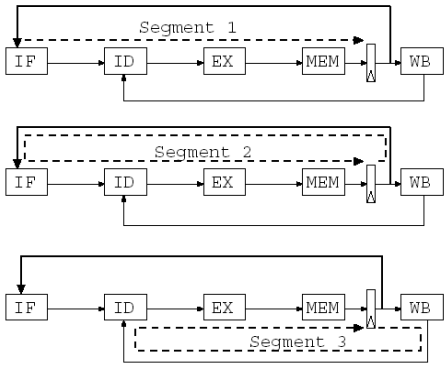

Fig. 2: 3 Segments in the Processing Operation

Fig. 3 shows how an instruction passes through every functional unit in one cycle.

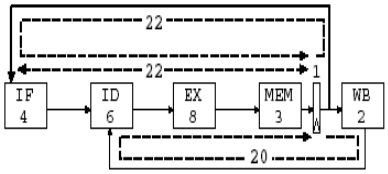

Fig. 3: The Sample delay operation

For the functional units, the calculations for the delay of each segment are shown. For example,
Segment 1 = IF + ID + EX + MEM + register = 4 + 6 + 8 + 3 + 1= 22
Segment 2 = IF + ID + EX + MEM + register = 4 + 6 + 8 + 3 + 1= 22
Segment 3 = WB + ID + EX + MEM + register = 2 + 6 + 8 + 3 + 1= 20
Clock Period = Max (Segment 1, Segment 2, Segment 3) time units = 22

The delay of Segment 1 and 2 is 22 time units, and delay of Segment 3 is 20 time units, so the clock period is 22 time units (equal to the segment with the highest delay). In an unpipelined microprocessor, each instruction is entirely processed in a single clock period; the throughput is equal to the inverse of the clock period. Therefore, decreasing the clock period increases the throughput.

In Fig. 2, the clock period is 22 time units and the throughput is 1/22. In other words, 1 instruction is processed every 22 time units. Pipelining can be used to reduce the clock period. Fig. 4 contains a partially pipelined Mesochronous and the clock period is reduced to 12, which improves the throughput to 1/12 or 1 instruction every 12 time units.

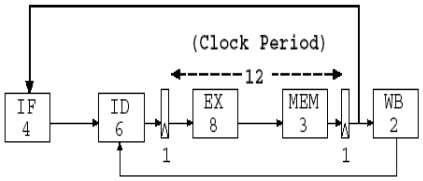

Fig. 4: Sample circuit partially pipelined with delay

The computation of an instructions being processed in the conventional processing is illustrated in Fig. 5, which shows instructions traveling.

		Instruction 1	Instruction 2	Instruction 3
Clock Cycle	1	IF ID EX MEM WB		
	2		IF ID EX MEM WB	
	3			IF ID EX MEM WB

Fig. 5: Instructions being processed in the conventional processing

A fully pipelined microprocessor has a pipeline register between each of the adjacent functional units. The fully pipelined Mesochronous is a 5-stage pipeline, because the 4 pipeline registers divide the linear data path into five sections, which are called stages. Through the stages of the pipeline "Instruction 1" enters the pipeline and the fetch task is completed by the IF unit. In the next clock cycle, the ID unit processes "Instruction 1", and the fetch unit processes "Instruction 2". At each clock cycle, a new instruction may enter the pipeline and each instruction already in the pipeline moves to its next stage. When an instruction moves out of the WB stage, it has been completely processed as shown in Fig. 6.

	Instruction 1	Instruction 2	Instruction 3
1	IF		
2	ID	IF	
3	EX	ID	IF
4	MEM	EX	ID
5	WB	MEM	EX
6		WB	MEM
7			WB

Fig. 6: Instructions being processed in the fully pipelined Mesochronous

The clock period must be long enough for each stage to complete its computation, which is why the clock period is equal to the segment with the highest delay. Less computation is done in a stage of the pipelined configuration (Fig. 6) than the unpipelined configuration (Fig. 5), means that the clock period is less. Based on the examples above, it could be inferred that a microprocessor with pipeline registers will operate faster in mesochronous design with delay synchronization as shown in Fig. 7.

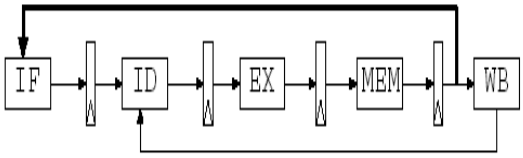

Fig. 7: Fully-pipelined Mesochronous Design with delay synchronization

The suggested approach is a promising architecture that enables Pipeline in the order of 6 instructions per cycle (IPC) for the next generation of processors by utilizing both instruction level Pipeline and thread level Pipeline (ILP and TLP respectively). The basic idea is to enable efficient processor resource sharing (i.e. to avoid partitioning) by pipeline threads. The goal is to achieve improved performance without extensive changes, as compared to an out-of-order wide-issue of Mesochronous design.

IV. Experimental Results

To evaluate the proposed approach, a HDL definition of the proposed architecture is developed, targeting onto Xilinx vetex2p, xc2vp1112tb-6 FPGA device. The timing results and the implementation results obtained for the developed approach is as presented. The Synchronous Clock Generation Process for the developed approach is illustrated in Fig. 8.

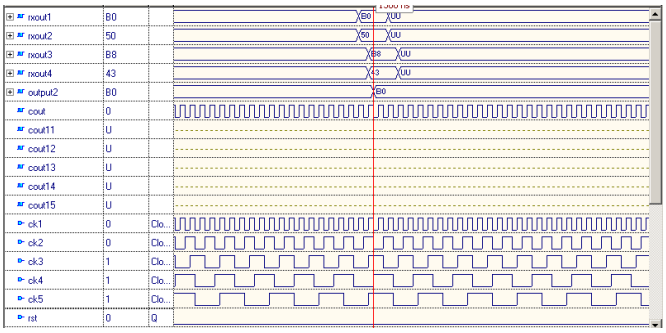

Fig. 8: Synchronous Clock Generation Process

The master clock time stamp is recorded and 5 sub synchronous clock pulse 'clk1-clk5' is derived. The best synchronous clock operation is as computed.
For a total number of communicating nodes = 4
Total amount of data generated per node = 3 bytes
Total amount of expected data in processor = 12 bytes
Total time taken = 5580 ns (under non synchronous round robin based communication)
(total time = processing time + communication Time)
Total time taken = 1445 ns (under 1588 synchronous mode communication)
Total time saved (Ts) = 4135 ns
For System clock perod of 10ns,
Total clock cycles saved = Ts / Sys_Clk
= 4135 / 10 = 413 cycles

Applying this clock pulse over the developed system, the timing result for the data input and the corresponding control signals are illustrated in Fig. 9. Input data of 4 simultaneous data were taken per cycle and as '1, 2, 3 and 4', which were then passed to FIFO unit to buffer based on the generated control signal.

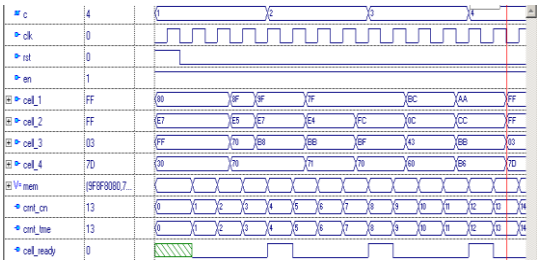

Fig. 9: Timing result for data input with control signals

For every data arrival, a 'ready' signal is initialized to indicate the availability of new cell data arrival. These data are buffered on the FIFO unit, as illustrated in Fig. 10.

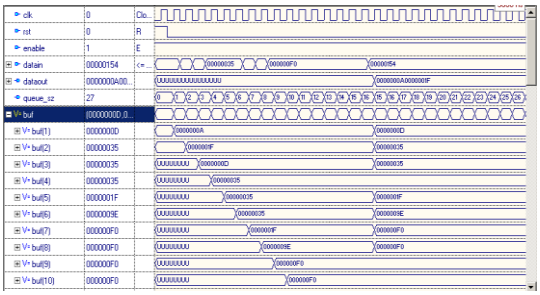

Fig. 10: Simulation result for buffering operation

The Buffer logic is developed as a FIFO unit, wherein the incoming bytes are buffered in a sequential order, on the write mode of operation, and are read out from top to bottom on read mode of operation. On reading stage, these data are passed to the transmit interface unit, to stream the data over for execution based on the instruction passed.

For each 'ready' with a value '1', a block of data is transferred out as transmitting stream from the buffer logic. A set of data bytes are read from the buffer unit and passed to the transmitting interface unit. Based on the allocated data rate, these data bytes are then passed for execution. The Timing simulation observation for the developed integrated model of synch_Meso coding is illustrated in Fig. 11.

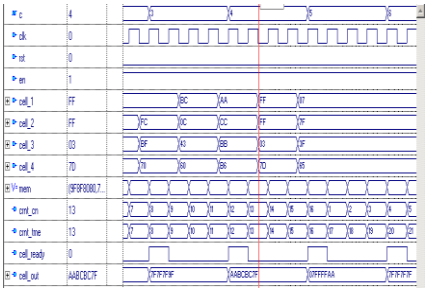

Fig. 11: Timing result for transmission operation

The operation of the developed processing unit and its control signals are shown in below Fig. 12-14. The drop signal, discardation signal, are set low at the initial conditions, as the buffered queue length is observed to be below the lower set limit of buffering. The data are passed to the processing unit from dataout line for buffering. On each read signal, from the buffered data, a set of 32 bits of information is passed out to forward.

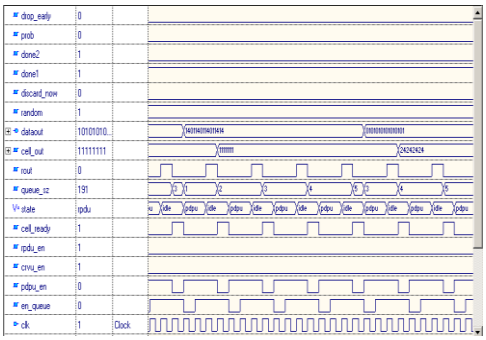

Fig. 12: Simulation result for Integrated synch_Meso

The Queue size is computed in accordance, and passed to the controller unit. The Controller operation is developed in a FSM modeling, wherein the controller state transitions are observed on signal 'state'. The system is passed with the master clock pulse through the signal port 'clk'.

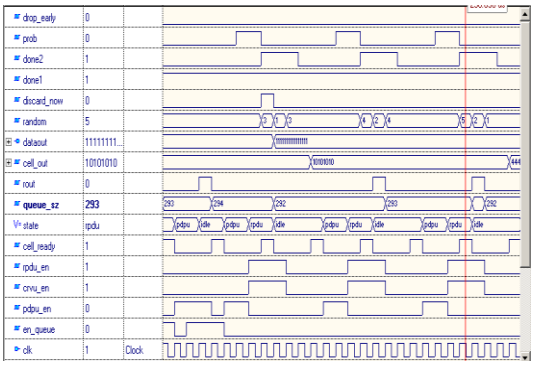

Fig. 13: Simulation result for Integrated synch_Meso

When the data packets are within the limit of lower queue margin all the packets are buffered in to the memory element and queue size keeps on increases. Once queue size exceeds minimize set limit the random dropping of incoming packet initialized. It is observed that the enabling of random packet dropping and random value computation is initialized The en-queuing packet is than dynamically toggled by the random count value generated. Once the limit of maximum queue size is exceeded the discardation of packets is prolonged, and all the packets are routed via other routing nodes. The operation of random count value is then processed periodically rather than randomly.

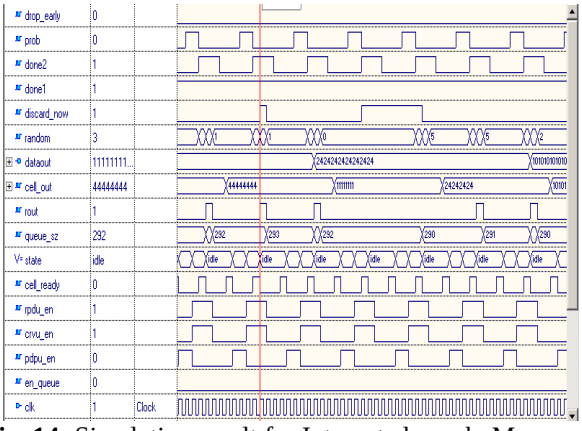

Fig. 14: Simulation result for Integrated synch_Meso unit

For the realization of the developed system and its feasibility for practical application, the proposed approach is targeted to FPGA device, targeting to Xilinx Vertex-2P FPGA. The Obtained Design statistic for the developed system is as outlined below.

Design Statistics
IOs : 90
Cell Usage :
BELS : 2194
Minimum period: 14.231ns (Maximum Frequency: 70.268MHz)
The required Number of I/O lines are observed to be 90.

With the usage of BELs of about 2194 logic elements. The obtained operation frequency for developed system targeting to Viretx2P-FPGA device is obtained as 70.268MHz. To observe the placement of the developed system a logical routing over the targeted device is carried out. The Logical routing for the developed system over the targeted FPGA is presented in Fig. 15.

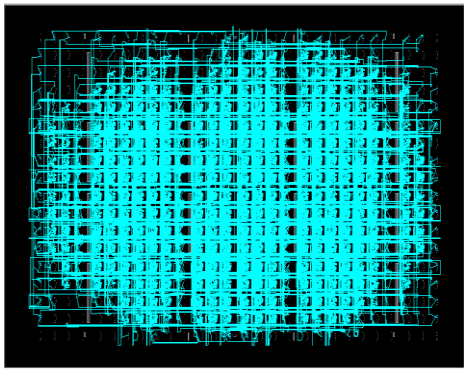

Fig. 15: Logical routing of the developed design targeting to Xilinx xc2vp112 tb-6 FPGA device

The logical block placement on to the targeted FPGA is observed in Fig. 15. The developed system occupies about 87% of the available logical resource for its implementation. Fig. 16 shows the floor plan for the developed unit.

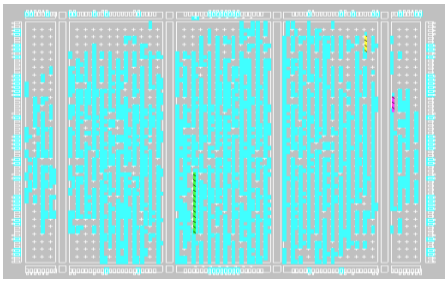

Fig. 16: Floor plan of the developed design unit for targeted FPGA device

The physical packaging of the developed system is presented in Fig. 17. The package pin configuration for the developed system illustrates a usage of 23% of the interface ports available after targeting onto the Xilinx vetex2p, xc2vp1112tb-6 FPGA device.

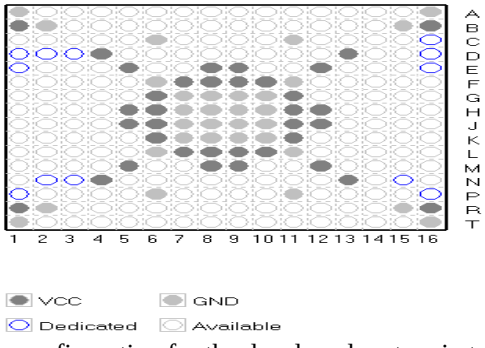

Fig. 17: Package Pin configuration for the developed system in targeted FPGA device

The comparative analysis for the developed approach is summarized in Table 1 below:

Table 1: Analysis for the developed method

Method	Clock Utilization	Registers Used	Logic Implemented	Total Resources Utilized
CPP (Existing) [1]	86.9	66.6	38.2	191.7
MPP (Existing) [1]	24.2	12.8	45.3	82.3
CPP/MPP (Developed)	3.6	5.2	0.84	2.3
Synch_Meso (Developed)	1.43	2.19	0.8	0.8

V. Conclusion

In this approach, pipeline register realignment is suggested using Mesochronous design. The approach is observed to be faster in processing and resulted in delay minimization as compared to the conventionally pipelined design. The delay synchronization w.r.to the delay parameter and operation instruction result in optimal design of a Mesochronous design approach. The suggested approach is tested for a targeted Xilinx vetex2p, xc2vp1112tb-6 FPGA device, and the implementation illustrates an improvement in realization efficiency (around 65% of resources saved) as compared to the conventional design.

References

[1]. Suryanarayana B. Tatapudi and Jose G. Delgado-Frias, "A Mesochronous Pipeline Scheme for High Performance Low Power Digital Systems", ISCAS, IEEE, 2006.

[2]. Chen Chen, Souad Koliai, and Guang Gao, "Exploitation of Locality for Energy Efficiency for Breadth First Search in Fine-Grain Execution Models", tsinghua science and technology, 18, 6, 2013.

[3]. Mansi Jhamb, R.K. Sharma , A.K. Gupta, "A high level implementation and performance evaluation of level-I asynchronous cache on FPGA", Journal of King Saud University – Computer and Information Sciences, Elsevier 2015.

[4]. Francesco Vitullo, Nicola E. L'Insalata, Esa Petri, Sergio Saponara, Luca Fanucci, "Low-Complexity Link Micro architecture for Mesochronous Communication in Networks on Chip", IEEE Transactions on Computers, Vol. 57, No. 9, September 2008.

[5]. Yaron Semiat and Ran Ginosar, "Timing Measurements of Synchronization Circuits", Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems, IEEE, 2003.

[6]. Abhishek Agwekar, Mohammed Ahmed, Ramanand Singh and A. Riyaz, "Synchronization Problems in Synchronous Digital Hierarchy (SDH) Communication System and Master Slave Strategies," International Journal of Scientific Engineering and Technology, 2012.

[7]. Ran Ginosar, "Metastability and Synchronizers : A Tutorial," IEEE Design & Test, 2011.

[8]. Bijoy A. Jose, HirenD. Patelb,Sandeep K. Shuklaa and Jean-Pierre Talpin, "Generating Multi-Threaded Code from Polychronous Specifications," Electronic Notes in Theoretical Computer Science, 2009.

K. SUKANYA presently working as Associate professor in the department of Electronics and Communication Engineering at TKR College of Engineering & Technology, Medbowli, Meerpet, SaroorNagar, Hyderabad, Telangana State, INDIA. She has 7 years of teaching experience. She is associated with ISTE as life member. She has obtained B. Tech. degree in Electronics and Communication Engineering from Jayamukhi Institute of Technological Sciences, Warangal, Jawaharlal Nehru Technological University Hyderabad, in 2006, M.Tech. degree in Embedded Systems from Ramappa Engineering College, Warangal, Jawaharlal Nehru Technological University Hyderabad, in 2011 and my area of Research interest is Embedded Systems, Ph.D (ECE) from Jawaharlal Nehru Technological University, Hyderabad and it is my part of Research work.

Dr. G. LAXMINARAYANA presently working as Principal of Anurag College of Engineering. He has obtained BE from Osmania university, M.Tech from Indian Institute of Science, Bangalore and Ph.D from JNTUH under the guidance of Dr. K. Lalkishore (VC of JNTUA). He has 5 years of industrial experience and 30 years of teaching experience. He worked in Osmania University from 1979 to 1998. He worked as Head of the department, ECE at Sreenidhi Engineering College, VBIT and Aurora. He also worked as Director of Aurora Scientific Technological and Research Academy and Principal of Holy Mary Institute of Technology. He is an industrial consultant in instrumentation and worked in South Central Railways. He is associated with ***IETE for last 20 years*** and also member of IEEE, ISOI and ISTE. Presently he is an ***Executive Committee member*** in the present body of Hyderabad IETE chapter and ***R&D subcommittee chair at IETE Hyderabad***. He is supervising 12 Ph.D students and published various papers in International journals and reputated National journals.