

CHALMERS



A remote controlled embedded system implemented in FPGA

*Master of Science Thesis in the Programme of Integrated Electronic System
Design*

ZIYI JIN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, 2009

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

A remote controlled embedded system implemented in FPGA

Z.JIN

© Z.JIN, 2009

Examiner: A.LINDE

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden 2009

Acknowledgments

I would like to thank my supervisor Arne Linde, who gives me a lot of help in the design work and many good advices in thesis writing. And my classmates Zhou Xuan and Gao Xingyu, I appreciate their sharing of information and experience when I met the problems. Further I would like to thank my parents and Qian for their encouragement and support.

Abstract

Since the Embedded system and SOC technology is increasingly developing and widely applied in industry, the education in this field is considered as an important part for the students in Electronic Engineering. Therefore, a basic but complete embedded system for demonstration is required to implement. In practice, the Altera DE2 board is adopted to construct an embedded system which consists of CPU (soft core "Nios"), ALU, memory chip and USB port. At the same time, a remote terminal in PC is created to control the system depending on the USB channel connecting two sides. In the project, the procedure for building a platform based embedded system is presented as well as the USB application in Windows environment.

Abbreviations

CPU — Central Process Unit
DDK —Driver Development Kit
DSP —Digital Signal Processor
FPGA —Field-Programmable Gate Array
GUI — Graphical User Interface
GUID —Globally Unique Identifier
HAL — Hardware Abstraction Layer
HDL — Hardware Description Language
HID — Human Interface Device
IP — Intellectual Property
ISR — Interrupt Service Routine
MFC — Microsoft Foundation Class Library
OS — Operation System
PLD —Programmable Logic Device
RF —Radio Frequency
RTOS —Real-Time Operating System
SDRAM —Synchronous Dynamic Random Access Memory
SIE —Serial Interface Engine
SOC — System On Chip
SRAM —Static Random Access Memory
USART —Universal Asynchronous Receiver/Transmitter
USB — Universal I Serial Bus

CONTENTS

1. Introduction	1
1.1 Background.....	1
1.2 Purpose.....	1
1.3 System overview.....	2
1.4 Limitation and delimitation.....	2
2. Theory	3
2.1 Embedded system design.....	3
2.1.1 Specification and Modeling.....	3
2.1.2 Hardware/Software codesign.....	4
2.1.3 Validation.....	6
2.2 SOC design.....	8
2.3 USB.....	9
2.3.1 Introduction.....	9
2.3.2 USB communication protocol.....	10
2.3.3 Enumeration and USB driver.....	13
2.4 Embedded system implementation platform.....	15
2.4.1 Hardware platform.....	15
2.4.2 Software environment.....	16
3. Method	17
4. Design and Implementation	18
4.1 System description.....	18
4.2 Embedded system.....	20
4.2.1 ALU.....	20
4.2.2 Memory and“Nios” CPU.....	21
4.3 Remote terminal.....	23
4.3.1 MFC.....	23
4.3.2 GUI.....	23
4.4 USB transmission.....	25
4.4.1 USB socket in remote terminal.....	25
4.4.2 USB firmware in embedded system.....	27
5. Discussion	30
5.1 Conclusion.....	30
5.2 Future work.....	31
6. References	32

1. Introduction

This chapter is to give a brief introduction for the motivation of the project and how it goes on.

1.1 Background

Embedded system is a compute system designing for a specific application. Both the hardware and software should be adjustable to insure the functionality, reliability, cost, volume and power dissipation could meet the requirements when serving in different field. The hardware mainly includes embedded CPU, memory chip, peripheral equipments etc. And software covers embedded operation system, developing/debugging environment and application software.

When the semiconductor process technology is developing into deep submicron range, the designers of embedded system have realized that it is possible to integrate the whole system or some key components together into a single chip. Therefore, System-On-Chip is coming forth as an essential technique to bring the embedded system design in an entire new scope and also a crucial application. It has many advantages due to its high integration: lower power dissipation for a single chip system; less pins and interface for peripherals which speed up the signal transmission and response time; less crosstalk by reducing wires and disjunct components. Accordingly the large scale integration also causes the high complexity in design, test and simulation. Besides, multi-core design is coming out as a new research hotspot with aim to explore the potential for parallel computation. A new program, named flexsoc, is launched which is focused on developing a new architecture for processor of embedded system.

1.2 Purpose

In under graduation education, the exiting embedded system developing board is too complicated for testing and demonstration. For the students to obtain an easy start of embedded system developing a new assemble evaluation system including the data input/output, basic computation and storage is required.

1.3 System overview

The whole system should be communicative with exterior and efficient to handle the input/output data. It consists of

- Input / Output terminal
- Communication channel
- Data process device

The Input/Output terminal has a user-friendly interface where the input 8-bit operands could be easily typed in as well as the required algorithm can be selected. After confirming the instruction, the result could be obtained and displayed immediately. Communication channel is responsible to transmit the data from Input/Output terminal to Data process device and vice versa. When the input data is arrived, Data process device will motivate the corresponding function to compute and the result is stored on chip or sent out when available.

1.4 Limitation and delimitation

The research will concentrate on wire communication that USB technique is applied. The wireless solution could be a further study case but out of scope in this project. Moreover, the data width is defined as 8 bits as a proper length for the beginning research work. At the same time, the implementation tool in practice is the Altera DE2 Development and Evaluation board.

For the operation system in PC terminal, the GUI is based on Microsoft XP, which has a good support for USB protocol and the programming tool: Visual C++. Some necessary functions and library files are directly used in project by installing third-party software: Windows DDK while the task for creating them is not the focus at the present period.

2. Theory

In this chapter, the relevant theory in embedded system and SOC design, USB protocols are described as well as the introduction on implementation environment.

2.1 Embedded system design

Embedded system has its own characteristics when comparing with other compute device [1]

1. With some high requirements in safety or to do some specific task, embedded systems are designed to meet the real-time performance constraints.
2. Embedded systems usually consist of small, computerized parts within a larger device that serves a more general purpose.
3. Embedded systems store the program instructions known as "firmware" in read-only memory or Flash memory chips.

Therefore in the design of embedded system there are several important steps should be emphasized and developed in new methodology.

2.1.1 Specification and Modeling

From the figure below we can observe that a design work is always launched by requirement of a practical application. But a common description for an application is vague and general, hence a specification for an engineering mission in embedded system should be abstracted which contains the following key points [2]:

- Hierarchy: Behavioral hierarchies to describe the system behavior while Structural hierarchies to present the physical components.
- Timing-behavior: timing requirements should be strictly followed for a real-time system.
- State-oriented/Exception-oriented behavior: The mechanism for state-machine portraying the system's whole working process.
- Environment supporting: Hardware platform which confines the design layout/ Software source such as IP cores, if applied, could save much time and guarantee the quality.

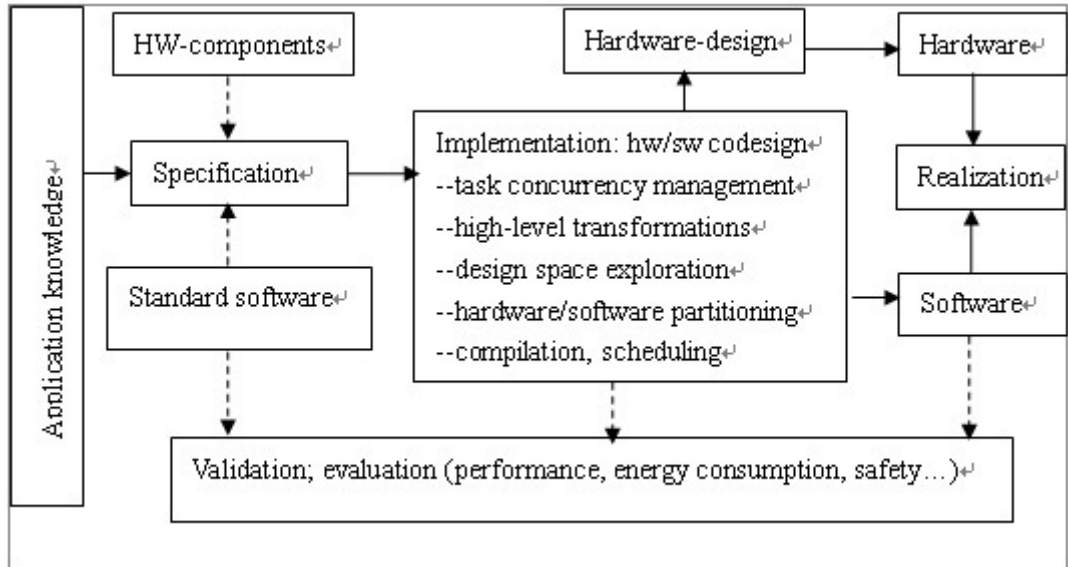


Figure 2.1 Simplified design flow

Obviously, all facts above have great impact that the designers should take into account and also some other limitations we can not ignore, for instance, the requirements on portability and flexibility, the synchronization and communication issues, etc.

After clarifying all specification of the design, it is the time to draw the outline of the system in a scientific method—modeling. On the top level, models of computation can help to define:

- Components, including procedures, processes, functions, finite state machine
- Communication protocols which delimit the interactions between different components

Then various practical means are applied to refine the modeling which are the necessary for complicated and large system. In our case, as a simple system in a matured developing environment, modeling is combined with hardware design to commence.

2.1.2 Hardware/Software codesign

One significant characteristic of embedded system is that both software and hardware design should be carried out in parallel, which is called hardware/software codesign as well. The aim is to decrease the conflict between hardware and software design while by the combination constructing, codesign would help pre-plan to fit into consequent implementation step, which can save the time for

unnecessary modification or redesign. Meanwhile, to meet the stringent time-to-market requirements and reduce the complexity of large scale system design, the design reusing has led to platform-based design. The term we can understand as Sangiovanni-Vincntelli [3] said:

“A platform is a family of architectures satisfying a set of constraints imposed to allow the reuse of hardware and software components. A quick, reliable, derivative design requires using a platform application programming interface (API) to extend the platform toward application software. In general, a platform is an abstraction layer that covers many possible refinements to a lower level. ”

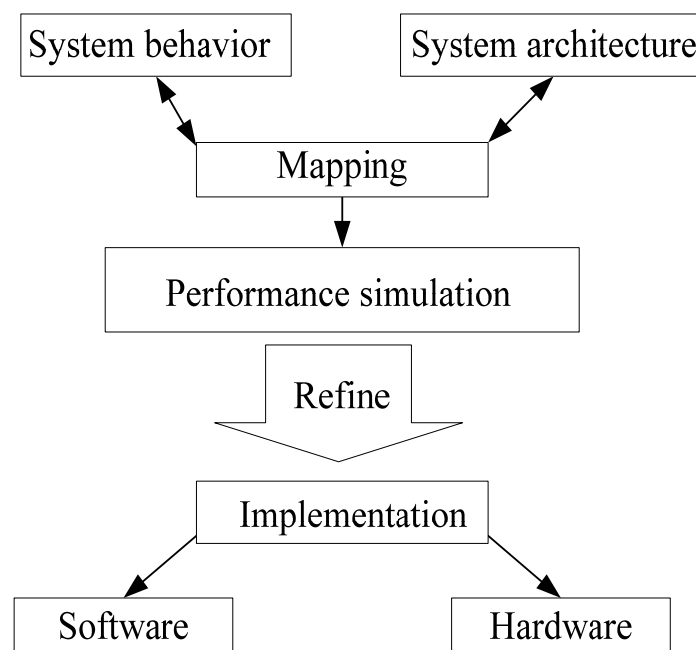


Figure 2.2 Platform-based design

With illustration in Figure 2.2, mapping is the iterative process which embodies the high-level abstraction in platform with help of evaluation tools. Then the result of performance simulation will direct the further modification until final implementation.

For platform-based embedded system design, several important steps are listed and explicated as well:

1. Task level concurrency management: Specification of design should be reviewed according to platform's features, so merging or splitting tasks are needed.
2. High-level optimizations: The methods to explore the maximum potential of software work includes: data conversion, loop transformation, array folding, etc.

3. Hardware/Software partitioning: As shown in Figure 2.3, hardware/software partitioning roots in system behavior and also takes the platform into account. Being an iterative approach, the feedback from the simulation for software compilation and hardware synthesis would effect on partitioning again until satisfied result is gained.
4. Compilation: The characters of embedded system demand the compiler to pay more attention on timing constraints, energy consumption or even thread distribution in multi-core system.
5. Scheduling: An approximate scheduling is arranged in hardware and software partitioning while after final coding precise scheduling could serialize executions.

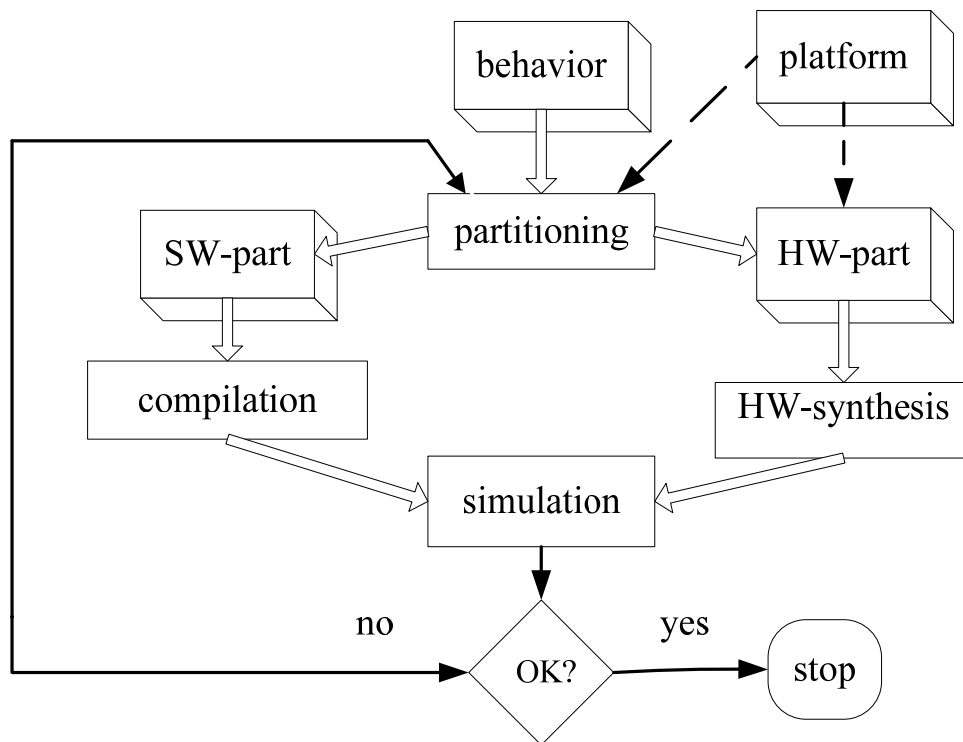


Figure 2.3 Overview of hardware/software partitioning

2.1.3 Validation

Finished design is far away from finished product because there is one step we can not ignore: Validation. Although difference simulation tools are applied in design, the final work could still be brittle because of the distance between reality environment and theoretical modeling. In validation there are several measures to remedy deviations.

Simulations in non-functional behaviors mimic practical condition, including the electro-magnetic compatibility and thermal behavior. Test on design which is a group of actives: Test pattern generation; Test pattern application; Response observation and result comparison. Fault injection and simulation that are utilized to know how system behaviors when fault occurs. Risk and dependability analysis assist design avoid of damage from hazards. Only qualified design could pass all measurements and then delivered to manufacture.

2.2 SOC design

System On Chip is requiring a whole system or most core components could be realized in a chip. It usually comprises [4]:

- One or more microcontroller, processor or DSP core(s)
- Timing generator as oscillator or phase-locked loops
- Memory blocks including ROM, RAM, EEPROM and Flash
- Peripherals including counter-timers, real-time timers and power-on reset generators.
- External interfaces including industry standards such as USB, FireWire, Ethernet, USART, and SPI.
- Analog interfaces including ADCs and DACs.

Comparing with common embedded system design, SOC has its own features:

1. High complexity and integrality. It is easy to understand that enormous transistors assembled in such a limited chip. Hence components in reusing and platform-based design are a natural solution to meet this challenge.
2. Low power consumption. SOC is widely adopted in portable devices for its small size while high speed running in a carried battery deserves low power consumption.
3. Mixed analog /RF signal design. Usually as a separate part in embedded system, analog/RF signal component could bring many new matters which designer have to handle with.
4. Testing in embedded IP. By applying the third-party IP, the difficulty of design could be less. But for testing, new methods should be implemented for others' creature.
5. Low power RTOS, chip package and process technology. All the issues are unnegligible that designers should take care of.

Moreover, the highlights in design are including:

1. System level hardware description language also object-oriented, is an ideal choice to get high efficiency in coding and good for software design as well. But it is still in development because not all hardware components can be well depicted at the moment.
2. Based on system task graph, task scheduling and resource allocation is the best way of SOC architecture design, effectively resolving the long-term problems like hardware/software partitioning [5].
3. Multi-voltage design and energy-aware multi-processor design are the most popular strategy to realize low power consumption.

2.3 USB

2.3.1 Introduction

Traditional devices for contacting peripheral equipment and computer are the Parallel port and Serial port. The serial communication protocol is RS-232, which is suitable for distance within 15 m and transmission speed as high as 20KB/S. For Parallel port, high speed is on the cost of even shorter distance and high complexity in developing. Besides, loose limitation on protocols and various standards make the designers and manufacturers difficult in production. All these obstacles force the manufacturer to introduce a new communication protocol—Universal Serial Bus (USB). Soon, USB has become a first choice in the application for low/medium speed transmission between computer and periphery devices. Its main advantages lie in [6]:

- High speed: In USB 2.0 standard, the high data transfer rate is 25Mbps-400Mbps
- Bus topology: The architecture consists of Host, Hub and Node. Hub behaves as key role in architecture: It could not only connect the Host and the Node but also support low-level hub. In one USB topology network, it can be as most as 4 level hub and 127 periphery devices.
- Plug and Play: It basically appears as two traits: hot swapping and auto-configuration. Depending on the improvement in physical design, hot swapping becomes an easy conduct. And auto-configuration is more relying on the support of software drivers.
- Low power consumption: Especially for bus-powered device, the maximum current is limited below 500mA. And two working status: Normal and Suspend insure the device working properly.

All USB periphery devices are distinctly defined from the bottom physical and electronic characteristics to upper software protocols and data communication. The host and node are treated as differ objects which are formed with various modules and levels. Demonstrated in figure 2.4, the black arrowhead means the actual communications flow and gray arrowhead means the logical communications flow. Besides, devices are divided into three levels: USB bus interface layer, USB device layer and Function layer although in practice there is not any clear interface among them.

However, when moving to Host, both the operation system (like Window or Linux) and difference application functions make the circumstance change. The three levels of USB Host Controller, USB System Software (SW) and Client SW independently exist and work. The interfaces between them are also well defined and strict in modification. The benefit is so visible that the standard device or host could make USB design compatible for all manufacturers. Moreover, the designer could more concentrate on partial work

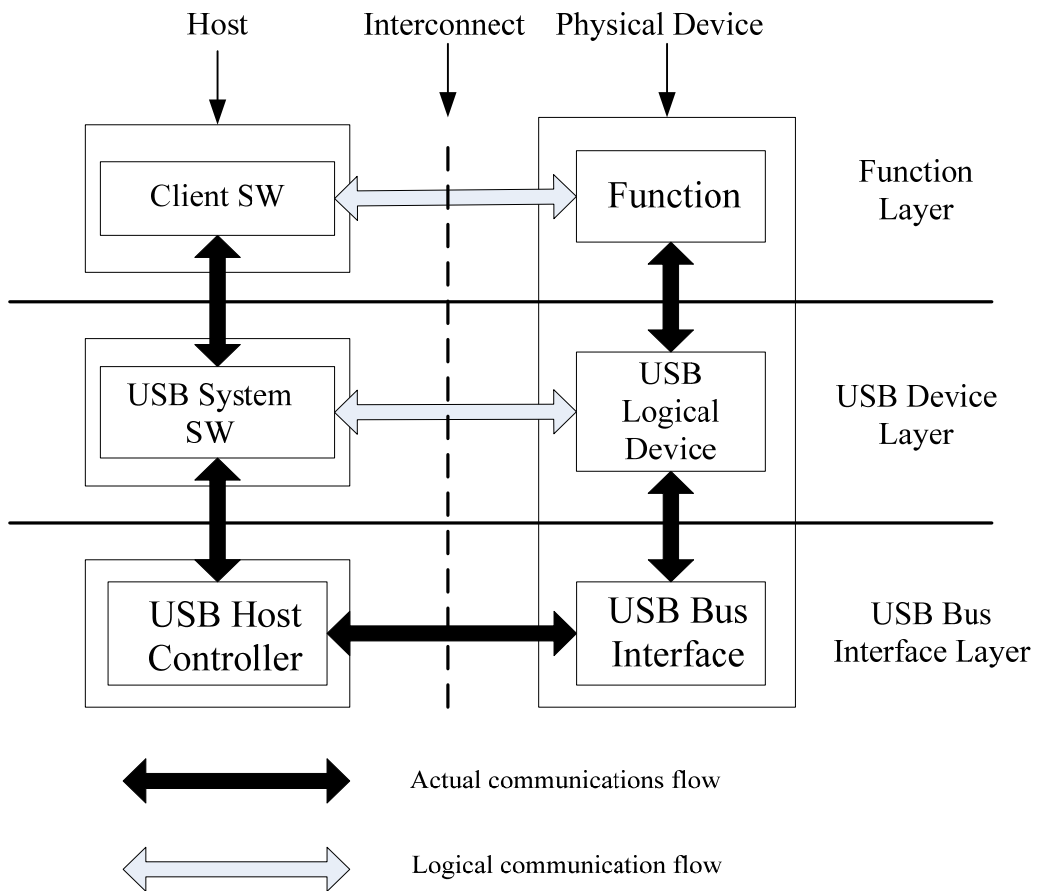


Figure 2.4 Object modules of USB

2.3.2 USB communication protocol

For the beginning, two essential concepts should be introduced:

- Endpoint: Each USB device is considered as a set of points while Host could only communicate with each device through diverse endpoints. And each endpoint is a data buffer with fixed size which is configured in manufacture. Its important features includes: transmission mode, bandwidth, mark number, the

maximum capacity for one packet, etc. All endpoints are activated only after configuration. Endpoint 0 is customarily used to initialize the parameters while endpoint 1, 2 is applied in data transfer between Host and devices.

- Pipe: A pipe is a logical concept to describe the connection between a data buffer in Host application software and an endpoint in periphery device.

The periphery device is configuring the endpoint 0 to create the pipe. On the other hand, the information of the periphery device, including: device type, power management, configuration, endpoint description, is obtained from the pipe. Therefore, once a device is connected to USB and powered, the endpoint 0 is accessible and the corresponding pipe is paved. Consequentially the device is identified by Host and the data transfer starts.

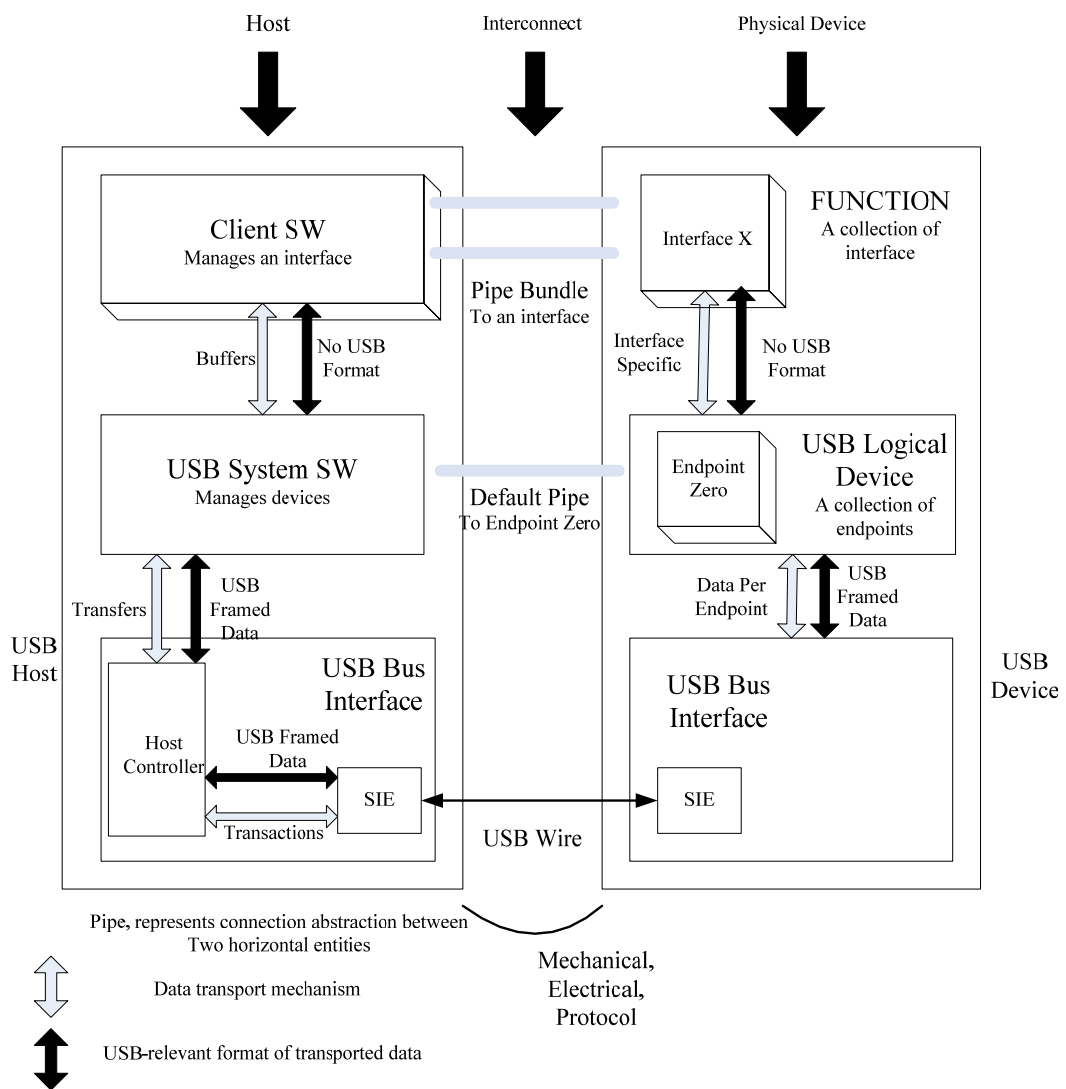


Figure 2.5 USB data flow

In Figure 2.5 above it has demonstrated the data flow in USB transmission. In logical sense, the data transfer is carried out in pipe which is under the supervision of the USB drives. In practical, the device driver is sending out the Input/Output Request Packet (IRP) from USB Drive Interface. When the request is received, USB drive in Host would interpret the I/O Request Packet into USB transfers through Host Controller Driver Interface (HCD). Then USB system software would exchange the data with high-level Client software with sharing buffers. Inversely, the USB transfers is decomposed as Transaction and sent out as Packet from Host to device.

In addition, the concept Frame is introduced in the figure. It is a unit of 1 ms for all activities in Bus transfer. In respect that the project does not refer to the research on Packet level, the relevant information will not be presented here as well as the Serial Interface Engine (SIE) which is in charge of realization of physical protocols. When comes to the transfer between two USB SIE, there are four kinds of transfer modes in USB standard to meet the requirement in different applications:

Control Transfer	Usually used in configuration, setup or status, the last two are in USB own format. Maximum capacity for each packet is either of 8, 16, 32 64 bytes. Retransmission is available in case of failure.
Isochronous Transfer	A periodic and continuous transmission in real-time required task. On direction for each endpoint and capacity for packet can be as high as 1032 bytes. Maintain a fixed data rate and have a certain error tolerance.
Interrupt Transfer	Non periodic and occasional transmission for small amount data from devices to host. No USB specific format for data and retransmission available if necessary.
Bulk Transfer	To transfer large amount of data while time is not critical. One direction for each endpoint and retransmission is available to promise the quality and veracity.

Table 2.1 Four modes in USB transfer

By observing their distinctness, it is naturally to find that: Isochronous Transfer is best for real-time video or audio application and Bulk Transfer is suitable for Printer or Scanner servers. But the Control

Transfer and Interrupt Transfer are fundamental mode and the project for the thesis also relies on.

2.3.3 Enumeration and USB driver

Enumeration is a term to describe the process that how the Host recognize a USB device. Therefore, a good understanding of enumeration is the first step to develop a USB application. The whole process can be briefly presented as follows:

Hub is the component in Host to detect whether there is a connection or not, and it has an Interrupt flow to report the Host. When the operation system is activated or periodic scanning, the Host would enquiry its Hub what kinds of device is/are connected. Once a new device is detected, Host sends a series requests to Hub for building a pipe between Host and device where Enumeration begins. It consists of several actions: allocate an address for new device; read the descriptor from it (descriptor is a set of message containing useful information of device and Host also depends on it to identify different devices); allocate and load the corresponding driver and configure the device by the receiving data. As far as the configuration is done, the device has been successfully recognized and waiting for coming employ.

Software in Windows runs in either two modes: user or kernel. Application is booted in user level while USB drivers should run in kernel mode in which drives is authorized to permit or deny an application to access a device [7]. When applications visit client derives, the Windows API function play as a bridge and Win32 system is the manager of communication process.

In Windows Drive Model (WDM), as a layered architecture, it can be sorted into two classes for a USB device: Bus drive and function drive. The former is supplied by operation system which lies beneath function drive. It is working with hardware to implement the complicated down-level communication. To a function drive, it is usually built by designers, on the top of bus drive. Hence the main task is not relating any hardware but responsible to send IRP, which requests a single input or output action. A concrete explanation is show as Figure 2.6.

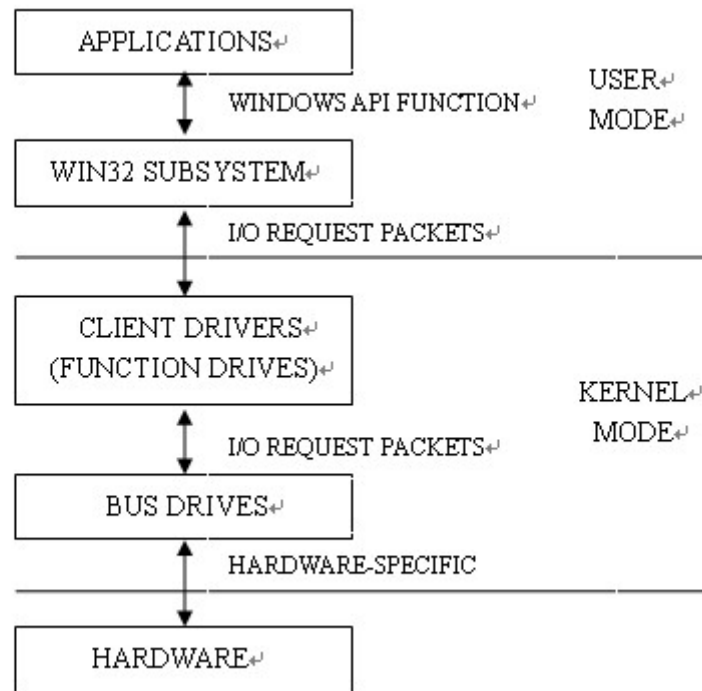


Figure 2.6 A layered driver model for USB under Windows OS

2.4 Embedded system implementation platform

2.4.1 Hardware platform

To fulfill the demands for design, the Altera DE2 Development and Education board is selected and the actual picture is presented here:

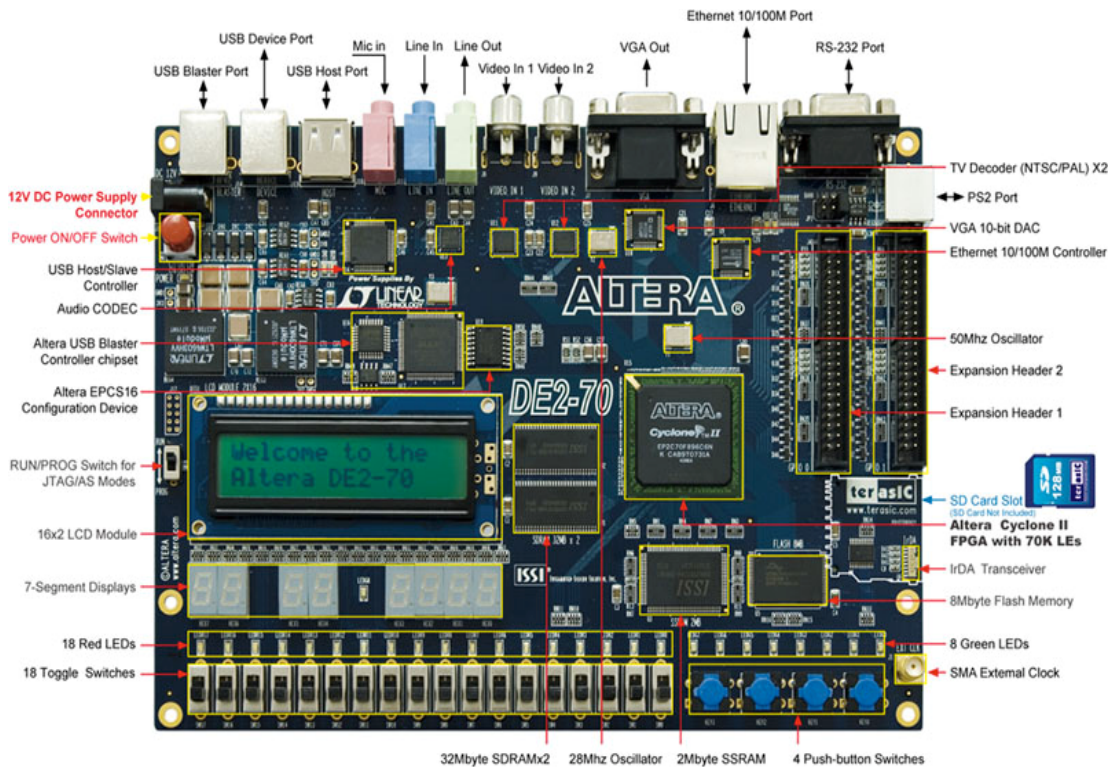


Figure 2.7 A platform of Altera DE2 board [8]

The figure 2.7 above which is labeled elaborately is the instrument where the embedded system would be constructed. Among all components indicated, only the ones relating to the project would be depicted here:

- Altera Cyclone® II 2C35 FPGA device. It contains a number of 33 216 Logic Elements, 483.840 RAM bits and 475 user I/O pins. The most important feature is that a customized softcore “Nios” is supported within the FPGA. There are three types of CPU: “Economic”, “Standard” and “Fast”, which acquire different portion of hardware resources and behaves in different performance that are chosen according to demands.
- Memory chips: SRAM organized as 256K X 16 bits; SDRAM organized as 1M X 16 bits X 4 banks and Flash as 4 M byte NAND flash memory.

- Periphery interfaces chips: 10/100 Ethernet controller; USB Host/Slave controller and serial ports.

2.4.2 Software environment

Quartus II is the integrated developing tool for PLD design by Altera, which supports several types of design inputs like schematic or HDL. With the embedded synthesizer and simulator, all work flow from design composing to hardware configuration could be realized.

However to build a system there are more tools should be applied in design. SOPC Build, a sub developing environment in Quartus, assists the designers to fast and easily construct a system level project [9]. SOPC Builder allows the user choose standard components as CPU, memory interface and other user-defined periphery components to form the system. And SOPC Builder would help to connect all components together by modeling all separate modules and creating bus logic for connecting.

When referring to manipulating the softcore—Nios II processor, the Nios II Integrated Development Environment (IDE) is introduced [10]. It is a GUI software developing programme where all software execution including editing, building and debugging for control CPU operation is accomplished.

3. Method

To achieve a success when a new project is launched, an overall plan is plot. Especially in engineering field, a project as in this thesis work is closely attached with hands-on practice which requires a good partition of the time in theory preparation and implementation in device. As a result, the developing process can be broke down as following steps:

1. Planning
2. Theory preparation
3. System development
4. Validation

The steps although is ranked as a sequence, iterative work is often exited as well.

The first phrase of the planning, the motivation for the project is the guide for the plan to revise the timing, resources and other requirements. What sort of theory should be aware? What kind of reference should be prepared? What is the topology for system modeling? All the relating issues are taken into account though far from concrete.

For theory preparation, attempt to find the materials which would be most similar to the project is important. Therefore the reference for embedded system design, USB protocols and Visual C application are in the scope. Certainly to pursuit a whole coverage of these subjects is impossible. But obtain primary instructions and a notice in relevant chapters is a good start.

In system development, the bottom-up thinking is to construct the whole system by “connecting” the each component which is devised to realize some functions. It is called Block Base Design as well. However among the detail work for each component, the theory study and design execution is always in accompany.

The final phrase, validation, is the stage to test the function behavior for the system and discuss whether the final target is satisfied or how much progress is made.

4. Design and Implementation

In this chapter, the specific work for the system would be presented while whole system is decomposed into three blocks from their diverse function. The implementation environment both the hardware and software would be introduced as well as the relating information. Then the key steps of the design are demonstrated which are combined with some testing functions. In short, a technical view for the system is offered.

4.1 System description

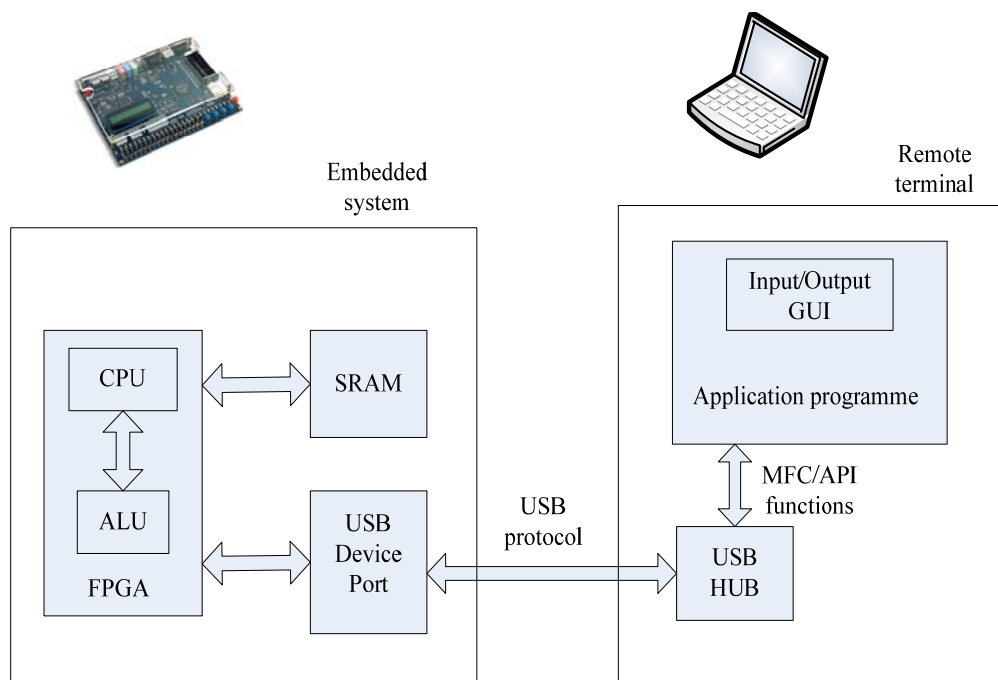


Figure 4.1 System overview

From the demonstration as Figure 5.1, we can have a general picture of the system. In practice there is a suit of Altera DE2 Development and Education Board which is the target for the implementation of embedded system. Its main task is responsible for data process and storage. While for remote terminal, a PC with Windows Operation System is prepared to build a control panel. Certainly a communication channel is necessary to connect two parts.

In Embedded system part, the Altera DE2 platform is capable to support a well-developed soft core Nios which could behavior as a

CPU in the system. And the data process could be achieved by ALU in the FPGA as well. Moreover, the SRAM is added as the memory for the data read and write. By considering the complexity and reality of the design work, USB is a most proper measure among several communication devices which the platform supports.

Naturally in Remote terminal, the application programme should contact well with embedded system by USB channel. Also the required function of the terminal include that in Input/Output GUI, 8-bit operands are typed in as well as the required algorithm can be selected. After confirming the instruction, the result could be obtained and displayed immediately. Besides, the read and write on data saved in embedded system are available. The Microsoft Visual developing tool is adopted to construct the terminal due to its good support for GUI programming and API functions which shield the low-level design for USB protocols.

4.2 Embedded system

4.2.1 ALU

The ALU is always a kernel component in CPU which carries out the basic arithmetic computation. Here the ALU would be built as a separate part from the CPU and its functions are defined as: Adder, Logic Calculation and Shift. From the Figure 4.2 below, a classical design would bring an intuitionistic image.

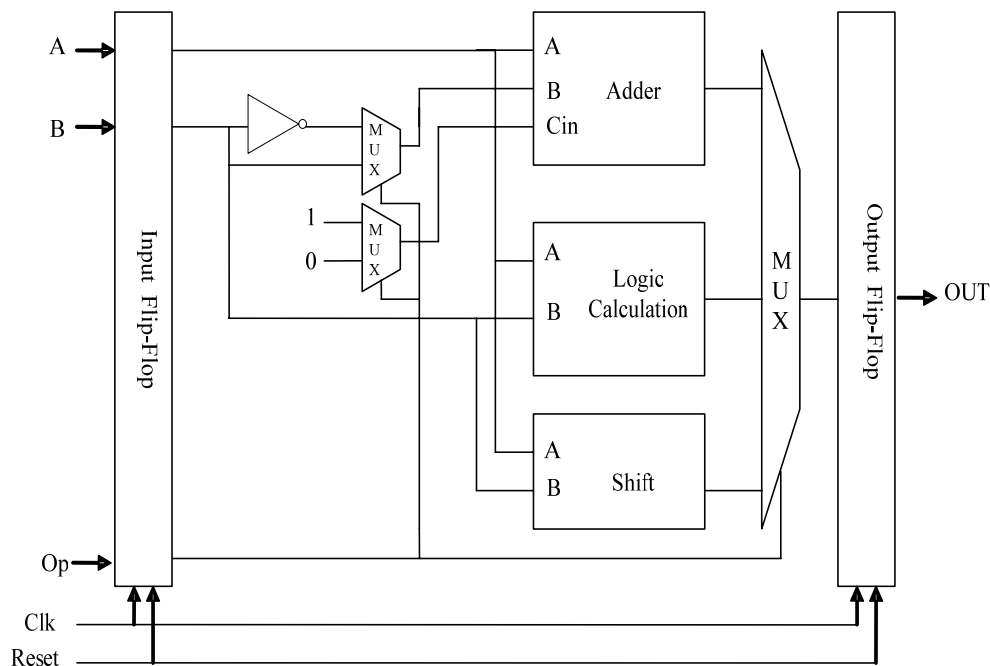


Figure 4.2 Function blocks of ALU

For the input signals, there are two operands: A and B, and one operator: Op. All the signals are delivered from CPU and the result "OUT" would be sent back when available. In execution step, there are three function blocks in charge of the responding computation. The adder performs four sorts of calculations: signed add, unsigned add, signed subtract, unsigned subtract which are determined by two mux. In logic calculation, AND, OR, NOR and XOR are implemented while in Shift block, we have signed left shift, unsigned left shift and signed right shift.

To test the ALU, the design is moving to ModelSim where a test bench is constructed. For each arithmetic 20 groups of test vectors

are applied to insure the correctness. After that, the ALU is saved as one project in Library for the further using in embedded system.

4.2.2 Memory and “Nios” CPU

For the memory to read/write the data, the DE2 board has provided three chips to select:

- 512-Kbyte SRAM
- 8-Mbyte SDRAM
- 4-Mbyte Flash memory

In practice SRAM is chose which is organized as 256K X 16bits. The function “IORD” and “IOWR” are the tools applied by CPU to access the memory that thanks to the Hardware Abstraction Layer (HAL) system library [9]. HAL create a simple device driver interface for communication with the underlying hardware, so the visiting for a memory becomes much easier that IORD and IOWR would execute the instructions to the target address. The later as the important parameters in the function are appointed in the SOPC Builder this will be discussed in following paragraph.

In the project to involve the CPU and memory as an integrated system, the SOPC Builder is started. The chief steps for setup the system with Nios soft core are performed:

1. Create a new project in Quartus and configure well based on the target board.
2. Assign the pins according the peripheral equipments (the USB controller and SRAM) used in system in form of Tcl Script File.
3. Activate SOPC builder and configure again.
4. Add the Nios soft core to the project where the standard one is adopted and configure it as default parameters.
5. Add other necessary components for the system: SRAM, Parallel I/O and ISP1362, the USB controller which manufactured by Philips. Configure them as well.
6. Generate the system.

When the system is successfully created, it means that the hardware frame is ready and the rest task is to start IDE tool for software debugging. However, by evaluating the complexity of the system, especially in USB communication, too much low-level protocols and

physical designs would cause a great deal of barrier. To conquer this challenge, a relevant demonstration project is referenced. The demonstration project is an onboard counter that the accumulation signal is transmitted by USB device port which has the similar hardware architecture. Therefore some modification on original project is carried out. To add the interfaces with the ALU block and memory, more I/O modules are created and mounted on the Avalon bus [10]. In the figure 4.3 there is the list of all components including the Parallel I/Os: pioA, pioB, pioOP; pioOUT and memory bus: sdram_0. Moreover, the base address is initialized as well which is the destination register for the corresponding functions.

Use	Connec...	Module Name	Description	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>		<input type="checkbox"/> cpu_0	Nios II Processor				
		instruction_master	Avalon Master	clk			
		data_master	Avalon Master				
		jtag_debug_module	Avalon Slave		IRQ 0	IRQ 31	
<input checked="" type="checkbox"/>		<input type="checkbox"/> tri_state_bridge_0	Avalon-MM Tristate Bridge	clk	0x00680000	0x006807ff	
		avalon_slave	Avalon Slave				
		tristate_master	Avalon Tristate Master				
<input checked="" type="checkbox"/>		<input type="checkbox"/> ISP1362	ISP1362				
		avalon_slave_0	Avalon Slave	clk	0x006810b0	0x006810bf	8 7
<input checked="" type="checkbox"/>		<input type="checkbox"/> sdram_0	SDRAM Controller	clk	0x00800000	0x0fffffff	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> pioA	PIO (Parallel I/O)	clk	0x00681110	0x0068111f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> pioB	PIO (Parallel I/O)	clk	0x00681120	0x0068112f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> pioOP	PIO (Parallel I/O)	clk	0x00681130	0x0068113f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> pioOUT	PIO (Parallel I/O)	clk	0x00681140	0x0068114f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> SEG7_Display	SEG7_LUT_8	clk	0x00681100	0x00681103	
		avalon_slave_0	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> cfi_flash_0	Flash Memory (CFI)	clk	0x00000000	0x003ffffff	
		s1	Avalon Tristate Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> epcs_controller	EPCS Serial Flash Controller	clk	0x00680800	0x00680fff	0
		epcs_control_port	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> jtag_uart_0	JTAG UART	clk	0x006810f0	0x006810f7	1
		avalon_jtag_slave	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> uart_0	UART (RS-232 Serial Port)	clk	0x00681000	0x0068101f	2
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		<input type="checkbox"/> timer_0	Interval Timer	clk	0x00681020	0x0068103f	3
		s1	Avalon Slave				

Figure 4.3 New added components in SOPC Builder

Then the SOPC Builder will generate the system according to the design requirement.

4.3 Remote terminal

Remote terminal is planting on the personal computer which is the interface between the user and the embedded system. From the terminal, user can send out the information by typing the key board or clicking on mouse which requires a user-friendly GUI. On the other hand, the terminal should work well in Windows environment and convenient in communication with USB port. The Micro Visual C++ is a reasonable tool to undertake the task.

4.3.1 MFC

MFC, Microsoft Foundation Class Library, is an important tool for designers to setup an application frame in Windows where the Windows Application Program Interface (API) is capsulated [11]. The later involves a great deal of functions that simply the communication between application software and Windows. Therefore some crucial MFC applied in project are displayed here:

- CWnd: Window, the parent class for most visible components, include: CFrameWnd (frame window); CToolBar (Tool bar); CDialog (Dialog), CButton (Button), etc.
- CFile: contains functions: Open, Read, Write file which are applied for the data transmit when a hardware device is simulated as a specific file in software.

4.3.2 GUI

The architecture of GUI is showed as Figure 4.4 that obviously the GUI is divided into three blocks which are representing different functions.

The left part is the module to setup the connection between the Remote terminal and the DE2 board. When the board is on power and connected to the computer, the driver would recognize device. Then by clicking the Connect button, the Remote terminal would scan all USB devices and point to the DE2 board. Afterwards, a report sent by firmware in DE2 would be received and the content is displayed in Text window which containing the configuration information for USB controller from the embedded system.

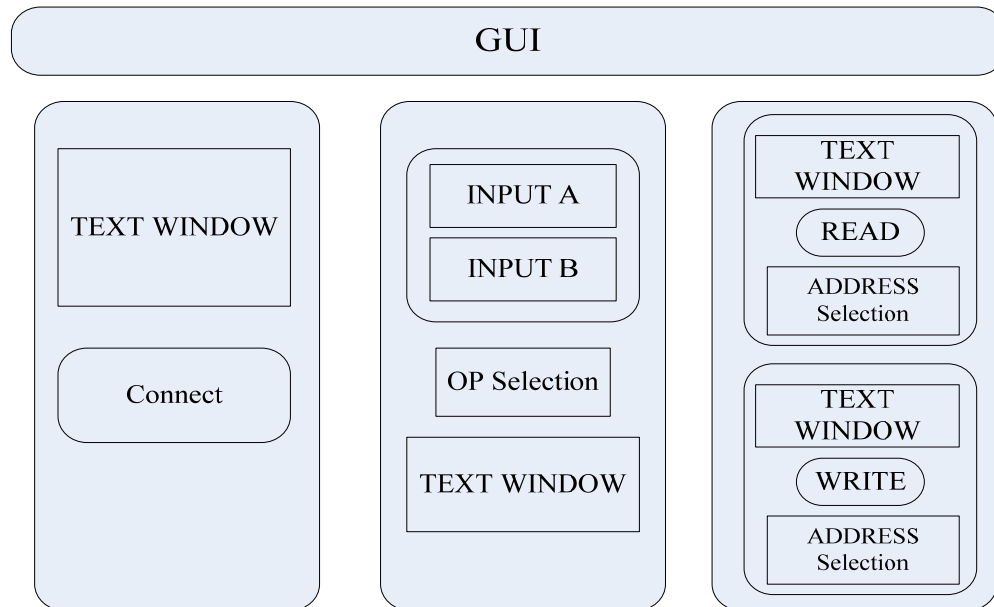


Figure 4.4 Topology of GUI for remote terminal

The middle part is in charge of arithmetic and logical computation. The input value is typed in Window A and B. Required Algorithms are selected in OP selection bar. Then the result is obtained and showed in the following Text window.

The right part is the place to access on-board memory. The assigned addresses are listed. When required address is chose in Address selection bar, the content in this cell would be presented after clicking Read button. Similar for write in memory, by choosing the required address and typing the corresponding data, the click on Write button would conduct this command.

4.4 USB transmission

Since the USB is adopted to carry out the transmission between the remote terminal and the embedded system, the task for developing the USB mechanism is becoming the substantial part in project. Based on the fundamental theory, a USB application usually consists of three portions: firmware in USB device; driver program and application program in USB host.

In this case, the configuration of firmware is a part of work in Nios programming while the application program in USB host is the primary mission for the application setup in Visual C++ environment. Due to the demonstration application utilized, the existed driver software is still in role. Consequently the following content is about the USB programming in remote terminal and embedded system.

4.4.1 USB socket in remote terminal

The application program in remote terminal is constructed under the Visual C++ environment. Since the GUI is the interface for user to manipulate the embedded system, the USB socket is the down-level component to transfer the message from human's instructions to characters in USB standard and vice versa.

Before to explore the procedure in detail, there is one concept should be mentioned here—Human Interface Device (HID). HID classes is one of the first API functions supporting by Windows for the communication with HID. There are three DLL files containing the necessary API functions: hid.dll; setupapi.dll and kernel 32.dll. However, the prerequisite to use them is that Device Development Kit (DDK) should be installed at first [12]. DDK is a suit of developing tools for Windows device driver while the detail information can be acquired in Microsoft website.

Moreover, to identify a HID, every device would be titled in a specific symbol that is Globally Unique Identifier (GUID). By pointed to a certain GUID, the corresponding HID is assigned. The data structure for the attributes of HID supplies information such as: Vendor ID (Identifier of the manufacturer); Product ID (Identifier of the product); Version Number, etc [6].

By catching the desired GUID, a piece of buffer is assigned for the pointer to store the relating information for HID. Then the handle representing the device is brought and configured as a file. Afterwards the actions of reading or writing on file take the equal result on the target USB device until the file is closed by corresponding motion. A simplified diagram is to present the all procedure as Figure 4.5.

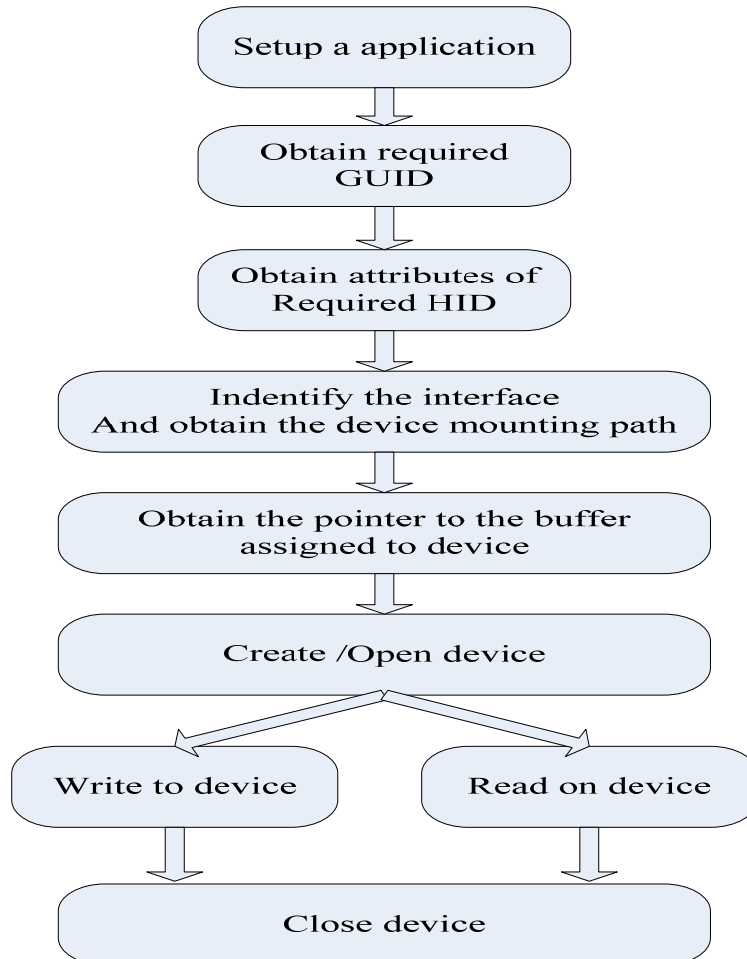


Figure 4.5 Design flow of the USB application

4.2 USB firmware in embedded system

The firmware is the configuration defined by both the manufacturer and the users to control the USB device port that the structure is clearly interpreted in Figure 4.7. The firmware is a typically interrupt-driven model that Interrupt Service Routine (ISR) hierarchizes all interruptions according to their priority and triggers corresponding operations. In the project, the flag of status in control IN or OUT would evoke ISR to carry out the right subroutine when the data transfer has happened.

The middle level, HAL4D13.C is a set of command defined by firmware to encapsulate all the functions to access the device—ISP1362 [13].

On top level, HAL4SYS.C has defined the I/O access of the device; Chap9.C is the standard file to address USB device requests within USB Specification Rev.2.0; D13bus.C is in charge for handle the specific vendor requests. Mainloop.C needs to check all event flags and hand it to the right subroutine to process [14]. Like in this project, the receiving data would induce the mainloop program passing to ALU block for further processing.

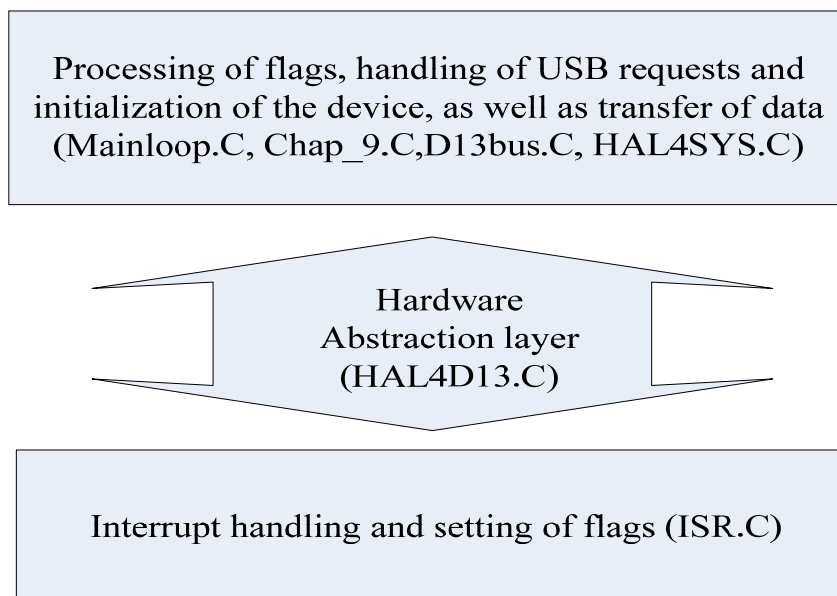


Figure 4.6 Software scheme of the firmware in USB controller

Before probing into the mechanism of ISR's function in firmware, two basic definitions is quoted here: An IN data transfer means the

transfer from the USB controller chip to external USB host, which is the personal computer here while an OUT data transfer means the transfer from the USB host to USB device controller chip(ref. datasheet for USB_ISP1362).

The evoking of ISR initiates with the read from the interrupt registers in USB device controller which are ranked according to their priorities. The first interrupt information is the Bus Reset signal as the device reset configuration while the second one is the Suspend status signal to confirm the working request on present device. Then the device should tell the main function whether the request is the IN data transfer or OUT data transfer. For the Control IN handler the function would clear the interrupt bit and judge whether the buffer is empty. If not, the status is keeping in Writing until the last packet arrived. And if the last packet is not the over flag, the buffer would record with remaining data size for the next term transfer. On the other hand for the Control OUT handler, the function would clear the interrupt bit first and judge whether the buffer is full. Then if the receiving data is the setup packets, the setup program would take over the information otherwise the status is in Reading on the buffer. To achieve the data transfer between the USB device and the Nios processor, the Endpoint 01 and Endpoint 02 is activated. The interruptions described above have constituted the whole ISR and the flowchart as Figure 4.7 would display a more intuitional impression.

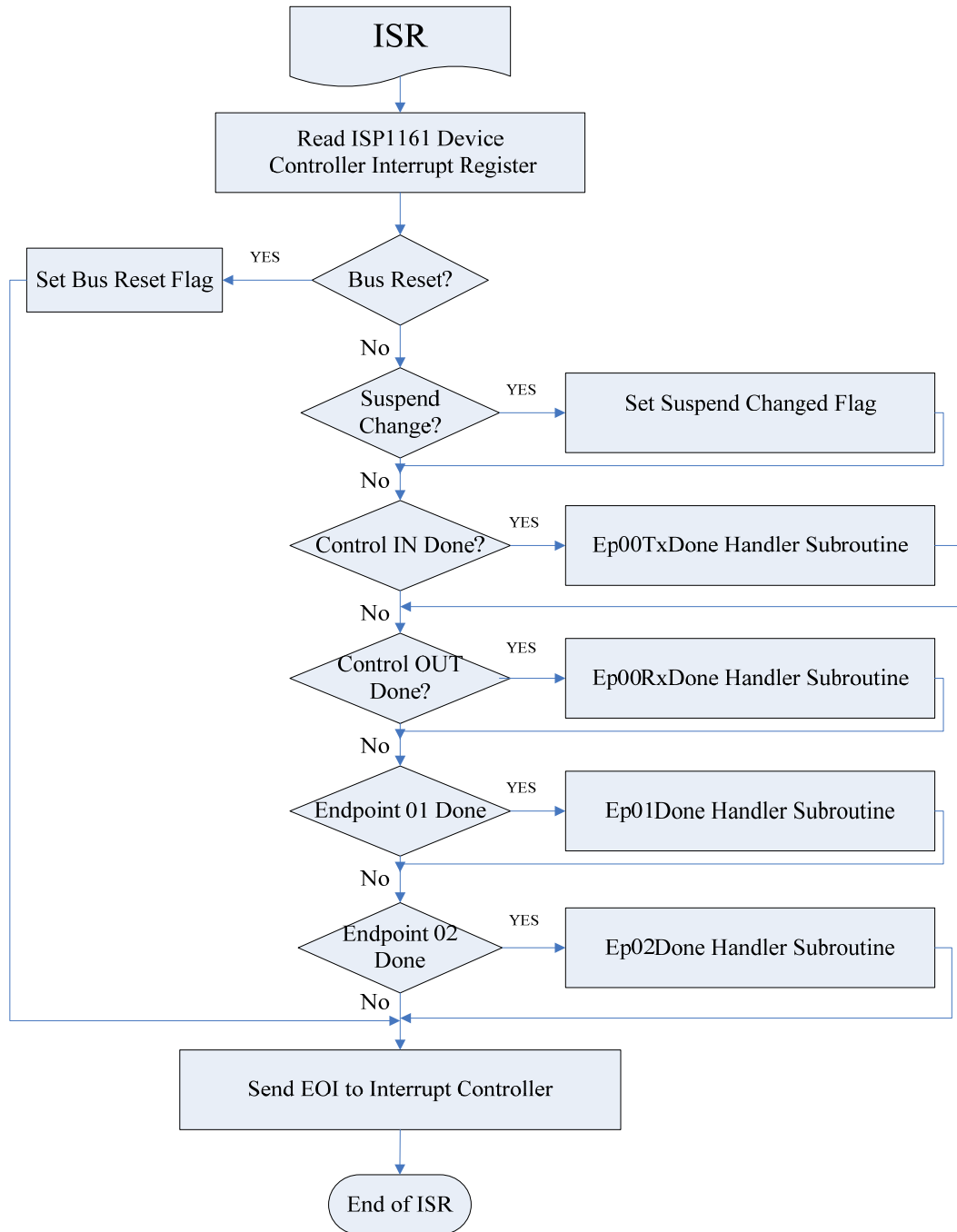


Figure 4.7 Work flow of ISR

5. Discussion

5.1 Conclusion

The motivation to commence this project is originating from the education demands that a straight and explicit embedded system model would be easy to fascinate the eyes of new students in Electrical Engineering. To build up a user-friendly interface for control and demonstration, a remote terminal is required in personal computer while the embedded system is implemented on Altera DE2 board. The initiate task for embedded system is to realize an elementary data process and storage which requests a CPU for schedule, an ALU for computation and a memory to save data. However, how to connect the embedded system and the computer is a choice-test that there are three candidates: RS232, Ethernet and USB. After carefully comparing the pros and cons of three methods, USB is finally selected as the communication channel to take.

The first barrier in design is how to integrate all components in DE2 board although as a separate part, design and test are not tough task. Especially for the soft core "Nios", it behaviors as the CPU so that all other components should be well joined. Fortunately a demonstration program is existed which try to utilize the USB function. So adding the own design work on old program is a proper solution in order to insure the system's validity and reduce the work in hardware mapping. However the cost is that too much unnecessary design is brought.

By the USB becoming the solution as the communication channel, a considerable part of work has been invested on exploring the USB protocols and its applications. The developing tool in personal computer is Visual C++ which is easy to access and huge resources can be found on net. Therefore the application program is accomplished which has referred some coherent design. However the testing for the data transmission is not ideal as imaging. The data from Remote computer down to embedded system on board is normal while in opposite way the computer can not receive anything. After several serious reviews of the design on both sides of USB port, the most puzzling obstacle is in the driver which is supplied with the demonstration program. In spite of unfinished dual-transmission, certain progress has been made in the application for USB communication.

5.2 Future work

To achieve the initial design, which relies on USB channel for communication, a full and intensive research on USB protocol is needed. In particular a proper driver program is the foundation to build the upper application.

Moreover, Ethernet is also a widely used solution for remote control system. Transplant the project to a Ethernet application environment would be meaningful as well.

On the other hand, to mimic the real status of the CPU operation, more devices can be added. The ALU could take more functions and even one more CPU can be utilized to form a dual-core system.

6. References

- [1] Michael Barr. "Embedded Systems Glossary", *Neutrino Technical Library*.
URL:<http://www.neutrino.com/Embedded-Systems/Glossary>
- [2] Marwedel,P. Embedded system design, 2005
- [3] Sangiovanni-Vincntelli,A The context for platform-based design, IEEE Design & Test of Computers, page 120.2002
- [4] System-on-chip, Product Technologies, STMicroelectronics
URL: <http://www.st.com/stonline/products/technologies/soc/soc.htm>
- [5] Shao, JW. Key technologies of system on chip design, Science in China Series F: Information Sciences, 2008
- [6] Universal Serial Bus Specification 2.0
URL: <http://www.usb.org>
- [7] Axelson,J. USB Complete, Third Edition, 2005
- [8] Altera DE2, FPGA product, Terasic Technologies
URL:<http://www.terasic.com.cn/cgi-bin/page/archive.pl?Language=China&CategoryNo=40&No=31>
- [9] Introduction to the Quartus II Software, Altera, 2009
- [10] Nios II Software Developer's Handbook, Altera, 2005
- [11] Chivers, I. Essential Visual C++ 6.0 fast: an introduction to Windows programming using the Microsoft Foundation, 2000
- [12] About Developer Tools, Windows Hardware Developer Central, Microsoft
URL:<http://www.microsoft.com/whdc/devtools/WDK/default.msp>
- [13] Philips, USB_ISP1362 Datasheet, 2004
- [14] Kunzang.D, USB Embedded Programming Guide, 2002