International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 12 (2018) pp. 10741-10748

HW-SW co-design on Zynq SoC
Case Study: Simple Miniature DC Motor Speed Control System

Sheetal Bhandari * and Shashank Pujari**

**Pimpri Chinchwad College of Engineering, Savitribai Phule University of Pune, Pune, India*

***Pimpri Chinchwad College of Engineering, Savitribai Phule University of Pune, Pune, India.*

Abstract

A holistic approach to Embedded System Design towards product development is based on sound knowledge, skill and practice of variety of computer languages, software tools and hardware platforms. The paper investigates the applicability of Hardware-Software co-design approach using Xilinx All Programmable System on Chip (SoC) Zynq platform based on a case study of a simple miniature DC motor closed loop speed control system in laboratory environment. The experimental result shows that HW-SW codesign with judicious partitioning of tasks sharing between Processor and Logic sections of Zynq SoC leads to optimum design in terms of resources, power and clock constriants.

Keywords: Zynq, SoC, Hw-Sw codesign, DC motor control

INTRODUCTION

Embedded System Design usually require a combination of parallel and sequential tasks, with data flow being predominantly parallel and control structures predominantly sequential. Processor based design can handle sequential tasks. Real time tasks demand a RTOS running on the Processor to achieve parallel processing of tasks. Real time processing, like RTOS based system, can also be achieved without much overhead on CPU time and system memory by prioritized interrupt driven tasks handling in small embedded system. In Programmable Logic design, all tasks can run concurrently at the speed of system clock, while sequential tasks handled by state machine.

Processor based design has the advantage of available software manpower, host of library resources, large range of processor vendor's offerings and short time to market embedded product design as compared to Programmable Logic, which requires long learning curve, costly product customization and long time to market. Programmable Logic based design has the advantage of high flexibility to designer for proof of concept at formative stage of the system design leading to manufacturable prototype at a later stage before the final ASIC implementation.

Hardware designer have familiarity of writing VHDL code for Programmable Logic based project, whereas avoiding a processor to build embedded systems is a difficult task for the software designer, who is familiar with conventional practice of writing code based on the instruction set, compiling the code and running on the processor. System design On Programmable Chip(SoPC) is a balanced approach between Processor only (Microcontroller) and Programmable Logic only (FPGA) for embedded product design [1][2]. The term SoC and SoPC are used interchangeably.

The SoC design with embedded hard core processor involves both software design in Processor Section (PS) and hardware design in Programmable Logic (PL) section. To migrate to a complete SoC design, some of the functions implemented in programmable logic in RTL code can be transferred to processor section. The decision on sharing of the functions/tasks by the PS (SW) and PL(HW) is judiciously done looking into the characteristics of the tasks and this process is termed as hardware-software co-design. as depicted in Fig. 1.

The case study taken up is a simple miniature DC motor closed loop speed control system in laboratory environment. The scope includes development of lab prototype model with available Components Off the Self (COTS) using DC motor with speed encoder and implementation of HW_SW co-design methodology on Xilinx All Programmable SoC Zed Board.

The paper is organized as; section II covers DC Motor Speed Control System section III covers Tasks partitioning of Motor Speed Control System, Section IV describes the Zynq SoC Design Platform, Section V covers Tasks to Architecture Translation and Section VI covers Architecture Mapping Analysis. Conclusion and References are in the concluding section.

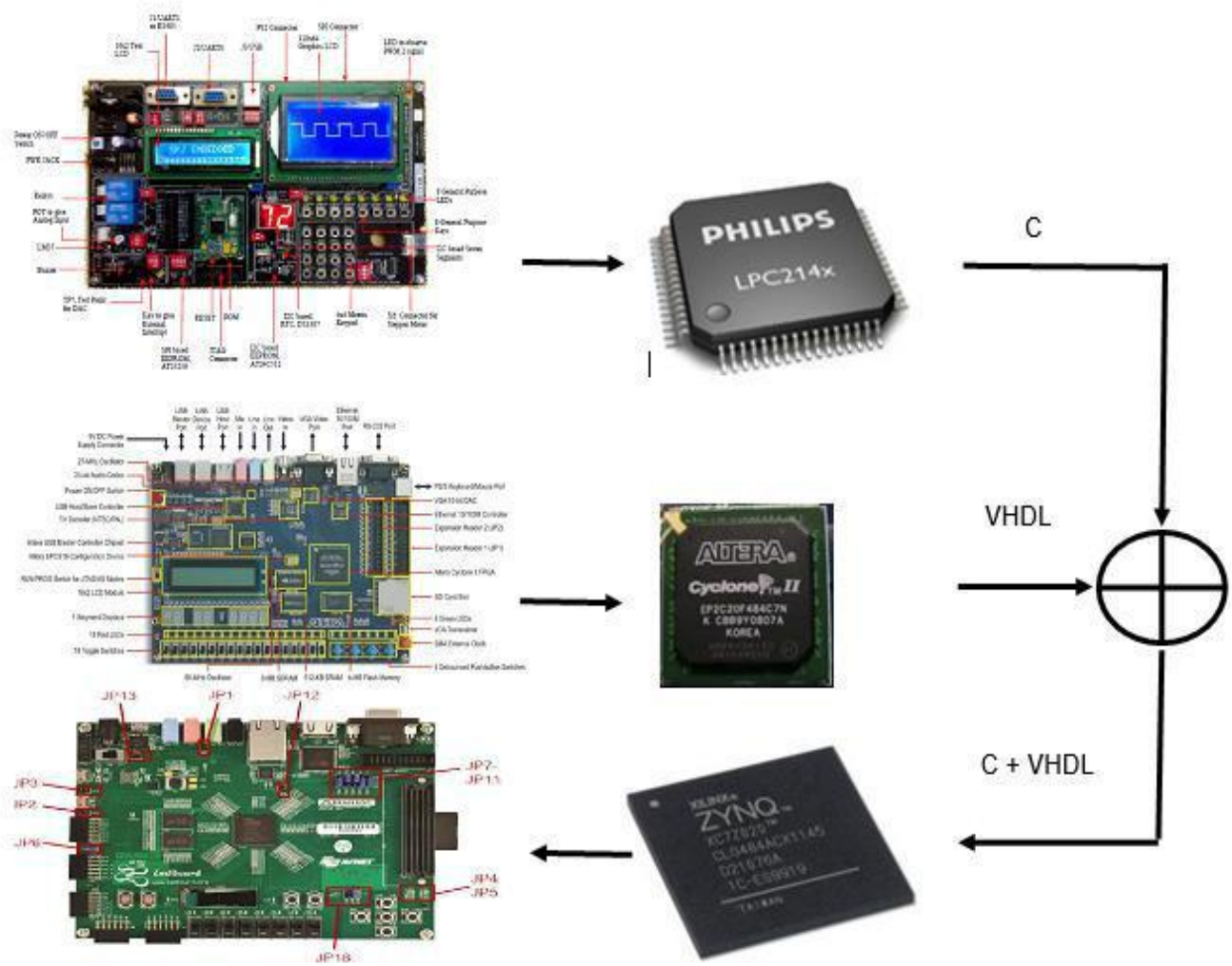


Figure 1. Microcontroller, FPGA and SoC HW-SW Co-design

DC MOTOR SPEED CONTROL SYSTEM

The Industrial DC motor drive control system is shown in Fig. 2. Generally, the Industrial drives have PID control system. The control system has two loops, an outer speed loop and inner current loop. The speed loop maintains the speed of the motor at various load condition. The speed loop is Proportional Integral (PI) control. The speed PI control works on the error difference between the desired reference speed and actual feedback speed of the DC motor. The PI control output is used as current reference to following current controller. The current loop is Proportional Derivative (PD) control. The current PD control works on the error difference between the reference current generated by the speed PI controller and the actual armature current feedback of the DC motor drive. The PD control output is supplied to the following triggering circuit of the power drive stage which could be high AC powered Thyristor or MOSFET or IGBT multi bridge rectifier. Both PI and PD control system has over-voltage and over-current limit protection including an automatic suicide stage to mitigate control failure in emergency condition.

The focus of this paper is not the design of such a complex industrial DC motor drive system. The primary goal is to design a System on a Programmable Chip (SoPC) solution based on the experience gained after a Microcontroller [3] and a FPGA based design [4]. However, the essence of full-fledged closed loop control system has been retained in the scaled down version of the lab prototype model of a simple miniature DC motor closed loop speed-only control system without any load condition.

The simplified model of DC motor control system is shown in Fig. 3. The motor speed encoder generates pulses at rotational speed of the motor. The pulses are sampled every second by the speed sampler logic. The sampled speed is compared with set speed. The difference between set speed and actual speed is used to vary the PWM duty cycle, which in turn vary the voltage across the motor so as to control the speed of the motor as per the set speed. The set and actual speed of the

motor as well as the status of the speed comparator outputs namely equal, greater or lesser are displayed.

The laboratory model of the DC motor mechanical system is shown in Fig. 4. The DC motor is a 6 Volt DC motor. The low power motor driver electronic system is shown in Fig. 5. The motor driver is a two stage transistors comprising of a switching transistor BC 457 and low power transistor SL100 to drive sufficient current to rotate the motor. The average voltage across motor decides speed of the motor. The speed encoder is made out of a wheel with 8 holes, mounted on geared motor shaft, and IR transmitter/receiver generates 8 pulses per one revolution of the motor. The IR LED based speed encoder generates pulses in proportion to the rotation of the motor.

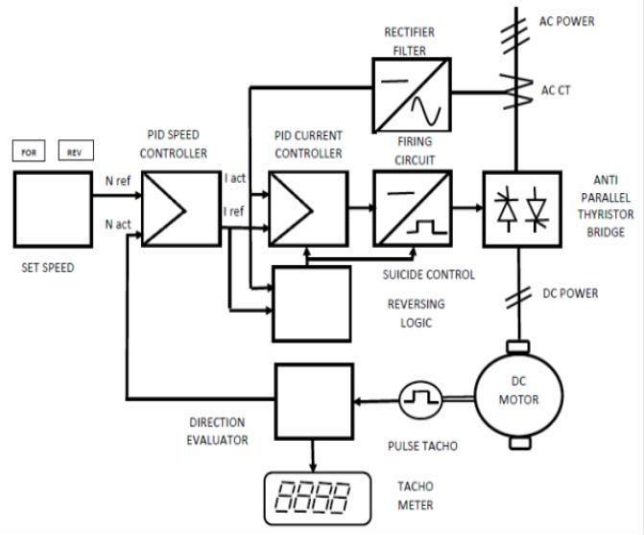


Figure 2. Industrial DC Motor Drive System

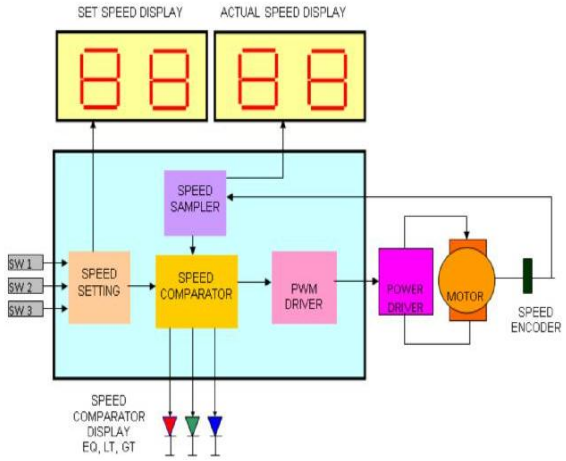


Figure 3. Simplified DC Motor Speed Control System

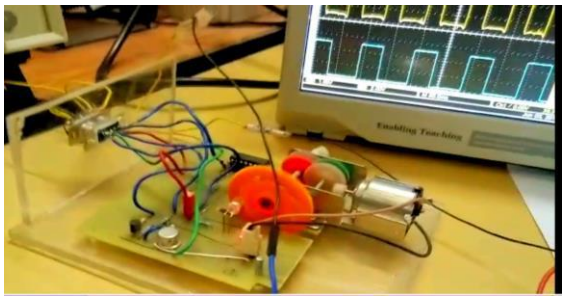


Figure 4. Lab prototype model of DC Motor with speed encoder

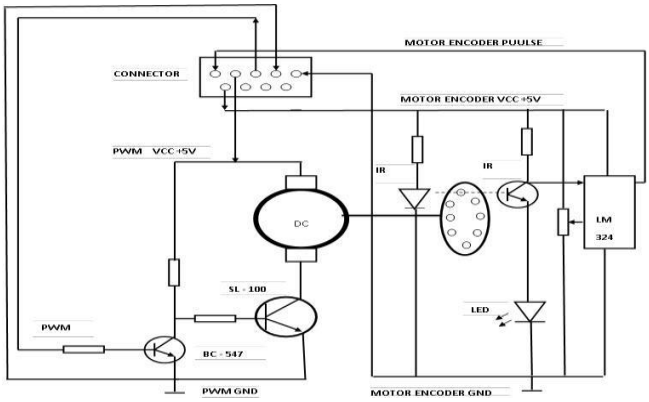


Figure 5. DC Motor drive circuit

TASK PARTITIONG

The Motor Speed Control System can be partitioned into five major components listed below and shown in Fig-6.

1. Human Machine Interface (Switch, Display on LED and 7 Segment, Host Communication)
2. Speed sampler
3. Speed controller
4. Speed actuator
5. Speed sensor

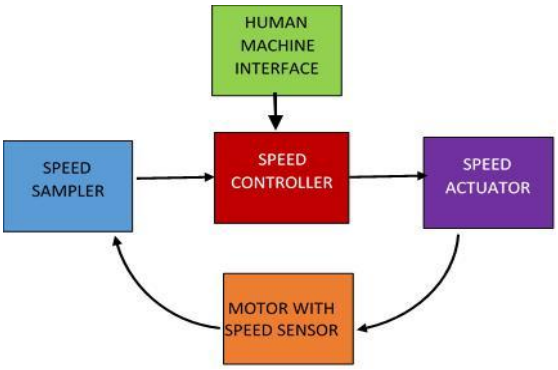


Figure 6. Tasks of Motor Control System

The tasks are

Task1 – Speed Sampler – Periodic at interval of 1 sec

Task2 – Speed Controller – PWM Controller at rate of PWM frequency – 1 milli sec (Electrical Time Constant of motor) Task3 - Human Machine Interface (Switch, Display on LED and 7 Segment, Host Communication)

Task1 and Task2 are real time periodic tasks at different periodicity and Task3 is aperiodic. The periodicity of Task1 is 1 sec if we consider sample the speed of the motor in terms of rotation per second. The periodicity of Task2 is in the order of millisecond which depends on the electrical time constant of the motor.

The top level entity of the Motor Control system is shown in Fig-7. Switches and buttons are used as human machine interface to set the motor speed and PWM clocks frequency. LEDs display the status of the speed error. Speed clock is the speed encoder output and PWM is the signal driving the Motor driver with variable duty cycle to regulate the motor speed.

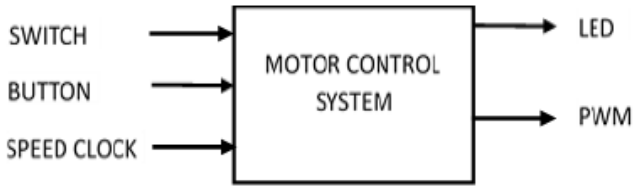


Figure 7. Signals of Motor Control System

ZYNQ SOC DESIGN PLATFORM

The Zynq architecture [5] comprises of two main parts as shown in Fig-8, a Processing System (PS) formed around a dual-core ARM Cortex-A9 processor, and Programmable Logic (PL), which is equivalent to that of an FPGA. It also features integrated memory, a variety of peripherals, and high-speed communications interfaces.

The PS has a fixed architecture and hosts the ARM processor and system memory, whereas the PL is completely flexible, giving the designer a 'blank canvas' to create custom peripherals, or to reuse standard ones. The interconnections are implemented via AXI interfaces linking the PS and PL.

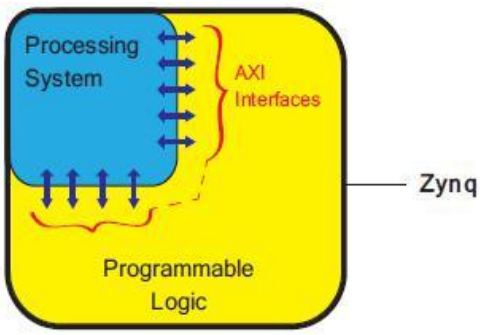


Figure 8. Zynq SoC System

The PL section is ideal for implementing high-speed logic, arithmetic and data flow subsystems, while the PS supports software routines and/or operating systems, meaning that the overall functionality of any designed system can be appropriately partitioned between hardware and software.

A key element of the system design stage is to partition the intended functionality appropriately between software and hardware, and to define the interfaces between the two sections. It is possible that this partitioning will subsequently be adjusted as the designers iterate towards optimum system design. Having partitioned the system, software and hardware development can then progress in parallel. In terms of hardware development, the task is to identify the necessary functional blocks to achieve the design, and to thereafter assemble them through some combination of design reuse and new IP development, and make appropriate connections between the blocks. Similarly, the software aspect of the project can be realised through developing custom code or by reusing pre-existing software. Verification of both software and hardware will be required, and this forms an integral and important part of the process. Lastly, the hardware and software elements of the system must be integrated according to the interfaces defined at the specification stage, and further 'whole system' testing undertaken.

The process of designing a Zynq system is around Xilinx Vivado™ Integrated Development Environment (IDE). The IP Integrator environment of Vivado IDE is used for the generation of a Zynq processor design and implemented on the ZedBoard. The Software Development Kit (SDK) is used to create the software application which run on the Zynq's ARM Processing System (PS) to control the hardware that is implemented in the Programmable Logic (PL).

TASKS TO ARCHITECTURE TRANSALTION

Architecting the five tasks and the associated signals into SoPC design can be accomplished by judicious sharing of the tasks by PS and PL sections in three stages as the learning of SoPC design process on Zynq platform [6] progresses.

Architecture 1

All the five tasks and signals interfaces are wholly implemented in PL section. This design is merely a Programmable Logic HW design with the coexistence of idle ARM processor without any contribution. This approach has no practical relevance and costly, except to gain experience of the SoPC design environment for making a workable Motor control IP (MCP). PS section provides bare minimum clock and reset to PL section. The AXI interface is dummy as it appears by default while creating and packaging MCP.

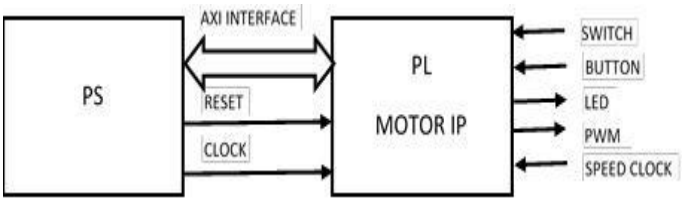


Figure 9. No tasks sharing between PS and PL, PL executes all tasks

The Zynq system Architecture 1 is shown Fig-10. The MCP core is generated using Xilinx Vivado IDE based on the earlier FPGA proven RTL code.

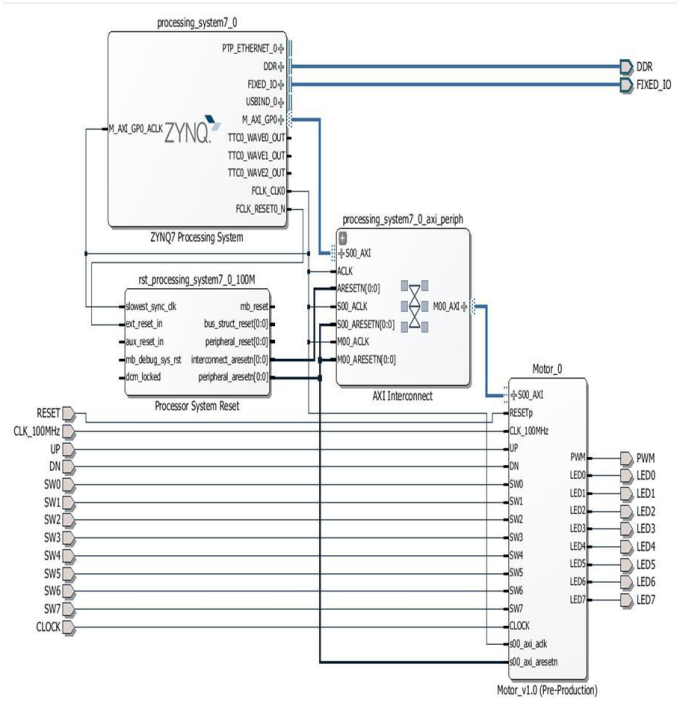
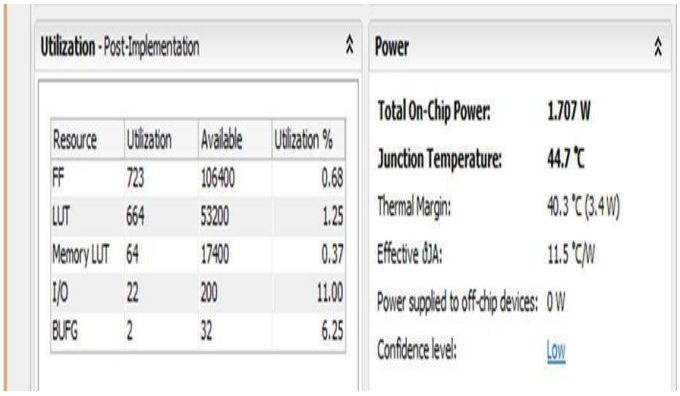


Figure 10. Zynq System Architecture 1

Fig-11 to Fig-13 shows the resource usage, power consumption and timing result of Architecture 1.



Resource	Utilization	Available	Utilization %
FF	723	106400	0.68
LUT	664	53200	1.25
Memory LUT	64	17400	0.37
I/O	22	200	11.00
BUFG	2	32	6.25

Figure 11. Tabular Result of resources usage and power of Architecture 1

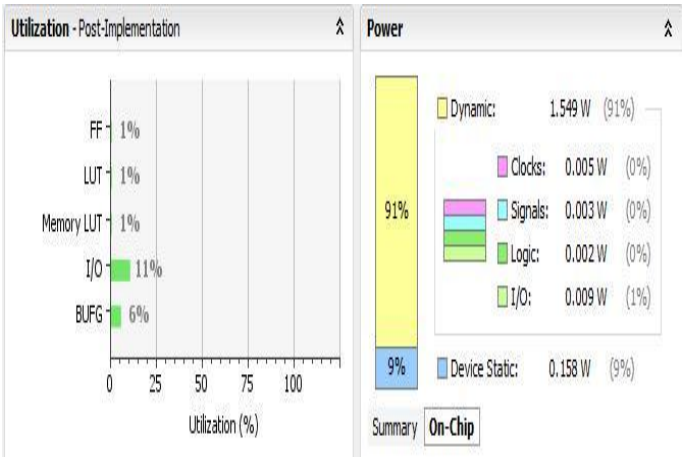


Figure 12. Graphical Result of resources usage and power of Architecture 1

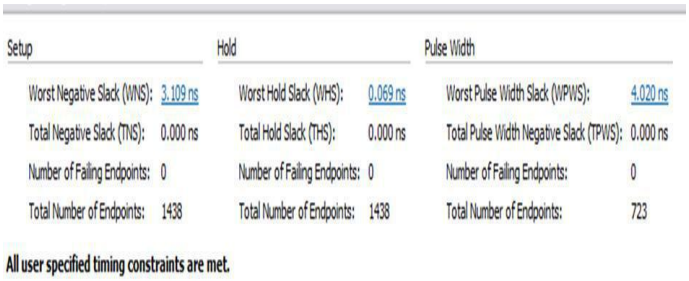


Figure 13. Timing result of Architecture 1

Architecture 2

This is HW and SW co-design. Partial tasks and signal interfaces are transferred to GPIO section of ARM. The hardware HMI interfaces via the GPIO are available as software functions in the processor section and used by the MCP. This approach has minimal introduction of the SDK environment in terms of simple HMI function implemented in software. The two real time tasks i.e., speed sampler and speed controller are retained in the MCP. The set speed decided by the switch selection and the actual speed sampled by the speed sampler are passed to the processor section via AXI interface for speed control and HMI display. This approach has practical relevance as a workable solution to Motor Control System design.

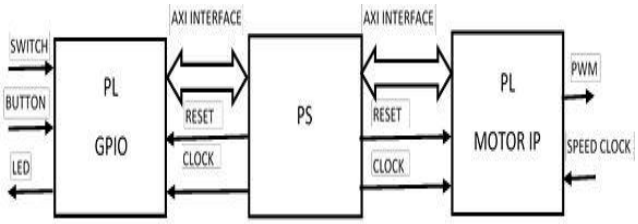


Figure 14. Partial tasks sharing between PS and PL, PS handles aperiodic HMI tasks PL executes time critical tasks

The Zynq system Architecture 2 is shown Fig-15. As seen the MCP has all the input and output signals except Switches and LEDs. Switches and LEDs are connected to GPIOs peripheral of the ARM processor to be used as HMI functions by the Processor section.

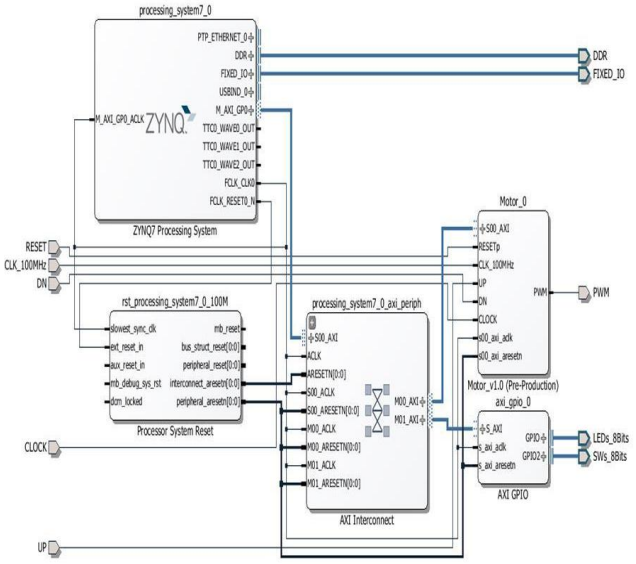


Figure 15. Zynq System Architecture 2

Fig-16 to Fig-18 show the resource usage, power consumption and timing result of Architecture 2.

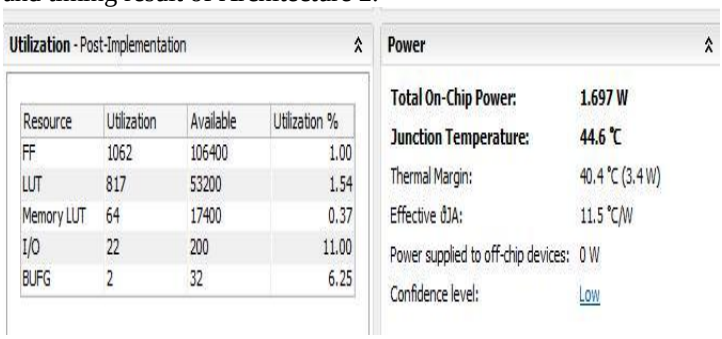


Figure 16. Tabular Result of resources usage and power of Architecture 2

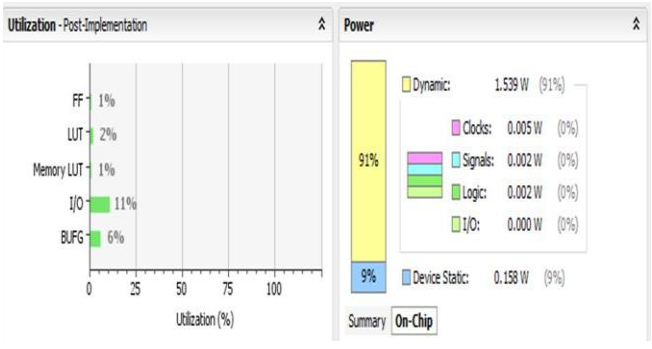


Figure 17. Graphical Result of resources usage and power of Architecture 1

Architecture 2

Figure 18. Timing result of Architecture 2

Architecture 3 :

The MCP in architecture 2 is replaced by the TIMRER_COUNTER functions of the ARM CPU. Rest of the tasks remain same as in architecture 2. This is a complete SW only design. This approach has maximum use of the SDK environment with interrupts handling. The two real time tasks i.e., speed sampler and speed controller are implemented using interrupts service routines. This approach has practical relevance as a workable solution to Motor Control System design without the use of any proprietary MCP.

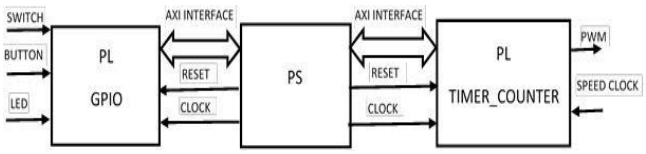


Figure 19. All tasks handled by PS

The Zynq system Architecture 3 is shown Fig-20. As can be seen there is no MCP. All the tasks are handled by the ARM Processor. This approach has maximal usage of the ARM processor in Software only environment with interrupts handling. All the signals are distributed over three GPIO peripherals connected to ARM processor. There are two timers. One sec timer is used for sampling speed at 1 sec interval and the second timer is used to generate PWM. The two real time tasks i.e., speed sampler and speed controller are handled by the two timer interrupts.

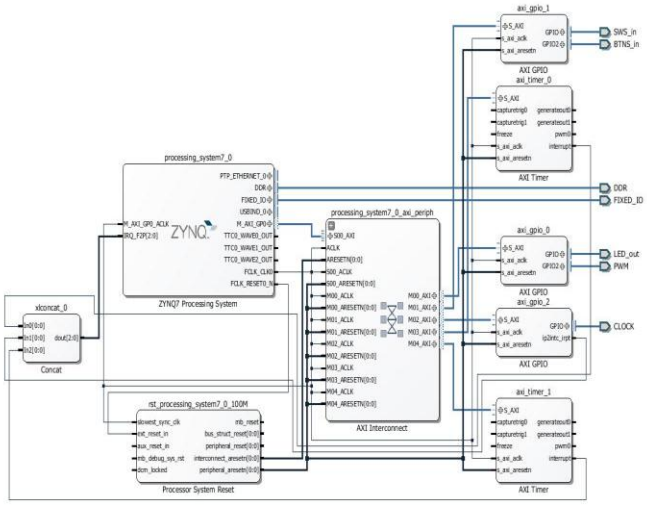


Figure 20. Zynq System Architecture 3

Fig-21 to Fig-12 show the resource usage, power consumption and timing result of Architecture 3.

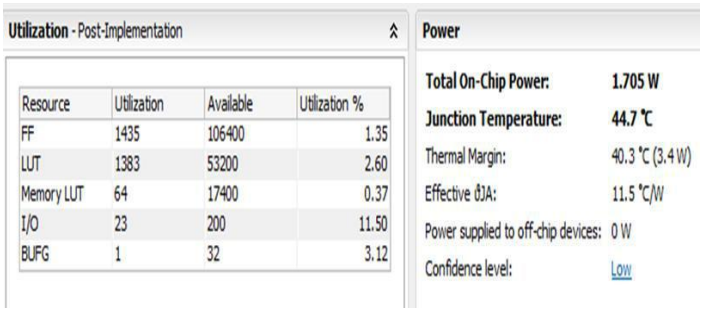


Figure 21. Tabular Result of resources usage and power of Architecture 3

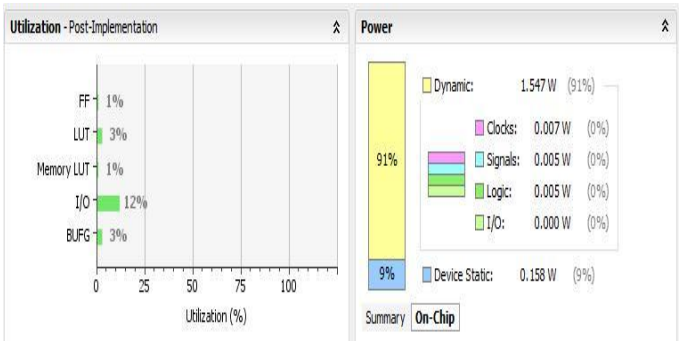


Figure 22. Graphical Result of resources usage and power of Architecture 3

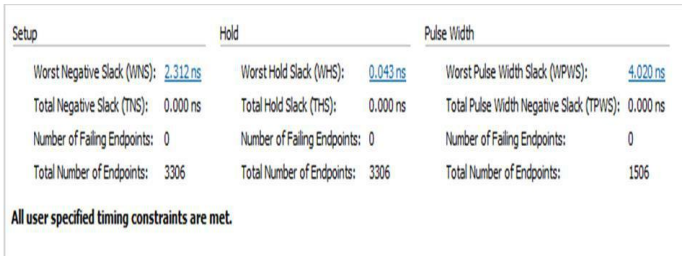


Figure 23. Timing result of Architecture 3

ARCHITECTURE MAPPING ANALYSIS

Architecture 1: - Programmable Logic section handles all the five tasks. In Programmable Logic design, all tasks can run concurrently at the speed of system clock, while sequential tasks can be handled by state machine. The FPGA resource requirement for this motor control logic experimented in [4] is just 165 functional slices and 59 registers, which can fit into a small size CPLD.

Architecture 2: - This is a HW-SW co-design methodology. The real time tasks are handled by the MCP in Programmable Logic section while the non-real time sequential tasks are handled by the ARM processor. The benchmark in Table-I shows this architecture is optimum.

Architecture 3: - The ARM Processor handles all the five tasks i.e. two real time tasks and three non-real time tasks. The real time tasks are handled by the interrupt generating built in peripherals Timers and PWM controller of the embedded ARM processor. This architecture tantamount to a Microcontroller based design.

Table I shows the resource usage, power consumption and clock end points of all architectures. It can be concluded that Architecture 2 is optimum. Architecture 2 is HW-SW codesign with the judicious partitioning of tasks sharing between Processor and Logic sections of Zynq SoC.

Table I. Benchmark of Architectures

	Architecture-1	**Architecture-2**	**Architecture-3**
Resource Nos. of FF	723	1062	1435
Resource Nos. of LUT	664	817	1383
Power in Watt	1.707	1.697	1.547
Nos. of Clock nodes	1438	2097	3306

CONCLUSION

System Design On Programmable Chip is an easy migration path from Processor based design to FPGA based Embedded Systems Design. Embedded processor in FPGA come in very handy in implementing sequential instruction based processing similar to Microcontroller based design. The Programmable Logic section can handle time critical concurrent tasks effectively. Judicious partitioning of tasks leads to HW-SW co-design in Zynq SoC platform, which is optimum in terms of resource, power and timing, compared to Hardware only or Software only design.

To address the needs of increasingly complex embedded systems, semiconductor vendors are offering a rapidly growing portfolio of System-on-Chip solutions with heterogeneous architectures. Xilinx SDSoC is one such an IDE comprising of Vivado, HLS and SDK tools for SoC design. Present work is based on Xilinx® XC7Z020-1CLG484CES Zynq-7000 AP SoC Zed Board [7].

REFERENCES

[1] Dick Selwood, Heterogeneous Processing, SoCs and FPGAs, A Path to Flexible System Implementation, Electronic Engineering Journal. October 29, 2015

[2] J. O. Hamblen, T. S. Hall Using System-on-a-Programmable-Chip Technology to Design Embedded Systems, IJCA, Vol. 13, No. 3, Sept. 2006

[3] Shashank Pujari, Amruta Panda, Prangya Paramita Muduli, Rasmita Badhai, Sofia Nayak, Yougajyoty Sahoo, "A Learning Model for 8051 Microcontroller Case Study on Closed Loop DC Motor Speed Control", International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 11, November 2013, ISSN 2250-2459, ISO 9001:2008 Certified Journal,

[4] Shashank Pujari "A simple closed loop dc motor speed control system on FPGA platform for VHDL beginner" International Journal of Scientific & Engineering Research Volume 4, Issue3, March-2013, ISSN 2229-5518

[5] "The Zynq Book, Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC" design by by Louise H Crockett, Rose A Elliot, Martin A. Enderwitz, Robert W. Stewart

[6] "The Zynq Book Tutorial" by Louise H Crockett, Rose A Elliot, Martin A. Enderwitz , Robert W. Stewart

[7] ZedBoard_HW_UG_v1_9.