© 2014 IJIRT | Volume 1 Issue 11 | ISSN: 2349-6002

Survey Paper on Designing of Portdog Using FPGA

Nilesh U. Kagde[1], Prof. Nandini Dhole[2], Prof. Chittaranjan P. Mahajan[2]
[1]*Department of Electronics & Telecommunication, R.M.D Sinhgad School of Engineering, Pune, India*
[2]*Director, Dolphin labs, Pune*

***Abstract*- Embedded system is having very important role in today's fastly developing in automation and industrial area. We can say embedded system is "Computer in Disguise" system, carrying along a computer or processor based system that which cannot be programmed by users. . hardware, software and firmware are some major components of embedded system and "Time to Market" is the key for successful launching of the product in today's rapid development world for capturing the market. The current trend for handheld device is to provide the users various embedded multimedia applications. These new applications constrain architecture developers to embed dedicated hardware accelerators in order to meet the application timing requirements. However the use of dedicated monolithic hardware accelerators is onerous to achieve due to physical and economical constraints. When the multimedia applications share common functionalities, monolithic hardware accelerators could be split into smaller accelerators in order to cut down redundancy in hardware implemented functionalities and save silicon area. Nevertheless lowering the granularity of accelerator will increase synchronization calls between the main processor, and the accelerators. Hardware acceleration is important for any handheld device in today's market.**

This paper presents one of the new concept of portdog for the hardware acceleration. To make the processor faster and to accelerate its processing speed we have to do multitasking. So for certain processes we can use this portdog for checking the outputs of real time system. For that particular time period we can use our main processor for some another task or we can us sleep mode for main processor. For this portdog can be used. Portdog can work as like co processor of the main processor. Using Xilinx the coading of this portdog can be completed and it can be implemented using FPGA Spartan 3E kit.

***Index Terms*- Portdog, Real time system, microprocessor, embedded system.**

I. INTRODUCTION

Hard real-time operating systems are mainly designed for uniprocessors. Time constrains are most important in hard real time system. If task cannot completed in its given time total system may get failure. System fails in such conditions. This system is used when task or event should be processed within a strict deadline. Such strong guarantees are required in systems for which not reacting in a certain interval of time would affect the system and cause great loss. Some examples of hard real-time systems are such as traffic control, industrial control, robotics, avionics, etc [1].

We can say that, An embedded system is 'system with constraint'. Design and development of highly constraints system is major challenge. Some constraints of embedded system are like small size, low weight is constraint which is applicable in hand held electronics, transportation applications and other devices. Another is low power that is device should use minimum power which is for battery powered applications; limited cooling etc. harsh environment is constraint which we have to consider in power fluctuation, RF interference, lighting, heating, vibration, shock etc. Many embedded systems are interactive and processor based system. Processor architectures are designed and that are developed to reduce system development time and to improve performance, throughput, energy saving and total system efficiency. Embedded application tasks in software are also very critical aspect. Due to this reason it specifies the need to look after co-design aspect. RISC architecture is dominating and it support for lot of applications designing in embedded systems. The risk architecture has less powerful instruction sets.

The basic fundamental study shows that the design and operations of embedded systems are separately partitioned in to software and hardware. This aspects suffers with various trade-offs in design metrics such as to make system flexible needed some more tasks to be performed in the software. So the size of software increases. If size increase memory requirement for that is also increases which in turn increase the size of hardware required for that memory, power consumption of the hardware and cost of total system. The system throughput is affected by various inherent latencies generated due to processor architecture. To avoid this situation various steps are to be taken like pipeline concept, Brach perdition etc. but still some challenges are present which required to pay attention like size of hardware and weight solution is reduction in memory requirement reduced OS footprints etc. Other is speed or response time solution for this is custom instruction set, hardware acceleration, multi core technique etc.

By using the hardware acceleration we can improve the speed of system. Hardware implementation of system is faster than that of software. So many systems are now building on hardware so hardware acceleration is needed in such system.

To make the processor faster and to accelerate its processing speed we have to do multitasking. So for certain processes we can use this portdog for checking the outputs of real time

system. For that particular time period we can use our main processor for some another task or we can us sleep mode for main processor. For this portdog can be used. Portdog can work as like co processor of the main processor. Using Xilinx the coding of this portdog can be completed and it can be implemented using FPGA Spartan 3E kit.

Portdog can be used to improve the hardware acceleration. It is itself hardware so its speed is much faster than the software. Due to this it is having much more scope in future.

II. HARDWARE ACCELERATION

A. Embedded system

An embedded system is an engineering artifact which is involving computation that is subject to physical constraints. The physical constraints can be arises by two different kinds of interactions of computational processes with the real physical world, one is reaction to a physical environment and another is execution on a physical platform. Accordingly, the two types of physical constraints are reaction constraints and execution constraints. Common reaction constraints specify some of the deadlines, throughput, and jitter; they originate from the behavioral requirements of the system. Common execution constraints put bounds on available processor speeds, power, and hardware failure rates; these are originated from the implementation requirements of the system. Reaction constraints can be studied in control theory and the execution constraints can be studied in computer engineering. Gaining control of the interplay of computation with the above both kinds of constraints, so as to meet a given set of requirements, is the key to embedded systems design. [2]

B. Embedded systems design

Embedded systems consist of different parts such as hardware, software, and an environment for which system is to be designed. These are in common with most computing systems. However, there is an essential difference between embedded and other computing systems: since embedded systems involve computation that is subject to physical constraints, the powerful separation of computation (software) from physicality (platform and environment), which has been one of the central ideas enabling the science of computing, does not work for embedded systems. Instead, the design of embedded systems requires a holistic approach that integrates essential paradigms from hardware and software design and control theory in a consistent manner. We can say that this approach may not be an extension of hardware and software design, but must be based on a new foundation that subsumes techniques from both worlds. Due to this current design theories and practices for hardware and software, are tailored towards the individual properties of these two domains; indeed, they often use abstractions that are diametrically opposed. We have to look at the abstractions that are commonly used in hardware design, and those that are used in software design.

C. Hardware acceleration in embedded system

Now days the handheld devices have to provide the users various embedded multimedia applications in single device. So the new applications constrain architecture developers to embed the dedicated hardware accelerators for the meeting of the timing requirements for particular application. So the use of particular dedicated monolithic hardware accelerators is onerous to achieve due to physical as well as economical constraints. When some common functionality are shared by the more multimedia applications, that time monolithic hardware accelerators could be spread into different smaller accelerators so that it cut down the redundancy in hardware implemented functionalities and so that it saves silicon area. Nevertheless it lowering the granularity of accelerator will increase synchronization calls which are between main processor and the accelerators.

Handheld devices integrate more and more functionality, and providing more multimedia applications is becoming a de facto requirement. Solutions are therefore needed for accelerating these computationally intensive applications in order to fulfill the requirements. The main acceleration approaches can be classified into two categories.

1. A short portion of code is accelerated by extending the processor instruction set with a corresponding instruction. In this case the new instruction has typical execution latency from 1 to 4 cycles, thus limiting the size of the accelerated software. Developing longer instruction would make the pipeline execution flow inefficient.
2. Full application functionality is accelerated with a monolithic hardware accelerator used as peripheral device. The hardware accelerator is then synchronized with the application by the means of interrupts. In this case the hardware accelerator has typical execution latency from several thousand cycles up to several hundreds of thousands of cycles. However dedicated monolithic hardware accelerators are onerous to achieve due to physical and economical constraints. [3]

The use of fine grained hardware accelerators has the advantage of saving silicon area by allowing collaborative use of common accelerated functionalities among several applications, thus cutting down implementation redundancy over several accelerators.

III. PROPOSED CONCEPT OF PORTDOG

Hardware acceleration is important factor in design of embedded real time applications. In these devices time constraints for completion of tasks or events in the system speed of processor should be high. Many time processors have to check the inputs which are from real time physical system or environment. So that for that particular task processor get busy and it cannot Handel other task at that time. So that energy loss takes place in system. Total system will need more energy. To avoid this situation portdog concept is proposed in this paper. It will work like dog. As dog give signal to human in his language if thief is around same way this portdog will give signal if some change occurs as per our desired change in

the real time physical system. So that this portdog can be used as coprocessor of the main processor. This will continuously check for the input signal from the real time system which we want to control. By continuous checking of input signal if some change occurs then this portdog will give output signal which we can use as control signal for the main processor. This protdog is to be implemented on hardware so that its speed will be more than that of software because software takes more time to execute program that that of hardware. Hardware execution is much faster than software so that acceleration and speed of system get increase and that is most important for today's rapidly development in embedded system devices. Due to this efficiency of system increase and also less power is required. So this portdog concept will increase the efficiency of system, save the power of system so that energy consumption will be reduced. This concept is shown in below figure 1.

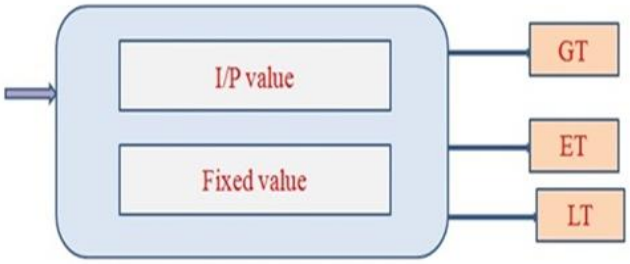

Figure 1 Conceptual block diagram of Portdog
[Courtesy Mr. Chittaranjan Mahajan, Dolphin labs, Pune]

Figure 1 shows the basic concept of portdog. Here we can measure the given input with our fixed input. Depend upon that we can give output as greater than, less than or equal to. So depend on our requirement we can use any one or all signals for our controlling system. In this port dog we can also check the status of ports. Due to this we can make our main system in sleep or ideal mode as per requirement and so that power consumption of system will be less. If we want we can assign another task to system during the period of this checking time. We our required signal if there from portdog that time we can bring our system in active mode. Due to this power consumption will be reduced and efficiency of system will be increase.

IV. INTRODUCTION TO FPGA

This proposed portdog can be implemented in FPGA using Xilinx Spartan 3E family. Coding for this portdog can be done in Xilinx VHDL language. This is the future work of this paper. A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). Contemporary FPGAs have large resources of logic gates and RAM blocks to implement complex digital computations. As FPGA designs employ very fast I/Os and bidirectional data buses it becomes a challenge to verify correct timing of valid data within setup time and hold time. Floor planning enables resources allocation within FPGA to meet these time constraints. FPGAs can be used t implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of a portion of the design and the low non recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications.

Figure 2 FPGA from Xilinx

FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together" – somewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. Some FPGAs have analog features in addition to digital functions. The most common analog feature is programmable slew rate and drive strength on each output pin, allowing the engineer to set slow rates on lightly loaded pins that would otherwise ring or couple unacceptably, and to set stronger, faster rates on heavily loaded pins on high-speed channels that would otherwise run too slowly. Another relatively common analog feature is differential comparators on input pins designed to be connected to differential signaling channels. A few "mixed signal FPGAs" have integrated peripheral analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) with analog signal conditioning blocks allowing them to operate as a system-on-a-chip. Such devices blur the line between an FPGA, which carries digital ones and zeros on its internal programmable interconnect fabric, and field-programmable analog array (FPAA), which carries analog values on its internal programmable interconnect fabric.

V. CONCLUSION

This paper has presented the importance of hardware acceleration for today's rapidly developing embedded system devices. Speed is important factor of the system. So that hardware should be work fast for better efficiency of the system. Hardware acceleration is most important for this. For increasing the efficiency of processor we can use portdog for some part of the input signal checking and its controlling. Protdog can be used as coprocessor of the main processor. This will help in increasing the hardware acceleration. This can be implemented using FPGA platform on Xilinx Spartan 3E family.

VI. ACKNOWLEDGEMENT

This proposed concept of portdog is part of patent numbered "4139/MUM/2013" dated "31/12/2013" by "Chittaranjan Pramod Mahajan, Pune"(Director, Dolphin labs Pune), Related to "Port dog/ Peripheral dog to monitor ports/ on chip peripherals present on microcontroller".

REFERENCES

[1] Andre Nogueira, Mario Calha, "Predictability And Efficiency In Contemporary Hard RTOS For Multiprocessor Systems", 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.

[2] Thomas A. Henzinger and Joseph Sifakis, "The Embedded System Design Challenge".

[3] Sebastien Lafond, Johan Lilius, "Interrupt Costs in Embedded System with Short Latency Hardware Accelerators", 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems.

[4] Scoot Mahlke, Rajiv Ravindran et.al, "Bitwidth Congnizant Architecture Synthesis Of Custom Hardware Acceleration", IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 20, No. 11, November 2001.

[5] Khai Lik, Khoo & Mohd. Fadzil Ain, Chee Hak, Teh & Weng Li, Leow, "Scable Storage Architecture In Modular Hardware Accelerator", 2011 IEEE Symposium on Industrial Electronics and Applications (ISIEA2011), September 25-28, 2011, Langkawi, Malaysia.

[6] Arnaldo S. R. Oliveira, Luís Almeida and António de Brito Ferrari, "The ARPA-MT Embedded SMT Processor and Its RTOS Hardware Accelerator", IEEE transactions on industrial electronics, vol. 58, no. 3, March 2011.

[7] Joao Bispo, Nuno Paulino, Joao M. P. Cardoso and Joao C. Ferreira, "Transparent Trace-Based Binary Acceleration for Reconfigurable HW/SW Systems", IEEE transactions on industrial informatics, vol. 9, no. 3, August 2013.

[8] Tran Nguyen Bao Anh, Su-Lim Tan, "Survey and performance evaluation of real-time operating systems (RTOS) for small microcontrollers".

[9] Jianjun Li, LihChyun Shu, Jian-Jia Chen and Guohui Li, "Energy-Efficient Scheduling in Nonpreemptive Systems With Real-Time Constraints", IEEE transactions on systems, man, and cybernetics: systems, vol. 43, no. 2, march 2013.

[10] Santhi Baskaran1 and P. Thambidurai2, "Energy Efficient Scheduling For Real-Time Embedded Systems With Precedence And Resource Constraints", International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.2, April 2012.

[11] Ravindra Jejurikar, Rajesh Gupta, "Energy Aware Non-Preemptive Scheduling for Hard Real-Time Systems".

[12] Santhi Baskaran1 and P. Thambidurai2, "Energy Efficient Real-Time Scheduling in Distributed Systems", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 4, May 2010.