
ISSN: 2321-2152
IJMECE
International Journal of modern
electronics and communication engineering
E-Mail
editor.ijmece@gmail.com
editor@ijmece.com
www.ijmece.com

ISSN2321-2152 www.ijmece .com

Vol 8, Issuse.4 Nov 2020

A MESOCHRNOUS TECHNIQUE based FPGA implementation of multibit flip-flops

P.Shankar[1]Dr.Tejawath Ramakrishna[2],

Abstract:

More relaxed clocking techniques such as mesochronous clocking replace completely sync clocking to enhance system composability and simplify timing closure. Under this regime, the modules on two ends of the mesokronous interface get the same clock signal, which works at the same clock frequency, but an unknown phase relationship might occur on the margins of the arrival clock signals. Clock synchronisation is required if data is sent across modules. In this short we introduce a unique mesochron first-input dual-clock first-output buffer (FIFO), which can manage clock synchronisation and temporary data storage, syncing data implicitly through explicit flow control synchronisation alone. Even if the transmitter and receiver are separated by a lengthy connection whereby delay cannot fit inside the intended operating frequency, the suggested system can function well. In such cases, the suggested mesochronous FIFO may be modified to accommodate delays with multi-cycle connections modularly and with little changes to the baseline design. The novel architecture is shown to produce a much-reduced cost implementation compared to prior state-of-the-art mesochronous FIFO architectures.

I. INTRODUCTION

The main development architecture in the field of rapid computer interfaces is Multiprocessor System-on-Chips (MPSoC). The evolution of new technologies has brought forth the necessity for MPSoC. However, the computer overhead and energy requirements have resulted in its optimization required for such a sophisticated design. The designers are dealing with this problem in two ways, by adapting the design to the application limit[1] and by scaling the operation to a restricted voltage / frequency operation[2,3]. Whereas adaptation is an optimum technique, the overall design is substantially high [4]. The design technique comprises monitoring the communications protocol and signal interface between different components [5] in the processor unit while optimising the overhead power and processing. The variety of the design units and the components utilised in this design are also a key restriction in the MPSoC optimization process[6]. The optimization restrictions also limit the operating frequency and system performance[7] in certain applications. This is why the design approach is described with an internal clock allocation updating process[8] and a FIFO-based technique for synchronisation across many units in sub unit activities. Here each core unit is linked to synchronise data exchange across various core units[9].Each of the IP core

Professor[1], Assistant Professor[2,]
Department of ECE Engineering,
Pallavi Engineering College,
Mail.id: ec_shankar@pallaviengineeringcollege.ac.in, Mail ID:pectejawath@gmail.com,
Kuntloor(V),Hayathnagar(M),Hyderabad,R.R.Dist.-501505.

processor blocks employs a FIFO dual clock design. However, if all IP blocks are using a dual-clock FIFO design for one common purpose, the resource is more at risk than the provision, because the configuration of all IP interfaces must be conservative, as the speed and throughput of each IP core is different[10]. For example, the buffering of this synchronisation parameter should be modified for a worst-case scenario based on the comparison between source and receiver frequency [11]. Furthermore, the descriptive existence of frequency ratio information (such as the interconnection of a chip operates at a quicker rate than the interconnected IP units) together with performance restriction information can lead to twin high impact specializations[12]. Therefore, the FIFO dual clock design has a wide area and power conservation at last. Different uses are given in [12-17]. Since the designs do not employ clocks, the synchronisation process is difficult to accept between two clock variations[18]. The delay factor in the clock system is ignored while synchronising the various core units. This restricts the synchronisation in this way. Recent advancements show that the delay factor in MPSoC architecture is minimised. The lag due to resource allocation is not overcome at the time of the processing. Here each instruction process has a delay in processing the clock; clock delays must be assigned and the transition to that assignment results in the system processing delays. A novel latency monitoring technique is given in this article by providing the enhanced clock library function. This technique minimises the interchange of data and the delay in instruction and overcomes the overhead latency during instruction. This document is described in 6 sections to present the paper. Section 2 outlines the MPSoC design approach to library coding. Section 3 describes the recommended technique for library code for Mesochronous operations to measure latency. Section 4 showed the results of the simulation for the strategy devised and the conclusion offered in section 5.

II. DISTRIBUTION OPERATION IN VA-MPSOC

CoreVA-MPSoC shows the target application for an integrated and energy-efficient hierarchical interconnect architecture. In this CPU cluster, multiple core CPUs in the cluster that share a comparable address space in the core are connected nicely. Every CPU may read and write local data from other CPUs through a bus-based interconnection at its original design. The CPU is connected to a cluster interconnection using a FIFO buffer to prevent cycles of penalty operations. The topology of the standard data bus width is specified when the processing unit is designed. For example, Advanced Bus Architecture Microcontroller (AMBA) utilises an AXI4 Interconnect Standard 32 bit or 64 bit data bus width. For particular addresses, AXI4 defines address and data transmission. In addition, distinct channels for reading and writing enable simultaneous read/write (R/W) bus requests. Steps can be added for the connection registration to simplify the compensation for space and route time, increasing the frequency to a maximum of MPSoC clocks. Here, the architecture does not allow the best read requests as it does not work for all cores in the sequence of execution. The CPU boasts the lowest 4-clock operating cycle CPU latency. For Share Bus implementing a total of five intermediaries (1 per channel) are required and for each operation (read/write) two intermediaries are required for crossbar linkages. The Network Interface (NI) interface is formed during the two CPUs connection via the Network-on-Chip (NoC) interface. Each CPU cluster is placed in a 2D structure via a separate X and Y coordinate index. For all cluster memory and units, a common address space is employed in the cluster. NI bridge-based communication during the interchange of CPU clusters and packets via interconnections with routers. As such, it offers the flow control capable of decreasing the operating duration of the CPU for this contact in the CPU core. In order to do so, packet data is saved and retrieved from all local memory of the CPU directly. CPUs therefore take use of local memory access delays. NI also functions as a DMA controller through the distribution of parallel data to the CPU. NI is connected to the cluster in the original setup through a master and a slave port. Packets may be routed to various R/W separation channels. Where the AXI master port of NI can be sent whilst writing data at the same time.

In the exchange of schedules for different CPU interfaces an effective communication strategy is necessary. CoreVA-MPSOC[19] employs a communications paradigm with a single communication channel integration. This method offers more scalability and efficiency than common memory ideas, which can interrupt memory accesses. In general, one job reads and writes on one or more output channels from one or more input channels. Each channel controls the data storage of one or more R/W. Synchronization is regulated by granularity of buffer size. When a channel's buffer requests, the CPU doesn't acquire enough data or no free basic buffer may be written.

However, since data is received via a channel, any memory location in that buffer is accessed by random application. Moreover, no additional synchronisation is required. When the work has been completed, the CPU communicates with the other units in order to exchange the status and reuse the registry. The pooling of resources therefore minimises the overhead. The allocation delay is nonetheless significant. The delay due to resource allocation and sync is also significant, as time delay computation is restricted to the data interface between various CPU units utilising the NoC interface. The NoC operates as a bus arbitrator as a data exchange routing bus. This exchange shows a substantial delay that leads to a reduction in processing speed. Therefore, the time limit, which is focused on in this article, must be decreased. A novel delay mapping technique employing the time stamp library is presented to produce a quicker clock allocation during the syncing procedure.

III. MESOCHRONOUS CLOCK VIRTUALIZATION IN MPSOC (VR-MPSOC)

A virtual delay calculation unit is provided in the suggested method to the virtualisation of the MPSoC operation. This suggested technique changes the current MPSoC design for each core unit with a library unit; for each operating instruction of the processor unit, the library Unit is specified by a pre-computed delay parameter. The instructions for a multi-core processing unit are referenced in this design approach where instructions are classified as 1, 2 or 3 byte instructions. Every operation is conducted here as a direct addressing operation or a direct addressing operation or two indirect addressing operations. The clock delay is calculated for each of these types. The time calculation is calculated as an aggregate delay of the physical delay due to the manufacture of the device and the total delay in the set-up and the time for data is maintained. Each delay is calculated for each type of instruction and set as a library for each core unit. This delay input procedure is conducted throughout the design process as the processing instructions of a design processor are constant and the delay parameter is constant for each instruction. Each instruction is processed to complete delay computing and a matching delay in the library function is set for each instruction. Each library is stored in the core unit as a synchronisation table. This library unit is mapped with the processing instructions during operational stage and the delay is mapped to the arbiter unit. The arbiter is specified at the MPSoCNoC interface that performs the bus arbitration procedure. Each bus line is allocated on the basis of the core bus request from the core unit. In this method of allocation, the delay of the mapped instructions is applied for each allocation. This minimises the extra calculation delay and reduces latency in MPSoC operation. The suggested approach's latency parameter is the accumulated delay owing to clock allocation and calculation delay. In each execution of the instructions, the data or instruction takes an instruction/data buffering time to synchronise the process. Each unit is processed for a delay value according to the instruction type. As the added latency corresponds to the entire delay from allocation to calculation, and the time delay for the clock to sync and the delay is high. In addition to the processing delay, the bus allocation and the data exchange also observes a route delay. Where attempts are made to decrease the road delay by appropriate architectural floor layout, the delay is substantial during operation. The main element of switching and calculation is eliminated by the library unit to overcome this delay. This is a virtual implementation of a time stamp unit that returns the appropriate delay value during operation. This leads to virtual MPSoC design being developed called 'Vr-MPSoC' units. The technique proposed is described in the following algorithm.

Algorithm (Clock switch virtualization)
Process Initialize:
Step 1: *define the cluster of CPU*
Step 2: *allocate the arbiter for data and instruction*
Step 3: *allocate the operation instruction for each CPU*
Process read:
Step 1: *generate a read offset signal to library latency*
Step 2: *recover the time stamp for each instruction*
Step 3: *record the delay to offset library*
Processes execute:
Step 1: *Read instruction*
Step 2: *Decode instruction type*
Step 3: *Read offset value*
Step 4: *Allocate to data and instruction register*
Step 5: *Read data*
Step 6: *Execute instruction*
Step 7: *Write back*
End

The operational block diagram for the proposed approach is presented in Fig. 1 below.

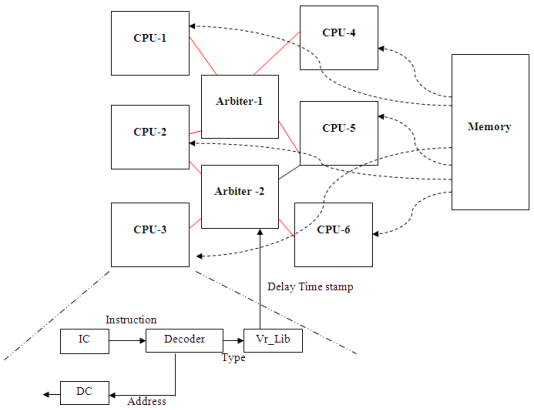


Fig. 1: System Architecture for the proposed MPSoC interface

The processor unit is handled in two operational phases, in which the update phase Phase-1 is performed when each of the decoded commands is mapped to the library unit. The arbiter is assigned a delay stamp for each type of instruction decoded, which internally assigns the delay value dependent on the clock cycle during execution. The procedure of computation and allocation is removed and the arbitration process is carried out on the basis of the scalar time delay value. This leads to a low system latency.

IV. SIMULATION RESULTS

This suggested work is validated in three stages of simulation, whereby the operational functionality of the proposed method is assessed for the scheduling of the proposed task. The allocation and operating overhead delay is calculated. The suggested technique is validated at the second level of implementation using the Xilinx ISE synthesiser for the FPGA device. This result shows the processing speed, area, latency and system performance. The final portion of the simulation result is analysed for the various instructional densities.

A) Operational verification functional

For time monitoring, the HDL description of the simulated particular job was generated in the Aldec tool. The created design is focused at Xilinx FPGA devices in order to implement the proposed method. Measurements are assessed for power, latency, throughput, and area. The results are shown below. Fig. 2 shows the set of instructions utilised to validate the proposed work. This technique works differently by using the instruction buffer set in the core CPU as the instruction cache. The processing instructions for each core unit are used to buffer the data collected from the main memory.

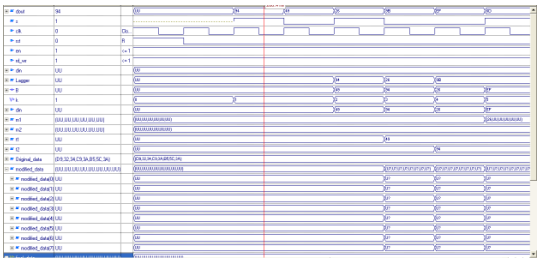


Fig. 2: Operational instruction used for testing

In the test process, each of the instruction is passed to the arbiter unit, where the instructions are mapped with the delay constraint as illustrated in Fig. 3.

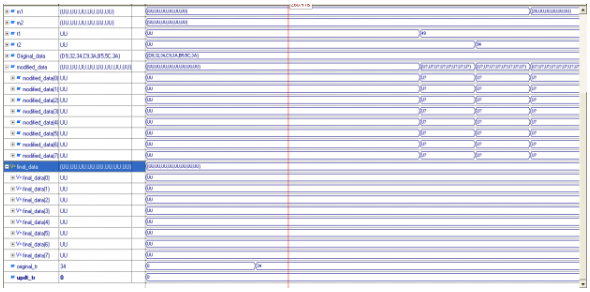


Fig. 3: Mapped instruction of delay metric at arbiter unit

The mapped clock pulse results in the creation of a new data, which is decoded in the arbiter as a delay instruction. Each register is assigned with the input and output clock delay value in the processing unit during execution. The mapped clock pulse results in a new data generated for each instruction and decoded as a delay instruction placed on the arbiter. Each processor unit is immediately allocated during processing when conducting an execution reading. The delay mapping and allocation procedure is shown in Fig. 4.

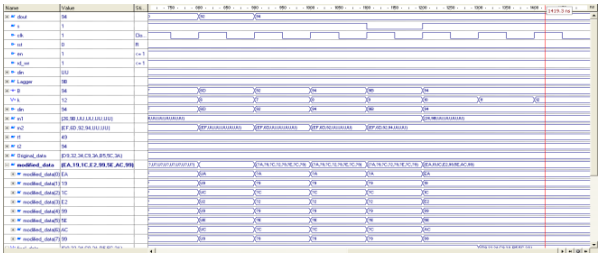


Fig. 4: Delay instruction allocation at arbiter unit

In the process of instruction execution for a 4 set of instruction, the latency for the proposed approach is observed to be 49 compared to 54 for VA-MPSoC design. The observation is illustrated in Fig. 5 below.

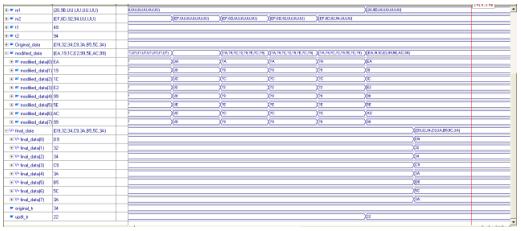
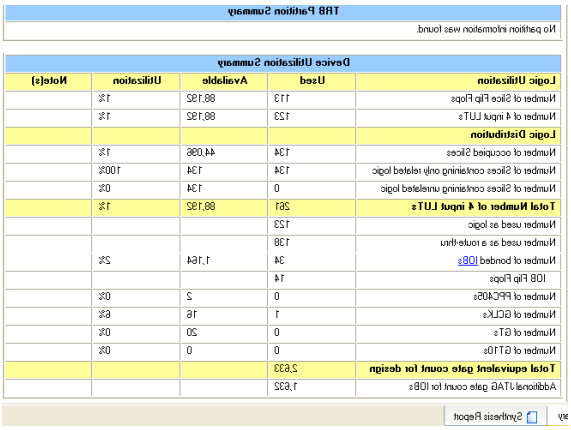


Fig. 5: Latency measurement for the developed system

B) Implementation result

The implementation of the developed approach is targeted to Xilinx FPGA device for a Spartan family. The implementation report obtained is presented in Fig. 6.



Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	113	88,192	1%	
Number of 4 input LUTs	123	88,192	1%	
Logic Distribution				
Number of occupied Slices	134	44,096	1%	
Number of Slices containing only related logic	134	134	100%	
Number of Slices containing unrelated logic	0	134	0%	
Total Number of 4 input LUTs	257	88,192	1%	
Number used as logic	123			
Number used as a route-thru	138			
Number of bonded IOBs	34	1,764	2%	
IOB Flip Flops	14			
Number of PPC405s	0	2	0%	
Number of BUFGCTRLs	1	16	6%	
Number of DSP48Es	0	20	0%	
Number of GT10Es	0	0	0%	
Total equivalent gate count for design	2,633			
Additional JTAG gate count for IOBs	1,632			

Fig.6: Report of Xilinx FPGA implementation for the developed system

A Power Analysis of the implemented design is computed using X-power analyzer of Xilinx tool. A power rating of 181mW of power rating is obtained.

```
Power summary:                                I(mA)      P(mW)
----------------------------------------------------------------
Total estimated power consumption:                         181
                                              ---
                                  Vccint 1.50V:       85       128
                                  Vccaux 2.50V:       20        50
                                  Vcco25 2.50V:        2         4
                                              ---
                                       Clocks:         0         0
                                       Inputs:         0         0
                                        Logic:         0         0
                                      Outputs:
                                         Vcco25        0         0
                                      Signals:         0         0
                                              ---
                          Quiescent Vccint  1.50V:       85       128
                          Quiescent Vccaux  2.50V:       20        50
                          Quiescent Vcco25  2.50V:        2         4
```

Fig. 7: X-power report for developed system

The timing report for the developed system illustrated a maximum operating frequency of 129.98MHz with a time period of 7.6ns. The device has a setup delay of 2.9ns and a hold delay of 3.6ns. The placement of the logical design with routing and area coverage is observed using Xilinx-route and place operation. The Interconnection

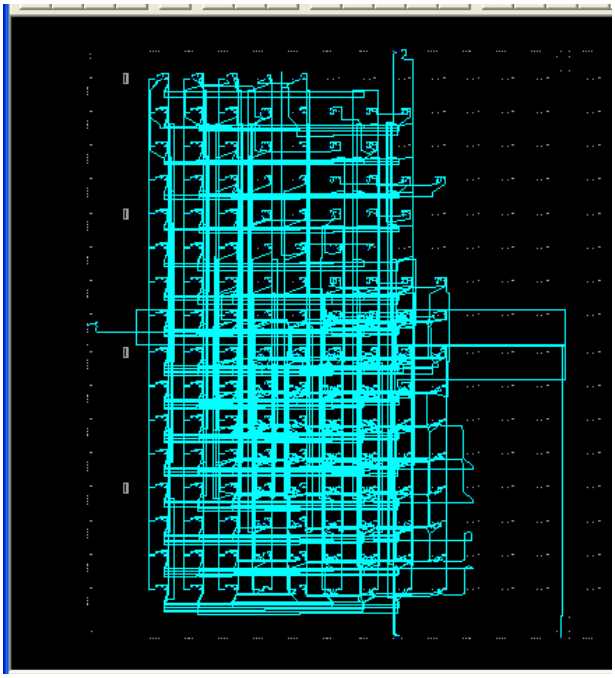


Fig. 8: Logical interconnect of CLB in targeted FPGA device

The logical placement of CLB unit is shown in Fig. 9.

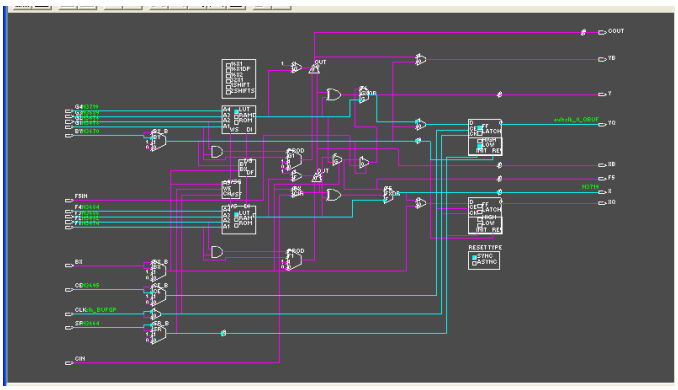


Fig. 9: Logical Placement of CLB unit

The pin layout for the targeted design is shown in Fig. 10. This implementation has dedicated lines of 12 IO lines with Vcc and ground pins as seen in Fig. 10.

The blue encircled are the allocated line here,

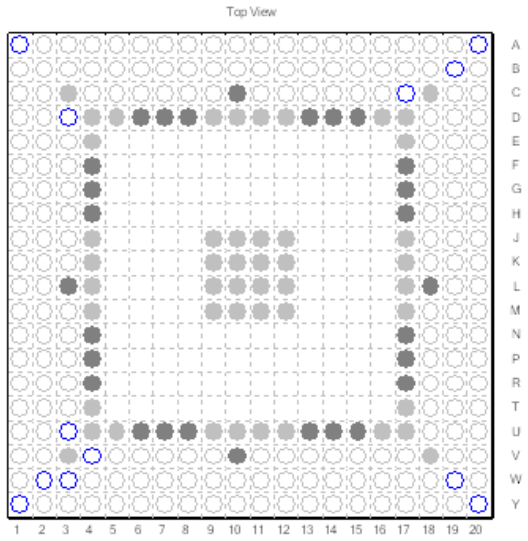


Fig. 10: Pin layout of the implemented design for the targeted FPGA

C) Analysis of developed approach

In the validation of operation performance power measure is critical. The power consumed in a processor unit is defined by,

(1)

Where is the capacitance, is the voltage, and is the operating frequency for a set of instruction executed. Here the power is defined as a function of device parameter and the operating frequency of the processing unit. Here, more the operation frequency is more the unit is enabling giving more dissipation of power. However, for reduced computations the operational iteration are reduced which leads to less number of operational cycles and hence reducing the power consumption. The analysis of the power utilization is presented in table 1 below.

Table 1: Observation for power utilization

Instruction density	Power (*mW*)	
	VA-MPSoC [19]	Vr-MPSoC
4	131.5	114.2
5	274.5	182.8
7	321.4	281.3
9	388.7	311.4
12	451.6	411.3

The comparison of power utilization for different instruction density is shown in Fig. 11

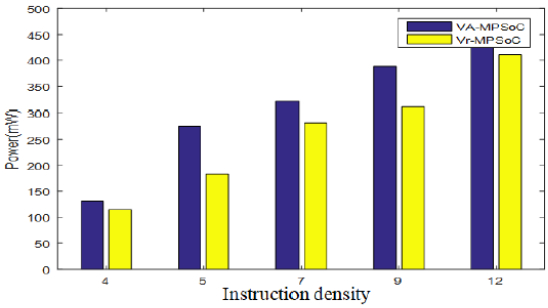


Fig. 11: Comparison of power utilization for different instruction density

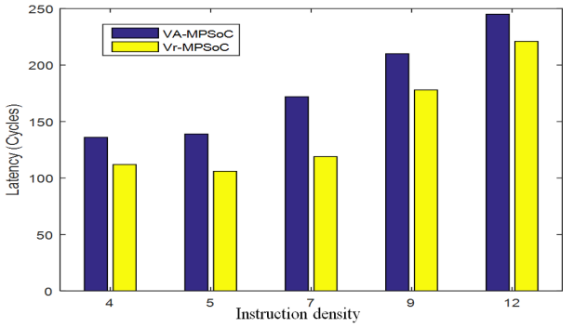
The power consumption is proportionately high for wide selection of instructional systems, however owing to low time computing cycles, the consumption is comparably smaller in the Vr-MPSoC architecture.

Latency is the number of calculation cycles used in a procedure. Table 2 below summarises the observed delay of the technique proposed.

Table 2: Latency observation for the developed approach

Instruction density	Latency (Cycles)	
	VA-MPSoC [19]	Vr-MPSoC
4	136	112
5	139	106
7	172	119
9	210	178
12	245	221

The comparison of latency for different instruction density is shown in Fig. 12



The system performance for a device design is validated by the efficiency of number of processing block per cycle which is termed as throughput. The throughput of a digital system is defined by,

$$THR = \frac{Fmax \times Bsize}{LAT} \qquad (2)$$

Where , and LAT are the maximum operating frequency, block size and latency measured.

Table 3: Throughput observation

Instruction density	Throughput	
	VA-MPSoC [19]	Vr-MPSoC
4	196.4	212.3
5	226.1	272.7
7	270.2	321.1
9	321.4	342.4
12	387.1	396.2

The comparison of the processing throughput for different instruction density is shown in Fig. 13

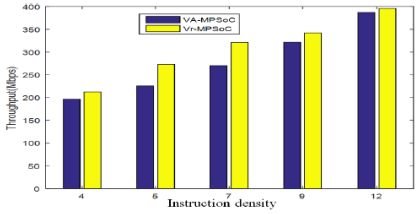


Fig. 13: Comparison of the processing throughput for different instruction density

V. CONCLUSION

This study introduced a novel technique to the mapping of chip multiprocessor system architecture (MPSoC). The distributed computing offers the benefit of quicker operation, but its functional performance is constrained by delays in resource allocation. In order to achieve optimum operating performance in spreading processing units, a novel clock time allocation virtualization with library mapping is introduced in the MPSoC architecture. The above technique significantly improves latency reduction and hence decreases electricity usage. This performance shows an increase in the system processing performance.

REFERENCES

1. L.Benini and G.DeMicheli, "Networks on chip: a new SoC paradigm". IEEE Computer, 35(1):70-78, January 2002.

2. T.Ono, M.Greenstreet, "A Modular Synchronizing FIFO for NOCs", Proceedings of International Symposium on Networks-on-Chip (NOCS), 2009

3. T.Chelcea, S.M.Nowick, "Robust Interfaces for Mixed-Timing Systems", IEEE Transactions on Very Large Scale Integration Systems, 12(8): 857-873, 2004.

4. D.Ludovici, A.Strano, G.N.Gaydadjiev, L.Benini, D.Bertozzi, "Design Space Exploration of a Mesochronous Link for Cost-Effective and Flexible GALS NOCs", Proceedings of Design, Automation and Test in Europe (DATE'10), pp. 679–684, Dresden, Germany, 2010.

5. C.Cummings, P.Alfke, "Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparison", SNUG-2002, San Jos`e, CA, 2002.

6. A.Edmanand, C.Svensson, "Timing Closure through Globally Synchronous, Timing Portioned Design Methodology", Proceedings of Design and Automation Conference (DAC), pp.71–74, 2004.

7. P.Caput, C.Svensson, "An On-Chip Delay- and Skew-Insensitive Multicycle Communication Scheme", IEEE Solid-State Circuits Conference (ISSCC), pp.1765–1774, 2006.

8. I.M.Panades, A.Greiner, "Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures", Proceedings of International Symposium on Networks-on-Chip (NOCS), pp.83–94, 2007.

9. D.Ludovici, A.Strano, D.Bertozzi "Architecture Design Principles for the Integration of Synchronization Interfaces into Network-on-Chip Switches", Proceedings of 2nd. International Workshop on Network on Chip Architecture (NoCArc), pp.31–36, New York City, NY, 2009.

10. D.Ludovici, D.Bertozzi, L.Benini and G.N.Gaydadjiev, "Capturing Topology-Level Implications of Link Synthesis Techniques for Nanoscale Networks-on-Chip", Proceedings of the 19th ACM/IEEE Great Lakes Symposium on VLSI (GLSVLSI), pp.125-128, 2009.

11. I.Loi, F.Angiolini, L.Benini, "Developing Mesochronous Synchronizers to Enable 3D NoCs", Proceedings of International Conference on VLSI Design, 2007.

12. D.Ludovici, A.Strano, D.Bertozzi, L.Benini, G.N.Gaydadjiev, "Comparing Tightly and Loosely Coupled Mesochronous Synchronizers in a NoC Switch Architecture", Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip, pp.244-249, 2009.

13. S.Stergiou, F.Angiolini, S.Carta, L.Raffo, D.Bertozzi, G.DeMicheli, "XPipes Lite: a Synthesis Oriented Design Library for Networks on Chips", Proceedings of Design, Automation and Test in Europe (DATE'05), pp.1188–1193, 2005.

14. F.Angiolini, L.Benini, P.Meloni, L.Raffo, S.Carta, "Contrasting a NoC and a Traditional Interconnect Fabric with Layout Awareness", Proceedings of Design, Automation and Test in Europe (DATE'06), March 2006.

15. "Specification of optimized GALS interfaces and application scenarios", GALAXY Project deliverable D3, online at http://www.galaxyproject. org/publ deliv.html

16. J.Ebergen, "Squaring the FIFO in GasP", Proceedings of International Symposium on Asynchronous Circuits and Systems, pp.194–205, 2001.

17. C.E.Molnar, I.W.Jones, W.S.Coates, J.K.Lexau, "A FIFO ring performance experiment", Proceedings of International

Symposium on Asynchronous Circuits and Systems, pp.279–289, 1997.

18. R.Apperson, Z.Yu, M.Meeuwsen, T.Mohsenin, B.Baas, "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 15(10), pp.1125–1134, 2007

19. Johannes Ax, Gregor Sievers, Julian Daberkow, Martin Flasskamp, Marten Vohrmann, Thorsten Jungeblut, Wayne Kelly, Mario Porrmann and Ulrich Ruckert, "CoreVA-MPSoC: A Many-core Architecture with Tightly Coupled Shared and Local Data Memories", IEEE Transactions on Parallel and Distributed Systems, Post-Print , December 2017.

Author Details:

SAMBA ANUSHA is pursuing Mtech in Holy Mary Institute of Technology & Science, Keesara, Bogaram, Ghatkesar Rd, Kondapur, Telangana 501301. Her interest areas VLSI.