Nuclear Science and Techniques **22** (2011) 193–199

An embedded single-board computer for BPM of SSRF

CHEN Kai[1,2] LIU Shubin[1,2] YAN Han[1,2] WU Weihao[1,2] ZHAO Lei[1,2,*] AN Qi[1,2]
LENG Yongbin[3] YI Xing[3] YAN Yingbing[3] LAI Longwei[3]

[1]*Department of Modern Physics, University of Science and Technology of China, Hefei 230026, China*

[2]*State Key Laboratory of Particle Detection & Electronics, Hefei 230026, China*

[3]*Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China*

Abstract An embedded single-board computer (SBC) system based on AT91RM9200 was designed for monitoring and controlling the digital beam position monitor system of Shanghai Synchrotron Radiation Facility (SSRF) through the Virtex-4 FPGA in the digital processing board. The SBC transfers the configuration commands from the remote EPICS to the FPGA, and calculates the beam position data. The interface between the FPGA and the SBC is the Static Memory Controller (SMC) with a peak transfer speed of up to 349 Mbps. The 100 Mb Ethernet is used for data transfer between the EPICS and SBC board, and a serial port serves as monitoring the status of the embedded system. Test results indicate that the SBC board functions well.

Key words Digital BPM, Embedded Linux system, Advanced RISC Machines, Interrupt, Multi-threads

1 Introduction

The 3.5-GeV storage ring of Shanghai Synchrotron Radiation Facility (SSRF) runs up to, 720 electron bunches to produce synchrotron lights[1–3]. Knowledge of the beam position is essential in the beam diagnostics[3–5]. A digital beam position monitor (DBPM) system is designed to measure the signal amplitudes from the four electrodes of the BPMs for calculating the beam position. The 499.654-MHz RF signal in pulse width of several hundred picoseconds is picked up by the DBPM, and the position resolution is required to be better than 10 μm for the turn-by-turn (TBT) data in *X* and *Y* directions.

The Advanced RISC Machines (ARM) has been widely used in control and data acquisition systems due to its flexibility and simplicity[6–9]. A single-board computer (SBC) board based on ARM CPU chip is designed for the communication between the DBPM system and EPICS(experimental physics and industrial control system).

As shown in Fig.1, the DBPM consists of the analog front end and the data processing board[3]. The 499.654-MHz RF signal from the four electrodes of DBPM are manipulated and sampled by the high speed high resolution A/D converters in the analog front end, and the intermediate frequency data are sent to the data processing board. All digital signal processing algorithms are integrated in a single Virtex-4 FPGA[10], including the digital down-converters, direct digital synthesizer, cascade integrator-comb decimation filter, finite impulse response filter and coordinate rotation digital computer. With the amplitude information of the 4 channel signals, the position of the beam is calculated, and transferred to the EPICS system for analysis and display.

The SBC acts as the interface between the FPGA and EPICS. It reads the calculates data in the FPGA and transfers them to the EPICS via a 100 Mb Ethernet[11–13]. It is used for receiving commands from the EPICS, controlling the DBPM system, and monitoring the system status as well. The data are of seven types, of which the sizes and rates are given in

Supported by the Knowledge Innovation Program of Chinese Academy of Science (KJCX2-YW-N27), the National Natural Science Foundation of China (10875119), and the 100-Talents Program of CAS.

* Corresponding author. *E-mail address*: zlei@ustc.edu.cn

Received date: 2011-03-25

Table 1. The ADC data are of digitized intermediate frequency of up to 117.28 MHz from the board of analog front end. The TBT data of up to 694 kHz are calculated from the ADC data using corresponding digital signal processing algorithms, so as to analyze the beam position and intensity[5]. The PM is a recoded TBT data before its trigger signal. The SA and FA are used for close orbit drift correction [5]. The TBT, PM, SA and FA data contain the 4- channel amplitudes, the position data in *X* and *Y* directions and sum of the amplitudes. The SADC and STBT are triggered synchronously by the 2-Hz injector[3,5]. The FA data are read out via optical fiber connecting the Rocket IO interface of the FPGA. Except for the FA data, all the data types are transferred to the EPICS system via the SBC, in a speed of over 33 Mbps.

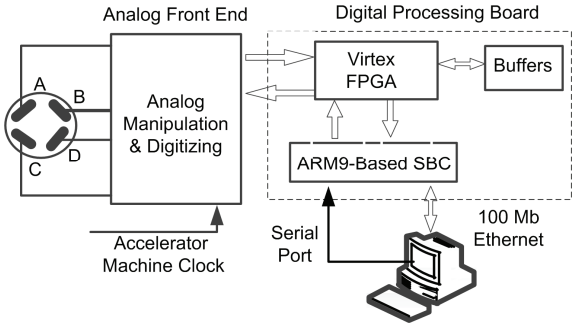

Fig.1 Function diagram of the DBPM system.

Table1 Event rate and data size of the position data in the FPGA.

Data type	Trigger rate	Data size
Small amount of ADC (SADC)	2 Hz	8 kB
Small amount of TBT (STBT)	2 Hz	32 kB
Big amount of ADC (BADC)	Manual	5920 kB
Big amount of TBT ADC (BTBT)	Manual	16 MB
Poster modern (PM)	Manual	112 kB
Slow acquisition (SA)	10 Hz	16 B
Fast acquisition (FA)	10 kHz	16 B

* The SBC is transfer port for all the data types except FA, which is transferred via Rocket IO

The ARM CPU chip AT91RM9200, a complete system-on-chip based on ARM920T Thumb processor [14], is used as the kernel of our prototype SBC, with an embedded Linux system[8]. Considering the transfer speed requirement, the SMC of low-latency external bus interface (EBI) is used for the data and commands transfer. The test results show that the SMC speed is up to 349 Mbps, far exceeding the requirement.

The Virtex-4 FPGA is a key device in this system[15]. The data is read out by requesting an interrupt to the SBC. Because there is only one transfer port between the FPGA and SBC, an interrupt arbiter is implemented within the FPGA, controlling the transfer priority of different data types.

2 Hardware design of the SBC

The hardware architecture of the SBC board is shown in Fig.2. The serial port for debugging and the Ethernet interface between the SBC and the EPICS are located in the digital processing board (DPB). The SBC board is inserted into the DPB by the dual inline memory modules (DIMM) interface. A flash memory is used to store the boot code of the embedded Linux system[8]. Two SDRAM (synchronous dynamic random access memory) chips work as memory of the embedded system. 'Arm_d[15:0]' is the data line with a width of 16 while data types are selected by the address lines 'Arm_a'. Because the read/write signal and chip select signal from the SBC to the FPGA is up to 180 MHz, all line routes are carefully designed into equal length, to provide enough set-up time and hold time for the transfer.

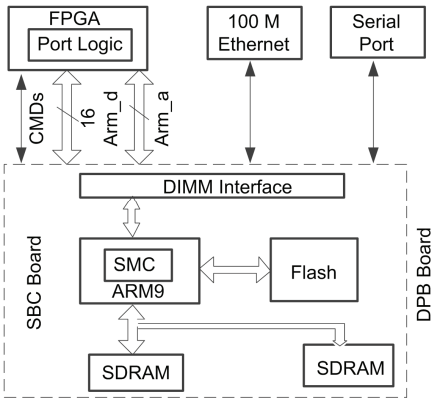

Fig.2 Hardware architecture of the SBC board.

The data transferred through SBC are categorized into the real-time sampled data (SADC, STBT, PM and SA), and non-real-time captured data (BADC and BTBT), which are read out when the capture command is received. The captured data are of bigger data volume size and smaller trigger rate. Transfer speed of the SADC, STBT, PM and SA data depends on their maximum requirement, with the 10/s SA being the highest trigger rate, and the 112 kB PM being the maximum data volume. The PM data transfer must be finished within the interval of two

adjacent SA transfers at a speed requirement of over 1120 kBps. According to the design requirement, the captured data in a maximum volume of 16 MB are sent out within 4 s at a speed of about 33 Mbps.

The data transmission path is from FPGA to SBC, and from SBC to EPICS. The 100 Mb Ethernet physical layer device between the SBC and EPICS is adequate for the captured data transfer[11,12]. Also, the transfer speed from the FPGA to the SBC is over 33 Mbps.

The AT91RM9200, an external bus interface, is chosen for the data transfer, in the burst and single word modes[14]. The former is used for all the data except SA to transfer a data block rapidly with 8 words each time, while the latter is used to transfer the SA data and other commands in small volume size.

3 FPGA logic design for the interface

Based on very-high-speed integrated circuit hardware description language, a logic module is implemented within the FPGA as a transfer port between the FPGA and SBC[16]. The data are read out under control of the 'nRD' signal's, while the 'nWE' controls the write process. The data transferred are synchronized to these signals (Fig.3). The interrupt transfer mode is applied. The 'Irq' as interrupt request will be described in more details below. In the SMC, two transfer modes are used for different types of data and commands. The two FIFOs in Fig.3 work in the burst transfer mode, one for captured data, and the other for real-time data (STBT, SADC and PM). The captured data buffering in a different FIFOs can satisfy the real-time requirement. The SA data and commands are stored in a register bank, and the registers are read out or updated by the address bus, 'Arm_a'.

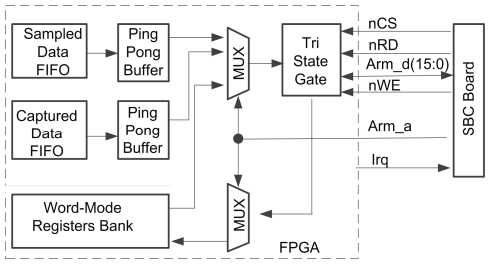

Fig.3 Logic diagram of the port in the FPGA.

The data transfer between the SBC and FPGA is synchronized to the 180-MHz system clock of the CPU in the SBC. As shown in Fig.4, the 'nRD' is a low level effective signal, and the data 'Arm_d[15:0]' from FPGA are read by the SBC at rising edges of the 'nRD'. In the burst transfer mode, 8 words (indicated as 'D0' to 'D7' in Fig.4) are read out continuously in 48 clock periods, and the 'nRD' interval of only 1 clock period is much shorter than the single word transfer mode, hence a much faster speed of the burst mode. The Chipscope software is used to test the transfer speed; the results indicate that the transfer rate is about 349 Mbps.

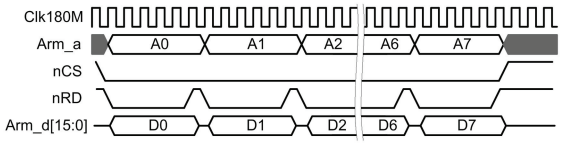

Fig.4 Timing diagram of the read operation in the burst transfer mode.

A ping-pong buffer in the FPGA is used to synchronize the position data with the falling edge of the 'nRD' signal, guaranteeing enough timing margin (Fig.5). Since the block size of the burst transfer mode is 8 words, two register banks with 8 words are implemented. When one bank is read out by the SBC, the other bank is refreshed by the FIFO data under a slower clock of 50 MHz, enhancing the stability of the logic. Two buffers of ping-pong in the FPGA are implemented for the real-time and captured data in Fig.3. For the single word transfer mode, the 'nRD' is directly synchronized to the 50 MHz clock in the FPGA, and the corresponding register is read out or refreshed by the synchronized signal.

4 Implementation of the interrupt transfer mode between the FPGA and SBC

The interrupt transfer mode, which is more effective than the round robin transfer mode, is employed between the FPGA and SBC. All types of data share one common transfer port to the SBC. An arbiter is implemented in the FPGA to determine which kind of data occupies the transfer port (Fig.6). When two or more types of data are ready for transfer, the type of the highest priority will request a control of the port that is free according to the 'Port_status' signal, while the other types of data wait in a queue. Since only one

interrupt line from the FPGA to the SBC is used in the design, the 'Int_type' register is implemented in the FPGA, which stands for the data type occupying the port. When a new type of data occupies the port, an interrupt request signal 'Irq' is asserted and sent to the SBC, and the 'Int_type' register is then updated.

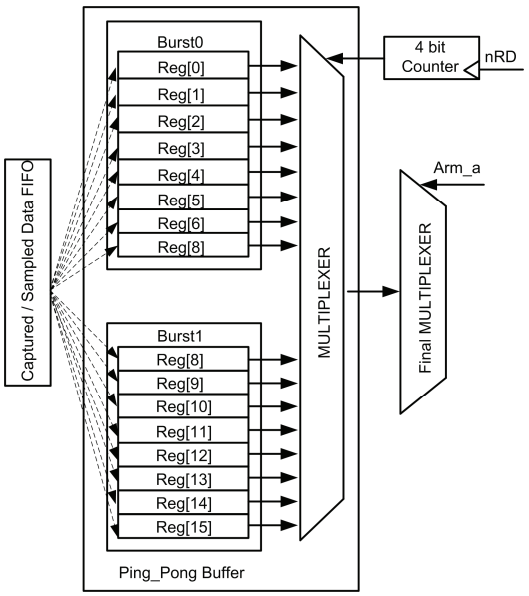

Fig.5 Block diagram of the ping-pong buffer for data transfer.

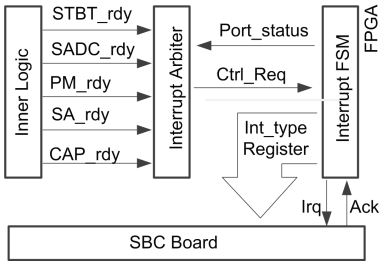

Fig.6 Logic module for the interrupt transfer.

As shown in Fig.7, an interrupt FSM (Finite State Machine) in the FPGA deals with the control request signal of arbiter, 'Ctrl_Req', and makes an SBC interrupt request. Sampled and captured data in the FSM are treated separately. As mentioned before, in order to read the sampled data as real-time data, each read process is finished within 0.1 s period of the SA data. But the captured data cannot be read out within 0.1 s, thus concedes occupying the port after 10 ms, so as to assure real-time ability of the sampled data. The 10 ms is long enough for the SBC to read correct data type storage in the 'Int_type' register.

When an interrupt request is sent to the SBC, the interrupt handler in the SBC awakes the sampling thread. Because the awaking moment may conflict with the sampling thread's blocked in the SBC, the interrupt request may not be acquired[15]. So the FSM in the FPGA checks the 'Ack' state in Fig.6 every 10 ms until the SBC acquires the request. The 10 ms check interval is shorter than the real-time data trigger (0.1 s). The sampled data occupy the port until all data are transferred, and then the captured data are read out.

All interrupt handling processes are integrated in the AT91RM9200 chip, and the performance is enhanced by the advanced interrupt controller (AIC) of the AT91RM9200. The interrupt signal, 'Irq' in Fig.6, is connected to the 4th Parallel Input/Output controller, 'PIOD'. When the 'Irq' signal changes its state, the PIOD requests an interrupt to the AIC[14], and then the interrupt handling function is executed.

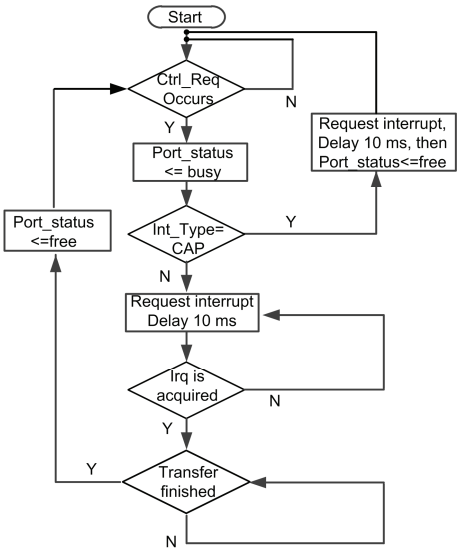

Fig.7 Finite state machine in the FPGA logic.

5 Driver design for the FPGA in the embedded system

The FPGA is a key device in the SBC embedded system and a communication driver is designed based on the Linux system. A remote personal computer (PC), instead of the EPICS system, is used in our prototype, and the application program on the embedded system is designed to test the driver in multithreaded programming[17,18], the communication, and the data transfer(Fig.8).

The driver of the FPGA device includes the 'fpga_init' function to initialize the character device, the 'fpga_open' and 'fpga_realse' function to open and release the device, respectively, the interrupt handling function 'handler', and the I/O control function 'ioctl'.

The 'fpga_init' function sets the configuration register of the 4th SMC[6], and calls the 'misc_register' function to register the FPGA as a character device in the embedded system.

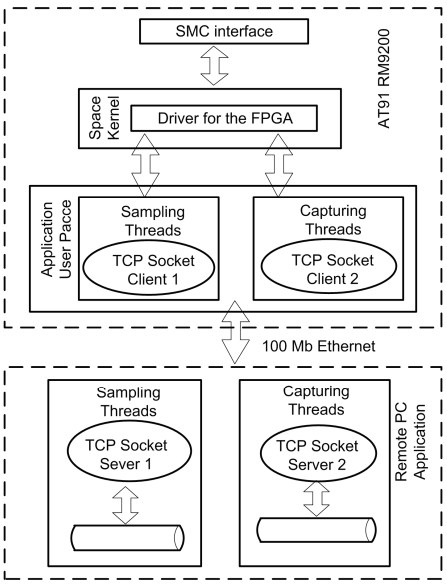

Fig.8 Software architecture of the SBC.

The 'fpga_open' function is executed to open the device. The FPGA is connected to the 4th SMC with a physics address. While only memory address in the driver is read and write, all physics addresses will be mapped to memory addresses. The 'ioremap' function is called to finish the mapping process. Meanwhile, some AIC and PIOD registers about the interrupt handlers are configured. Also, the interrupt handling function is registered and initialized in the 'fpga_open' function. The interrupt handling for the FPGA possesses an interrupt priority of 4.

As the inverse procedure of the 'fpga_open', the 'fpga_release' function is executed when the device is closed. It disables the interrupt from the FPGA, releases the interrupt number, and cancels the mapping of the addresses when the high level application program is ceased.

All the communication with the FPGA in the 'ioctl' function is implemented by the commands which include: reading all kinds of data from the FPGA, configuring the registers of the DBPM, and refreshing the configuration data in the Flash memory for the update of the FPGA logic.

Above two transfer modes for the SMC interface are used in the design, and the data readout for the burst transfer mode is implemented by using the following code piece:

```
copy_to_user((void *)arg, vaddr, 1024);
```

The 'vaddr' is the remapped address of the data array to be read out from the FPGA. The number of 1024 here refers to the block size, and the 'copy_to_user' function copies the data from the kernel space to the user space based on the burst transfer mode.

The code piece for the single word transfer mode is:

```
unsigned long i, data16b;
unsigned char data1024[1024];
for (i=0; i<512; i++)
{data16b = *(volatile unsigned short*)(vaddr); data
1024[2*i] = data16b >>8; data1024 [2*i+1] = data
16b;}
copy_to_user((void *)arg, data1024,1024));
```

It finishes the data reading word by word in the kernel space, and transfers the data to the user space by 'copy_to_user' function. Single word transfer mode is used to transfer the low-speed, small amount data of the SA data, commands, status, and logic data.

A client program in the SBC and relevant server software in the remote PC are designed for system test, and socket programming based on Linux system is used for communication between them[11–13]. Both the server software and client program are developed using C language in Linux system.

In Fig.9, the sampling thread reads the sampled data from FPGA, and the capturing thread reads captured data, and both will call the commands provided by the 'ioctl' function in the driver. On executing the command, the sampling thread blocked by the 'interruptible_sleep_on' function sleeps in its waiting queue. Also the capturing thread is blocked after sending a capture command to the FPGA. At the interrupt request from the FPGA, the interrupt handling function reads the 'Int_Type' register in the FPGA, and awakes the corresponding sleeping thread according to the register value. In order to assure the

sampled data to be real-time, the sampling thread is prior to the capturing thread. The awakened sampling thread reads all the sampled data immediately, and the awakened capturing thread reads the captured data only when the former is blocked.

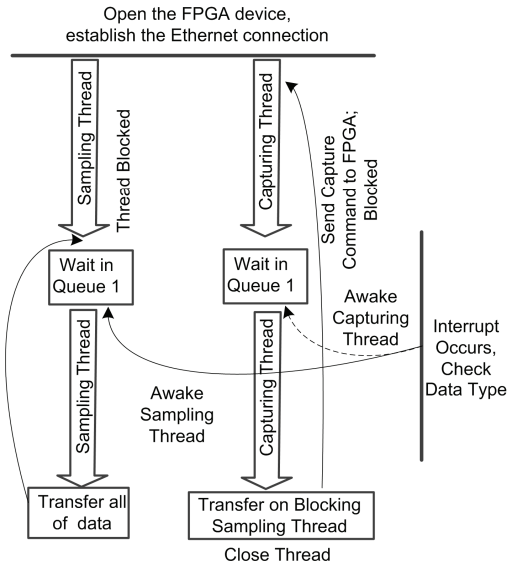

Fig.9 Sampling and capturing threads of the program in user space.

All the data are sent to the remote PC by the 100 Mb Ethernet. In its 'server' software, two data ports are used to receive the sampled and captured data from the SBC. Therefore, multithreaded programming is employed for receiving the sampled and captured data, and saving to the hard disk[17,18].

6 Test results

To confirm the transfer of all data types and the control of the DBPM, the system is tested in the laboratory (Fig.10). The 499.654 MHz RF signal is connected from the SMA 100 A signal generator (Rohde & Schwarz Corp). Position data are readout from the FPGA, transferred by the SBC, and recorded by the remote PC. The test results indicate that the total transfer speed is up to 61 Mbps. With multithreaded programming, all the sampled data can be read out in real-time, and a maximum volume of captured data of BTBT type in 16-MB size can be readout in 3 s. The performance satisfies the design requirement.

Fig.11 shows the typical position distribution in X direction tested in the laboratory, and the RMS position deviation is about 0.35 μm with the input amplitude of –5 dBm. Also, the DBPM system was tested on the SSRF storage ring. At the 200-mA beam current in multi-bunch operation mode, the position resolution are about 3.93 and 1.51 μm in X and Y direction, respectively. The results are better than the requirement.

Fig.10 The DBPM system in laboratory test.

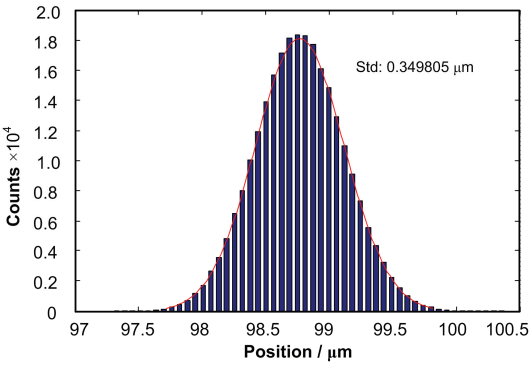

Fig.11 Distribution of the X position data in laboratory test.

7 Conclusions

The SBC board in the DBPM system of SSRF has been designed by employing an embedded Linux system. Results show that all the six types of data can be read simultaneously and transferred to the EPICS system. Also, commands and statuses are transferred by the SBC. The SBC board functions well in the DBPM system, and a good enough position resolution is achieved for the beam diagnostics in SSRF. The transfer speed of the SBC board is fast enough for the data readout and communication with the remote system.

References

1 Zhao Z, Xu H. SSRF: A 3.5 Gev synchrotron light source for China, Proc. of EPAC'04, Lucerne, Switzerland, 2004, 2368–2370.

2 Chen S, Xu H, Zhao Z. Shanghai Synchrotron Radiation Facility, Proc. of PAC'99, New York, USA, 1999, 209–211.

3 Zhou H, Liu S, Zhao L. Design of the fully digital beam position monitor for beam position measurement in SSRF, ICEMI'2009, Beijing, China, Aug 2009, **1:**1045–1051.

4 Dai Z, Huang N. Study of orbit stability in the SSRF storing ring, Processing of APAC'01, Beijing, China, Sept. 2001, 293–295.

5 Yan Y, Leng Y, Chen Y. Data acquisition and analysis in SSRF BPM system, Proc. of EPAC'08, Genoa, Italy, 2008, 1077–1079.

6 Tian J, Gao M. Image data acquisition and transmission system based on ARM, MMIT'08, Three Gorges, China, 2008, 361–364.

7 Zhang X, Song L. "Implementation of video data transmission between ARM and DSP through embedded Linux", ICESS'2008, Chengdu, China, Jul 2008, 292–295.

8 Bi C, Liu Y. Research of key technologies for embedded Linux based on ARM, ICCASM'2010, Taiyuan, China, 2010, **8:** 373–378.

9 Yu S, Chen W, LI L. Development of ARM-based embedded system for robot applications, RAM'2006, Bangkok, Thailand, 2006, 1–6.

10 Xilinx Corporation. Virtex-4 User Guide, 2007.

11 Gay W. Linux Socket Programming by Example, Que, 2000.

12 Walton S. Linux Socket Programming, Sams, 2001.

13 Benvenuti C. Understanding Linux Network Internals, O'Reilly Media, 2005.

14 Atmel Corporation. ARM920T-based Microcontroller AT91RM9200 Handbook, 2009.

15 Rubini A, Corbet J. Linux Device Driver (2nd Edition), O'Reilly Media, Inc. 2001.

16 Armstrong J, Gray F. VHDL Design Representation and Synthesis (2nd Edition), Prentice Hall. 2000.

17 Butenhof B. Programming with POSIX(R) Threads, Addison-Wesley Professional, 1997.

18 Lewis B, Berg D. Threads Primer: A Guide to Multi-threaded Programming, Prentice Hall PTR, 1995.