DEVELOPMENT AND PERFORMANCE ANALYSIS OF EPICS CHANNEL ACCESS SERVER ON FPGA BASED SOFT-CORE PROCESSOR

S. Sahoo#, T. Bhattacharjee, S. Pal, VECC, Kolkata, India

Abstract

A soft core processor is a flexible hardware description language (HDL) model of a specific processor (CPU) that can be customized for a given application and synthesized for an FPGA as opposed to a hard core processor which is fixed in silicon. Combined with an on-board ethernet port, the technology incorporates integrating the IOC and digital control hardware within a single FPGA thus reducing the overall hardware complexities of field devices. In this paper, the technical details of porting EPICS Channel Access Server on MicroBlaze soft-core processor are explained. The EPICS performance on the MicroBlaze processor is analyzed. For this, the CPU load and server processing time for different numbers of Process Variables (PVs) have been studied for this platform. On the basis of the analysis, critical parameters of EPICS on this embedded platform have been derived and a few modifications in the channel access protocol are proposed for MicroBlaze soft-core processor.

INTRODUCTION

Experimental Physics and Industrial Control System (EPICS) has been used in many accelerator laboratories to design the control system of accelerator under a unified architecture for better reliability, integrity and security of the overall control system of an accelerator.

The Input Output Controller (IOC) is the heart of an EPICS distributed control system and many such IOCs can be distributed over the control network. These IOCs are generally loaded in Personal Computers (PCs) running windows or linux operating system and placed near the field devices. Nowadays, the PC-based EPICS IOC is used in many laboratories, but it has many maintainability issues.

A soft core processor is a flexible CPU architecture that is configured in the FPGA as opposed to a hard core processor which is fixed in silicon. Combined with an on-board Ethernet port, the technology incorporates the IOC and digital control hardware within a single FPGA [1]. Nios II (by Altera), MicroBlaze (by Xilinx), OpenRISC 1200 (by OpenCores.org), LatticeMicro32 (by Lattice Semiconductors) and Cortex-M1 (by ARM Limited) are some of the soft-core processors that are targeted mainly for FPGA implementation. On the basis of various features like portability of Linux operating system, availability of FPU (Floating Point Unit) and MMU (Memory management Unit) and number of logic elements occupied, it has been seen that MicroBlaze processor core is the most optimized target soft-core processor for our application. The use of MicroBlaze processor and the uC-linux operating system has been very successful to date [2]. Also placing the processor and the user-defined hardware on the same device does offer many benefits and better reflects the state-of-the-art in systems-on-chip [3]. Our 62.5 MHz MicroBlaze provides a great deal of processing power, the Spartan-3A FPGA provides the capability to implement a significant amount of user-logic, and the uC-linux operating system provides a good platform for software development and debugging.

The scope of this paper includes porting EPICS on FPGA based MicroBlaze soft-core processor and analyze the EPICS record processing and channel access performance on it. To achieve this, broad steps which are necessary to be performed are described in the subsequent sections.

# ssahoo@vecc.gov.in

BUILDING MICROBLAZE PROCESSOR

We have used mainly Xilinx Platform studio for building the MicroBlaze system [4] (i.e. the processor along with the peripherals and interconnects). Figure1 is the basic flow diagram representing the steps involved in building a MicroBlaze system that has been customized as per requirements in our case.

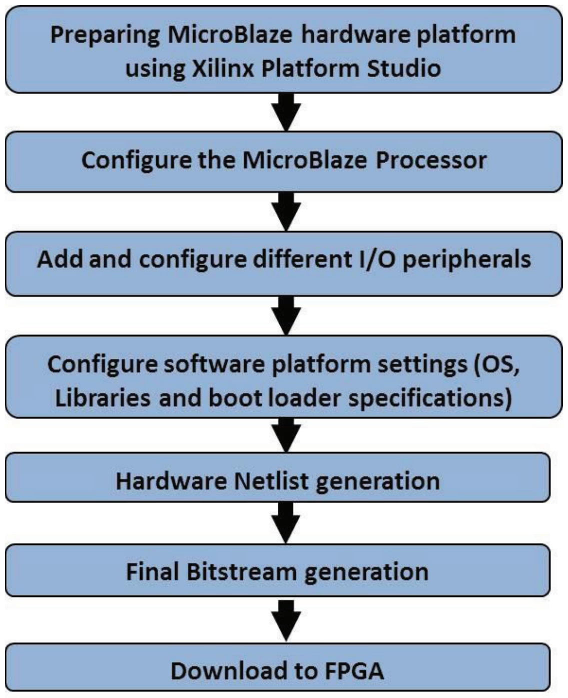

Preparing MicroBlaze hardware platform using Xilinx Platform Studio
Configure the MicroBlaze Processor
Add and configure different I/O peripherals
Configure software platform settings (OS, Libraries and boot loader specifications)
Hardware Netlist generation
Final Bitstream generation
Download to FPGA

Figure 1: Flow Diagram for Building MicroBlaze System.

ISBN 978-3-95450-124-3

PORTING LINUX ON MICROBLAZE

The uC-linux 2.6 has been ported [5] on the MicroBlaze system built for our purpose. The original uC-linux was a derivative of Linux 2.0 kernel intended for microprocessor based systems without a Memory Management Unit (MMU). Though the present version of uC-linux 2.6 also supports MMUs, however, in our system we have implemented it without an MMU. Figure 2 describes broadly the steps followed by us for the porting of linux kernel on the MicroBlaze soft-core processor.

PORTING EPICS ON MICROBLAZE

Portable Channel Access Server was cross compiled manually for MicroBlaze architecture. Successful testing was performed on Xilinx Spartan-3A DSP 1800 evaluation board. It was found that following are the steps need to be followed for porting channel access server on MicroBlaze.

Building GNU Cross Compiler Tool-chain for MicroBlaze-uC-linux Platform

Xilinx MicroBlaze GNU tools source package was built which includes binutils-2.16, gcc-4.1.2, gdb-6.5 and newlib-1.14.0.

Building the Necessary Library Packages for MicroBlaze-uC-linux Platform

The libCom, libca, libcas, libgdd and librt libraries were needed to be built in addition to prebuilt libraries in the compiler tool-chain.

Compiling the Portable Channel Access Source-codes Using MicroBlaze-uC-linux Tool-chain

Portable Channel Access source codes provided in makeBaseApp are compiled using a suitable makefile, created manually.

Building the uC-linux Kernel Image Along with Server Application

The server application created above is included in the kernel image of uC-linux 2.6.

Downloading the Kernel Image into FPGA

The resultant kernel image is downloaded to the configuration flash memory of the FPGA board.

Table 1: List of MicroBlaze Configurations

Configuration	Pipelining	I-Cache	D-Cache
Conf. 1	3 Stage	2 kB	2 kB
Conf. 2	3 Stage	8 kB	8 kB
Conf. 3	5 Stage	2 kB	2 kB
Conf. 4	5 Stage	8 kB	8 kB

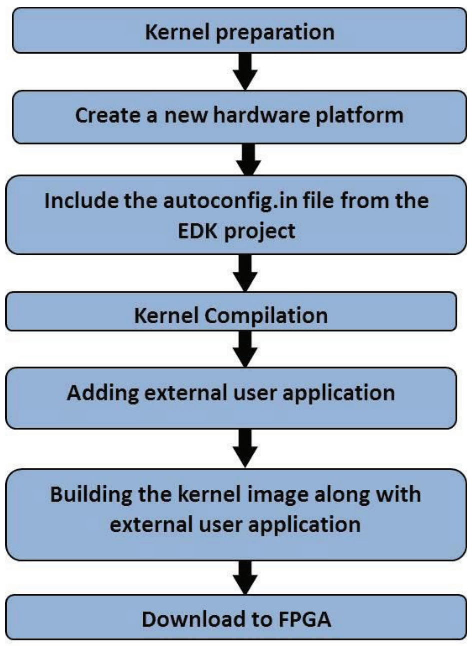

Kernel preparation
Create a new hardware platform
Include the autoconfig.in file from the EDK project
Kernel Compilation
Adding external user application
Building the kernel image along with external user application
Download to FPGA

Figure 2: Flow Diagram for porting uC-linux.

PERFORMANCE ANALYSIS OF EPICS

We have calculated mainly two parameters while doing the performance analysis of Channel Access Server on MicroBlaze processor [6].

1. **Server CPU Load**: This is percentage utilization of CPU resource at the server end while servicing to the client requests.
2. **Server Processing Time**: Time required by the server to accept the client requests, process them and publish them back to the client.

The Channel Access Protocol [7] is consisting broadly of four steps viz. Channel Connect, Put, Get and free. The individual Server CPU load and processing time for each of these steps are measured for MicroBlaze and compared to ARM9 processor which has a similar architecture of MicroBlaze but having a fixed-core that lacks the facility of changing the processor architecture and peripherals unlike soft-core processors. For each processor type, the maximum number of PVs in the database is calculated from the Server Processing Time and safe limit of number of PVs is estimated from the CPU load parameter. Since MicroBlaze processor is fully customizable soft-core target, so four different configurations are built and tested in order to optimize the results. The four configurations are listed in Table 1.

Figure 3 and 4 shows the graph of CPU Load (in %) to number of PV requests at the server machine for ARM9 and MicroBlaze processor (Configuration 4) respectively. For the other configurations of MicroBlaze processor we have got similar results.

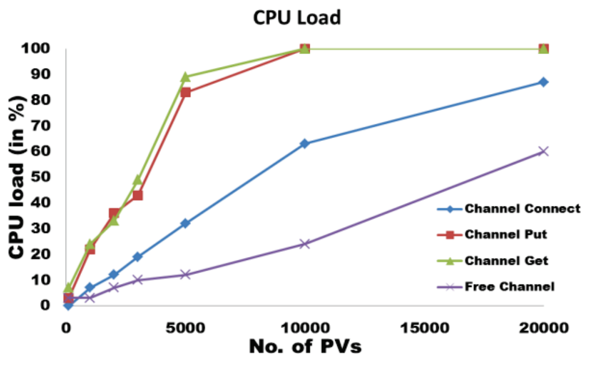


Figure 3: CPU load in ARM9.

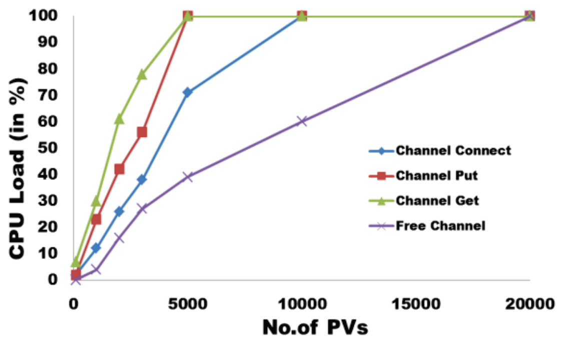


Figure 4: CPU load in MicroBlaze (Conf. 4).

Figure 5 and 6 shows the server processing time per PV for channel access get and put on different platforms. As expected with higher cache memory and greater number of stages of pipelining the performance of the MicroBlaze processor is enhanced. But the ARM9 processor is almost twice as fast as MicroBlaze (8 kB cache and 5 stage pipelining). The probable reasons for this are as follows:

1. No memory management unit in MicroBlaze processor.
2. Fork is not supported.
3. Higher processor speed (200 MHz for ARM9 processor and 62.5 MHz for MicroBlaze Processor).

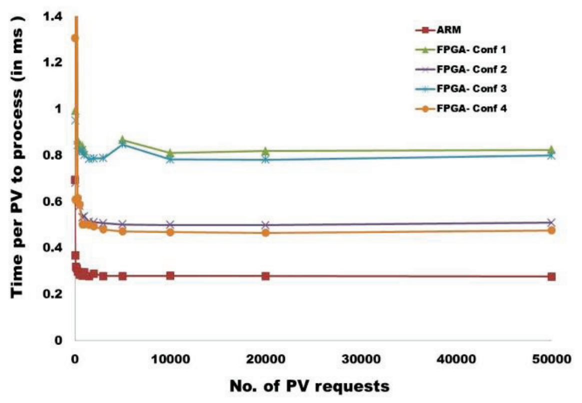


Figure 5: Server processing time per PV for CA_Put.

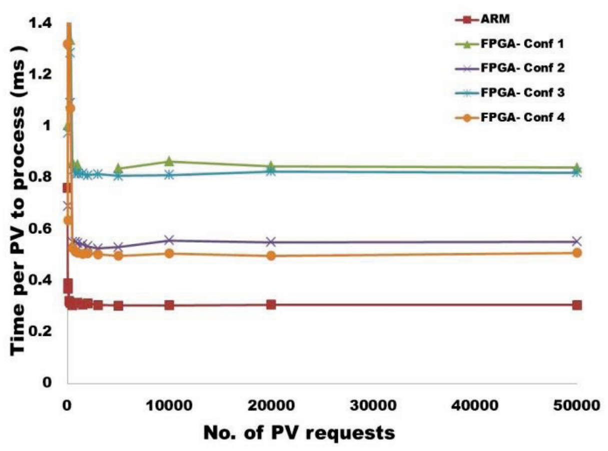


Figure 6: Server processing time per PV for CA_Get.

From the server processing time per PV graph, we have calculated the maximum number of PVs that are allowed in the server database. Results are calculated for three different scan periods of PVs viz. 0.1, 1 and 10 seconds and listed in Table 2. These values are estimated considering only Channel Access Get. Also we have estimated a safe limit of PVs from Figure 3 and 4 which determines the number of PV requests from client up to which the server can process without overloading itself.

Table 2: Results for Maximum Number of PVs at Different Scan Rate and Safe lLimit of PVs

Config.	Max. PV Limit(0.1s)	Max. PV Limit (1s)	Max. PV Limit (10s)	Safe PV Limit
Conf. 1	80	1200	12000	2500
Conf. 2	80	2000	20000	3000
Conf. 3	80	1300	13000	2500
Conf. 4	80	2200	22000	4000
ARM9	400	4000	40000	5000

SOME ABNORMAL BEHAVIOURS IN LOWER RANGE OF PVS

Figure 7, 8 and 9 shows the graph between processing time per PV vs. No. of PVs in Channel Connect, Put and Get respectively. The graphs show the detailed trend of curve in lower range of PVs (i.e. between 1 to 1000). In all the three graphs we see that a sudden rise in server processing time per PV takes place at different number of PVs for MicroBlaze processors. Only configuration 1 and 4 are shown here but similar nature is obtained for all the MicroBlaze configurations. For Channel Connect there is a sharp rise in server processing time per PV when the number of PV requests crosses 430. Similar peaks can be seen at 60 and 100 for Channel Put and Get respectively. It can be noted that for the ARM9 processor such peaks are not seen.

ISBN 978-3-95450-124-3

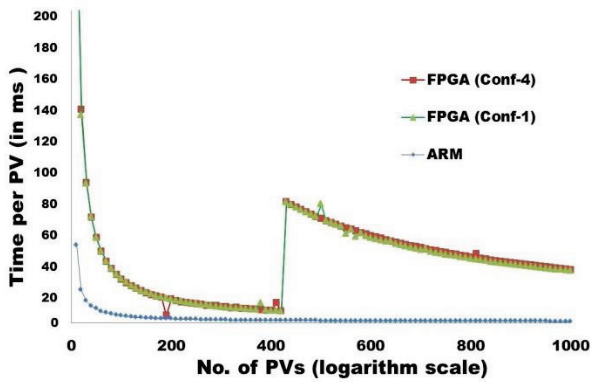


Figure 7: CA_Connect in MicroBlaze.

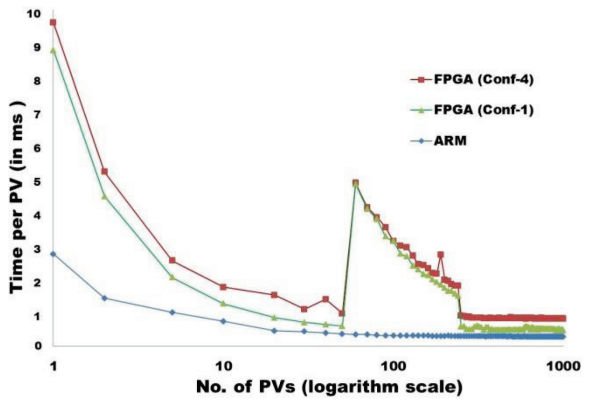


Figure 8: CA_Put in MicroBlaze.

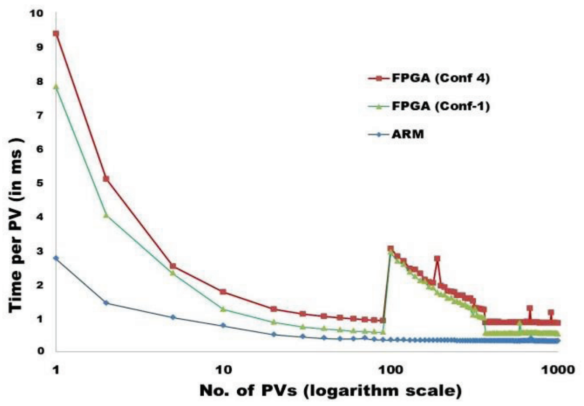
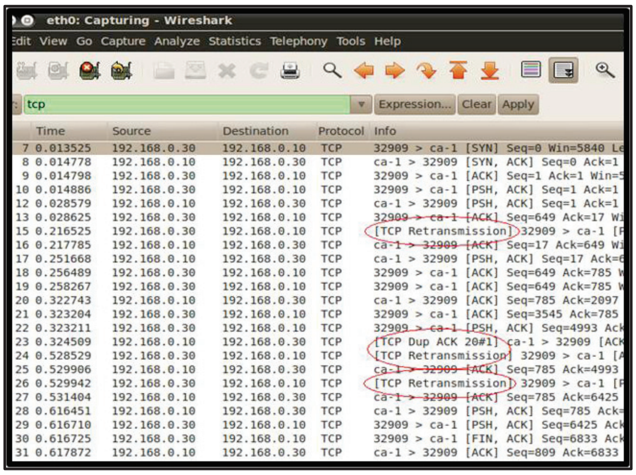


Figure 9: CA_Get in MicroBlaze.

After analyzing the reason behind these peaks, it has been found that repetitive retransmission of TCP segments of the TCP/IP stack is taking place while communication with MicroBlaze processor. This is making per PV transaction time nondeterministic in a congestion free network channel, hence the soft real time response of EPICS channel access is being violated which is normally achieved in a network with 30% load, possibly with minimum or no collision domain in the network design by intelligent network switches.


eth0: Capturing - Wireshark
Edit View Go Capture Analyze Statistics Telephony Tools Help
tcp
Expression...
Clear
Apply

```
   Time        Source         Destination    Protocol  Info
 7 0.013525    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [SYN] Seq=0 Win=5840 Le
 8 0.014778    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [SYN, ACK] Seq=0 Ack=1
 9 0.014798    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [ACK] Seq=1 Ack=1 Win=5
10 0.014886    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [PSH, ACK] Seq=1 Ack=1
12 0.028579    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [PSH, ACK] Seq=1 Ack=1
13 0.028625    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [ACK] Seq=649 Ack=17 Wi
15 0.216525    192.168.0.30   192.168.0.10   TCP       [TCP Retransmission] 32909 > ca-1 [F
16 0.217785    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [ACK] Seq=17 Ack=649 Wi
17 0.251668    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [PSH, ACK] Seq=17 Ack=6
18 0.256489    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [ACK] Seq=649 Ack=785 W
19 0.258267    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [ACK] Seq=649 Ack=785 W
20 0.322743    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [ACK] Seq=785 Ack=2097
21 0.323204    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [ACK] Seq=3545 Ack=785
22 0.323211    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [PSH, ACK] Seq=4993 Ack
23 0.324509    192.168.0.10   192.168.0.30   TCP       [TCP Dup ACK 20#1] ca-1 > 32909 [ACK
24 0.528529    192.168.0.30   192.168.0.10   TCP       [TCP Retransmission] 32909 > ca-1 [A
25 0.529906    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [ACK] Seq=785 Ack=4993
26 0.529942    192.168.0.30   192.168.0.10   TCP       [TCP Retransmission] 32909 > ca-1 [F
27 0.531404    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [ACK] Seq=785 Ack=6425
28 0.616451    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [PSH, ACK] Seq=785 Ack=
29 0.616710    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [PSH, ACK] Seq=6425 Ack
30 0.616725    192.168.0.30   192.168.0.10   TCP       32909 > ca-1 [FIN, ACK] Seq=6833 Ack
31 0.617872    192.168.0.10   192.168.0.30   TCP       ca-1 > 32909 [ACK] Seq=809 Ack=6833
```

Figure 10: Wireshark screenshot for TCP communication with MicroBlaze Server.

It has been seen that in MicroBlaze Processor due to limited memory resources and lack of memory management unit and a known bug in soft_irq process, a few data packets are often lost which triggers the retransmission in the client socket. This phenomenon is totally unreliable as it depends on the non-deterministic transmission delay of packets with adaptive retransmission algorithm. In the case of less number of PV requests there is no retransmission taking place because only one TCP data packet (1448 Bytes) is enough and as soon as multiple TCP packets are sent, retransmission is happening due to the inability of multiple data packet handling of the MicroBlaze processor. The repetitive retransmission enables the retransmission algorithm to increase the retransmission timeout. Moreover, in some cases Nagles algorithm comes into picture which tries to increase the packet size in order to compensate the time loss. This worsens the situation because the server (MicroBlaze) is not capable of handling bigger data packets which again causes retransmission. This phenomenon deteriorates the overall network performance.

PROPOSED IMPROVEMENT

Looking into the above scenario of retransmission a solution can be proposed which will be effective for any embedded system lacking the hardware resource which causes non-deterministic network communication with EPICS channel access protocol.

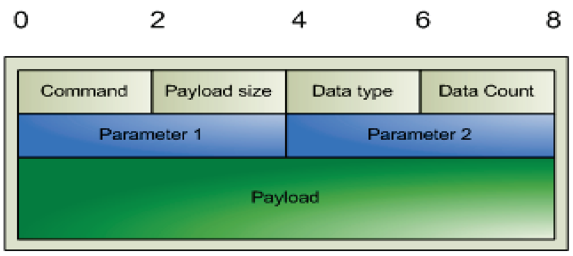


Figure 11: General Message Header in Channel Access.

ISBN 978-3-95450-124-3

In the EPICS channel access a technique, the concept of message buffering is introduced which increases the efficiency of the channel by grouping individual messages in a single message segment. In congestion free network where repetitive retransmission is a common phenomenon due to the inefficiency of handling multiple TCP packets by the server socket, this message buffering concept is a bottleneck. It can be straight way proposed for the MicroBlaze processor that the optimized channel access performance would be with the message buffer size equal to 1448 with BSD socket (derived from wireshark results). Nonetheless, considering this retransmission phenomena in other embedded systems lacking resources, we can think of an adaptive message buffering algorithm in channel access to change the payload size dynamically during the initial connection with server or on notification from the server which can be described as follows:

- Start with a fixed payload size (say $\text{Payload}_{\text{Original}}$) for channel access message buffer.
- If retransmissions are detected in the media, reduce the payload size to half of its previous value.
  i.e. $\text{Payload}_{\text{New}} = \text{Payload}_{\text{Original}} / 2$
- Detect for any retransmission in the media. If retransmission is detected reduce the payload size further by half of its previous value.
- Thus after N iterations when there is no retransmission in the media, the final payload size becomes
  $\text{Payload}_{\text{Final}} = \text{Payload}_{\text{Original}} / 2^N$

$$N = \log_2 \frac{\text{Payload}_{\text{Original}}}{\text{Payload}_{\text{Final}}}$$

$\text{Payload}_{\text{Final}}$ is the optimum payload size for message buffer in channel access for no retransmission.

This algorithm has to be embedded with the general message buffering available in channel access as the same client may connect a IOC server where the performance is better with the general message buffer and hence the additional logic in the embedded server should be defined to notify the client that it needs the proposed adaptive message buffering algorithm to establish connection.

CONCLUSION

In this project we have successfully ported EPICS channel access server on MicroBlaze soft-core processor. To understand the suitability of embedded systems in real-time environment, EPICS channel access and record processing performances have been analyzed for MicroBlaze platform and the results are compared with standard ARM9. The CPU load and server processing time for different numbers of client requests have been studied. Maximum number of PV limit in a server database is calculated for both Microblaze and ARM9 server machine. The performance in MicroBlaze soft-core processor can be considerably improved by suitably tuning the processor architecture (like size of cache memory and number of stages of pipelining). Another improvement being proposed for embedded system is the change in message buffer size of EPICS channel access which quite frequently causes retransmission and thus degrading the real-time performance of the system.

REFERENCES

[1] D. Curry, A. Hofler, H. Dong, T. Allison, C. Hovater, K. Mahoney,“Implementation of an EPICS IOC on an Embedded Soft Core Processor Using Field Programmable Gate Arrays”, Proceedings of 10th ICALEPCS, Geneva, 2005.

[2] J.G. Tong, Ian D.L. Anderson, Md. A. S. Khalid, “Soft-Core Processors for Embedded Systems”, Proceedings of 18th International Conference on Microelectronics (ICM), 2006.

[3] R. H. Klenke, “Experiences Using the Xilinx MicroBlaze Soft-core Processor and uC-linux in Computer Engineering Capstone Senior Design Projects”, Proceedings of IEEE International Conf. on Microelectronic Systems Education, 2007.

[4] R. Jesman , F. M. Vallina and J. Saniie, “MicroBlaze Tutorial for Creating a Simple Embedded System and Adding Custom Peripherals Using Xilinx EDK Software Tools”, 2006.

[5] J. Wu, I. Syed and J. Williams, “Creating a simple uC-linux ready MicroBlaze Design version 1.05a”, http://itee.uq.edu.au/~wu/downloads/uC-linux_ready_Microblaze_design.pdf

[6] J. Odagiri, A. Akiyama, N. Yamamoto and T. Katoh, “Performance Evaluation of EPICS on PowerPC” , Proceedings of ICALEPCS, Beijing, 1997.

[7] K. Žagar, “Channel Access - Protocol Specification”, http:// epics.cosylab.com/cosyjava/JCA-Common/ Documentation/ CAproto.html, 2003.

ISBN 978-3-95450-124-3