Embedding Soft processor based USB device driver on FPGA

Prof. Shashank Pujari

Abstract— Embedding a System on a Programmable Chip with in the constraint of available resources brings out ingenuity of a SoPC designer. The paper covers an efficient Soft processor based USB communication device driver implementation on FPGA using a 10% foot print so as to set aside the rest 90% of the logic resources of the programmable chip for other compute and control intensive tasks. Using a Soft processor on FPGA to interface a peripheral is a convenient choice so that the task of the device driver development can be off loaded to a software developer. Software developer takes a black box approach and uses the programmer model of the Soft processor and this is where things can go wrong. SoPC designer looks critically at the device driver requirement and brings in hardware/software co-design approach to think out of the box to meet the challenges of the constraint aware embedded design in terms of cost, size, memory, performance and time to market.

Index Terms— Device Driver, Embedded, FPGA, Soft Processor, SOPC,

1 INTRODUCTION

USB is an integral part of an embedded system requiring communication between host PC and target system. There are two types of USB controllers available in the market, one having built in Microcontroller and other without it. The microcontroller based USB controller has built in device driver, which handles all data communication tasks, is an easy choice but not very cost effective for small-embedded design targeted for $10 cost. For example Cypress CY7C68013A- 16A family USB controller has 8051 core and the unit cost is $5 which is not economical for use in a $10 cost FPGA based embedded product. To design a low cost FPGA based embedded system, the cost of the USB2.0 device and the selection of FPGA should be proper. In this paper a NXP USB 2.0 controller ISP1582 costing $3 and $10 Spartan-3 FPGA device from Xilinx are used.

For FPGA based design requiring USB data communication, it is desirable to build the USB driver along with target design in the FPGA. The USB interface logic should use minimal resources of FPGA to accommodate other main logics. Present paper discusses issues and solutions for embedding USB driver in FPGA in a cost effective manner by a memory efficient design. The hardware-software co-design approach is based on the embedded soft processor and internal memory in side FPGA. An 8-bit RISC Soft Micro Controller (SMC) core namely PicoBlaze offered by Xilinx is used. The small footprint PicoBlaze supports 1K Byte of instruction memory, which is not sufficient to hold the complete USB device driver. A technique for increasing the size of the instruction memory by using a switchable tri-pair memory bank, there by scaling the addressability of the soft processor, is the key motivation behind this paper.

The paper is organized as follows; Soft processor is introduced in section II, section III covers USB device driver flow, section IV covers Design Implementation; section V covers Hardware/software co-design approach with Tri-pair switched memory bank. Concluding remark is given in section VI followed by acknowledgement and reference.

• *Prof. Shashank Pujari is currently with Sambalpur University Institute of Information Technology (www.suiit.ac.in) , Jyotivihar, Burla, Sambalpur, Orissa-768019*
pujarishashank@gmail.com

2 SOFT PROCESSOR

The PicoBlaze SMC core is embedded within the target FPGA and requires no external resources. The SMC is optimized for efficiency and low deployment cost. It occupies just 96 FPGA slices, or only 20% of an XC3S200 FPGA. The SMC performs a respectable 44 to 100 million instructions per second (MIPS) depending on the target FPGA family and speed grade. The SMC works at 50 MIPS, high enough to handle High-speed data rate (480Mbits) of USB2.0 controller. PicoBlaze SMC is delivered as synthesizable VHDL source code and hence the core is future-proof and flexible enough to be adapted to future FPGA architectures, effectively eliminating fears of product obsolescence. Being integrated within the FPGA, the PicoBlaze SMC reduces board space, design cost, and inventory.

A suite of development tools supports the PicoBlaze SMC software development, including an assembler, a graphical Integrated Development Environment (IDE), a graphical instruction set simulator.

For porting the Picoblaze SMC to FPGA devices of other vendors such as Altera and Actel, the VHDL code has been made vendor independent by using commonly available logic primitives of the FPGA [1].

3 USB DEVICE DRIVER

The USB device driver software flow chart is shown in Fig – 1. After power on reset, the registers of the USB controller ISP 1582 are initialized for high speed (480Mbits) data communication, nos. of endpoint and DMA. Then the enumeration process begins, which establishes communication between the slave target USB device and the PC host. A set of descriptor is sent from the target USB device to the host that describes the device's USB capabilities and how they will be used. Next the application communication begins. The host sends read or write command to slave and in return the slave sends data packets requested by host or receives data packets from host respectively. All packet transactions are handled through interrupts under DMA control. All types of communication i.e., Bulk, Interrupt, Isochronous and Control are handled through proper initialization of USB device registers. The Processor program is stored in three ROMs as shown in flow chart in three respective shades i.e., Dark shade – ROM1 – All initializations, Light shade – ROM2 – Enumeration, transmit and receive packet, No shade - ROM3 – USB decision control flow and descriptors.

4 DESIGN IMPLEMENTATION

The design was implemented on a Spartan3 XC3S200 device using Xilinx Kit, connected to a NXP ISP1582 device, over the general-purpose connectors A1 and A2 available on the kit. The clock speed of the FPGA device decides the speed of the USB communication. The System clock of USB interface is 80 MHz and sufficient for High-speed 480 Mbits/sec USB 2.0 performance. The VHDL code is compiled on Xilinx ISE 7.2 Software tools. The assembly language program of Soft Processor is compiled using KCPSM3 assembler. The internal and external components of SPARTAN XC3S200 are shown in Fig - 2 i.e., Soft Microcontroller, Read/write FIFO Interface, USB interface and an external NXP ISP1582 USB controller. Fig-3 shows the detailed internal blocks of the FPGA.

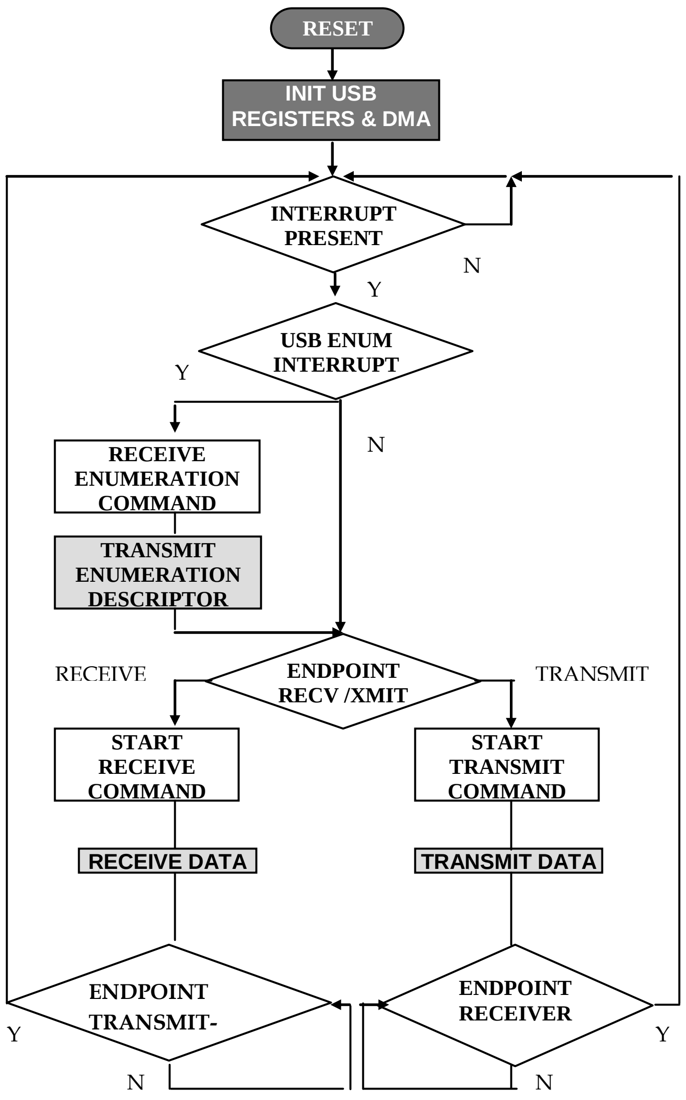

Fig- 1 Flow chart of USB device driver

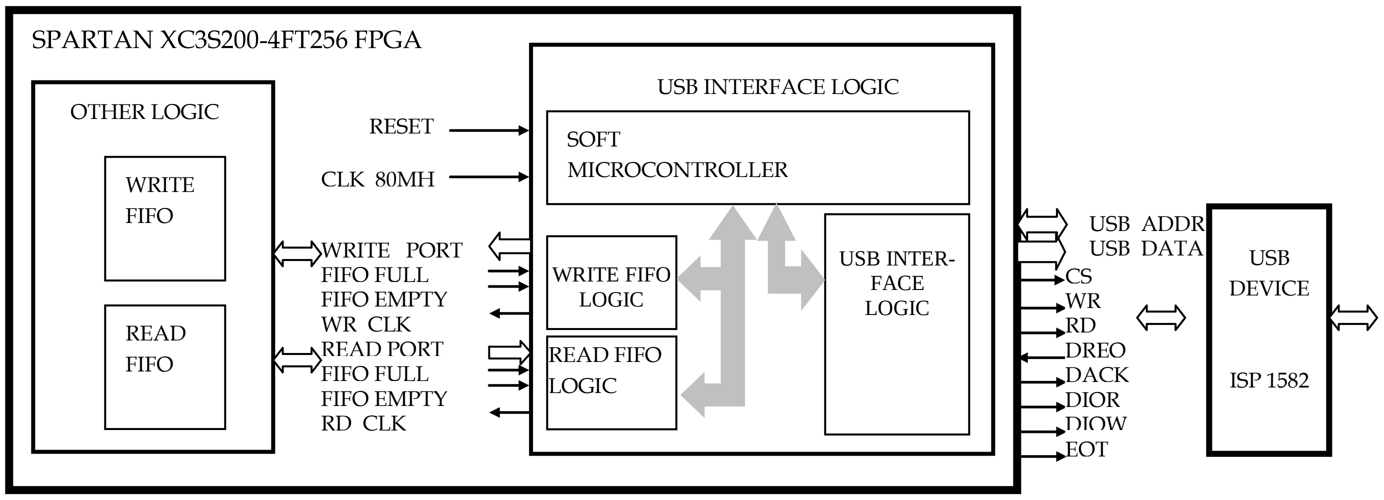

Fig –2 FPGA System Block diagram

SOFT MICROCONTROLLER

The SMC is implemented using Xilinx Picoblaze core available with Xilinx Core Generator library. It has one 8 bit input port, one 8 bit output port and one 8 bit port ID for selection of 256 input or output ports. It has 10 bit ROM address and accepts 18 bit instruction code from ROM. This ROM is internal to the FPGA implemented using Block RAM. The processor fetches instruction in one clock period and executes in next clock period. There is one interrupt input and one interrupt acknowledge output. Multiple interrupts can be gated by an OR gate.

The SMC over all controls the sub blocks. It initializes the USB device and completes the Enumeration process and then waits for command from PC host. It generates Read FIFO control signal, when it receives a USB read command from PC host. The Read FIFO control signal are used to read data stored in FIFO and then send it to USB. Similarly SMC generates Write FIFO control signal when it receives a USB write command from PC host. The Write FIFO control signal are used to write data received from USB into FIFO.

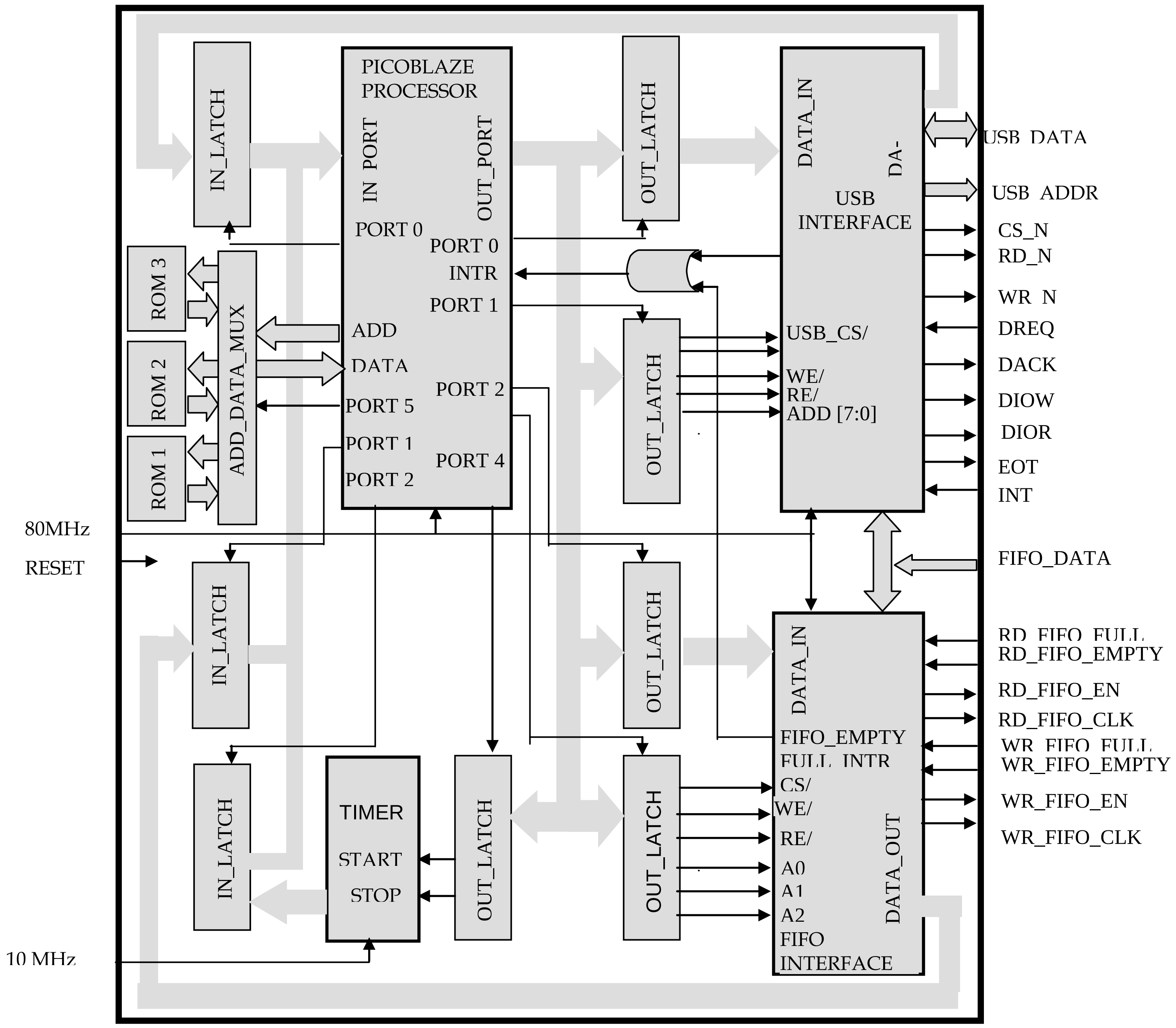

Fig – 3 FPGA functional block diagram

ROM Interface

The ROM is implemented inside the FPGA using Block RAM. The ROM code is created by a standard text editor and assembled using the KPCSM3 assembler provided by Xilinx. The instruction set is provided in the user guide of the Pico-blaze Processor. The ROM and the Soft Processor run by the same 80 MHz clock. The addressability of Pico Blaze Processor has been extended from 1K ROM to 3K ROM by address and instruction data multiplexor logic.

FIFO Interface

The FIFO collects data from other logic of the FPGA. The SMC monitors the FIFO empty and full signal during initial start up. The FIFO works on dual clock and operates on separate write and read clock. The depth and width of the FIFO are 512 Byte and 16 bit respectively for the present design but can be resized depending on applications. The SMC generates the FIFO enable and read/write clock to read/write data.

USB Interface

The USB interface connects to an external USB device NXP ISP1582. Additional DMA controller logic is implemented in the FPGA to transfer data in DMA mode as shown in Fig – 4 and explained in next paragraph. Interrupt generated by the USB device controls the SMC program execution flow shown in flow chart Fig –1.

DMA Controller

The USB data transfer rate is at high-speed (480 Mbit/s). The USB Interface has a DMA controller for fast data transfer from board to USB host and vice versa as explained in following section.

The USB controller NXP ISP1582 operates in slave mode and the DMA controller of FPGA in master mode. The DMA data will be stored in local Block RAM inside USB interface first and then transferred. For reading data from USB host the data is read by DMA and stored in local Block Ram and then forwarded to processor. For writing data to USB host, Processor writes data to local Block Ram and then forwarded to USB host through DMA. This method is used for small chunk of data transfer. For large data transfer the DMA is directly between the FIFO and the USB device NXP ISP1582. The DMA transaction per word is @80 nsec equivalent to 25 Mbytes/sec.

USB Controller

NXP ISP1582 has 16bit wide data bus and 256 addressable registers and supports USB 2.0 standard. The USB controller uses a 16-bit data bus access. For single-byte registers, the upper byte (MSB Byte) is ignored.

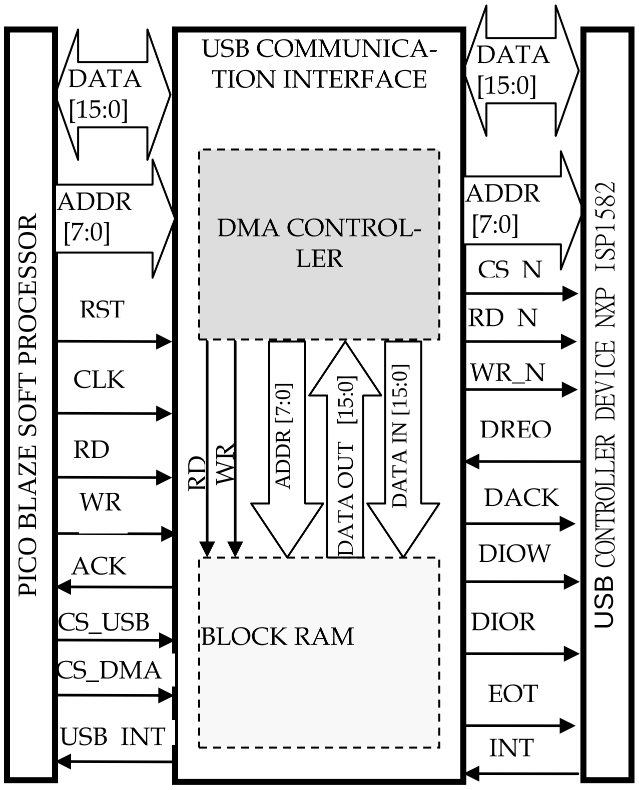

Fig – 4 USB interface with DMA controller interface

FPGA resource utilization

The design resource used in Spartan-3 device is shown in Table-1. The resource utilization shows that there is 70% of the total resources free to accommodate other logics of the system.

Table - 1 FPGA resource utilization

PROPERTY	VALUE		
TARGET DEVICE:	XC3S200		
LOGIC	USED	AVAILABLE	UTILIZATION
SLICES	512	1920	26%
SLICE FLIP FLOPS:	503	3840	13%
4 INPUT LUTS:	819	3840	21%
BRAMS:	6	12	50%

5 HARDWARE/SOFTWARE CO-DESIGN APPROACH WITH TRI-PAIR SWITCHED MEMORY BANK

The simplified flow chart of the USB device driver is shown in Fig-5. The driver has been partitioned in three main modules each occupying 1K. The USB registers are initialized in ROM1. The main program for enumeration process, transmit and receive data packet is managed in second ROM2. Third ROM3 occupies decision control flow and assorted subroutine functions including descriptor details. The memory bank switch program flowchart is shown in Fig-6. This program is there at the end of each ROM as shown in Table 2-3-4, which links the software modules resident in three separate ROMs. The associated VHDL code is given and its equivalent hardware is shown in Fig- 7.

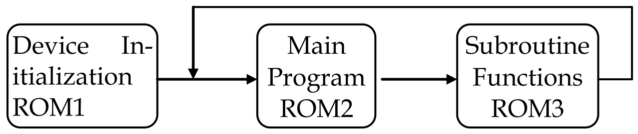

Fig-5 Device driver flow

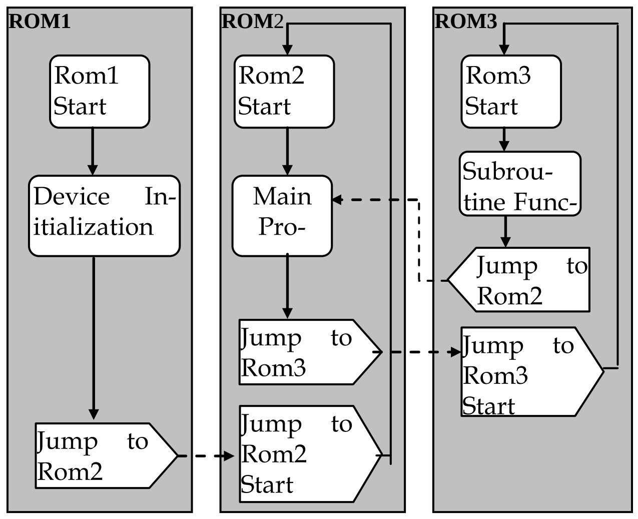

Fig-6 Memory Banks switch flow chart

The simulation diagram can be understood with reference to Table –2. It is seen that while switching from ROM1 to ROM2, the continuity of processor execution is maintained. A dummy instruction 'NOP' is inserted at the same address of both ROM1 and ROM2. Actually there is no 'NOP' instruction in Picoblaze, it is mentioned for simplicity of explanation. Actual instruction used is "AND s0, s0", which means logical "ANDING" of the register s0 with it self. The total overhead of switching between ROMs is 4 instructions.

Table 2 –MEMORY SWITCH PROGRAM IN ROM1

MEMORY_SWITCH_FROM_ROM1_ TO_ ROM2:		
	ADDRESS	3EE
0X3EE	LOAD	Port Data, 01
0X3EF	OUTPUT	Port Data, FF
0X3F0	NOP	

Table 3 – MEMORY SWITCH PROGRAM IN ROM2

MEMORY_SWITCH_FROM_ROM2_ TO_ ROM3:		
	ADDRESS	3E8
0X3E8	LOAD	Port Data, 02
0X3E9	OUTPUT	Port Data, PORT_SEL
	NOP	
MEMORY_SWITCH_FROM_ROM3_ TO_ ROM2:		
	ADDRESS	3EE
0X3EC	NOP	
0X3ED	JUMP	ROM3_TO_ROM2_PROG
MEMORY_SWITCH_FROM_ROM1_ TO_ ROM2:		
	ADDRESS	3F0
0X3F0	NOP	
0X3F1	JUMP	ROM2_START

Table 4 – MEMORY SWITCH PROGRAM IN ROM3

MEMORY_SWITCH_FROM_ROM2_ TO_ ROM3:		
	ADDRESS	3EA
0X3EA	NOP	
0X3EB	JUMP	ROM2_ROM3_PROG
MEMORY_SWITCH_FROM_ROM3_ TO_ ROM2:		
	ADDRESS	3EC
0X3EC	LOAD	Port Data, 01
0X3ED	OUTPUT	Port Data, PORT_SEL

```
; VHDL CODE  FOR ADDRESS & DATA MUX
 addr1      <= address when sel_rom = "00" else x"3FF";
 addr2      <=  address when sel_rom = "01" else x"3FF";
 addr3      <=  address when sel_rom = "10" else x"3FF";
 instruction <= instr1 when sel_rom = "00" else
                instr2 when sel_rom = "01" else
                 instr3 when sel_rom = "10";
sel_rom <= When (Port_sel = 1) then Port_data(1 downto 0)
else sel_rom;
```

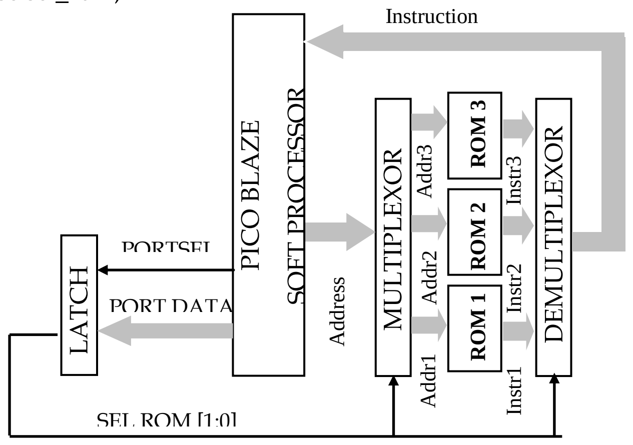

Fig- 7 Hardware equivalent of above VHDL code for Memory Bank switch

6 Conclusion

USB communication protocol stack was ported to an embedded target system on a FPGA with minimal usage of logic resources. A unique methodology of extending the address ability of the PicoBlaze Soft Microcontroller from 1K to 3K, by address and data multiplexing, was devised to accommodate the USB driver.

Scope of the paper can be extended in future for comparative study of a modified Picoblaze designed for 3K ROM addressability with the present design as well as with a Dynamically Partially Reconfigurable FPGA, where instead of switching between the three ROMs, ROMs are replaced by downloading the next ROM on a clock boundary.

References

[1] Farhad Marchant, Shashank Pujari, Manish Patil "Platform Independent 8-bit Soft core for SoPC", International Multiconference of Engineers and Computer Scientists 2009, Hong Kong, 18 March, 2009
[2] Technical specs of Xilinx FPGA Spartan 3, NXP USB ISP1582;
[3] Xilinx ISE 7.2, Chipscope & Modelsim tools manuals.
[4] USB 2.0 standard specification.
[5] Jan Axelson "USB Complete"
[6] www.nxp.com
[7] www.xilinx.com