ASH1: a Stack-Based Input/ Output Processor for USB Operations

Abdullah Al-Dujaili[#1], Lo Hai Hiung [#2], Shawn Tan [*3]

[#]*Universiti Teknologi PETRONAS*
Perak, Malaysia
[1]ash.aldujaili@gmail.com
[2]lo_haihiung@petronas.com.my
[*]*Aeste Works (M) Sdn Bhd*
Kuala Lumpur, Malaysia
[3]shawn.tan@aeste.my

***Abstract*— This paper describes a work in progress: ASH1, an 8-bit input/ output processor (IOP) that is designed to be able to perform USB operations. It has a stack-based architecture where most of the operations are done on the top elements of the stack. The instruction set consists of 17 14-bit instructions optimized for framing and driving software code. ASH1 communicates with the main processing unit (Master CPU) through a wishbone bus. It has been proven reliably at 50 MHz in an Altera Cyclone II FPGA device. With around 1400 FPGA slices and a maximum clock frequency of 90 MHz, ASH1 could make a good substitute for big USB IP Cores. Future work includes making ASH1 MAC Ethernet capable and USB2 compatible.**

***Keywords*— input/ output processor, usb, stack, wishbone bus.**

I. INTRODUCTION

The input/ output subsystem is an essential component of any computer system nowadays, as shown in Fig 1. It provides the necessary means of communication between the central processing unit and the external environment. It supplies the computer with the capability of receiving information from an outside resource, delivering them to the user in an appropriate form and vice versa via input/ output (I/O) devices (peripherals) [1].

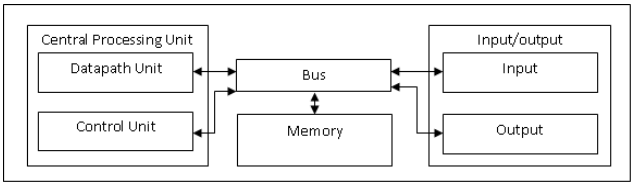

Fig. 1 A typical computer system

I/O devices vary in data rate and this might cause a bottlenecking effect due to the vast gap between the very fast CPU and the relatively slow I/O devices. Therefore, different modes of data transfer have been developed since the early history of computers to deal with that problem. In some modes, the CPU acts as an intermediate unit to control the transfer of data, whereas in other modes the transfer takes place directly between the peripheral and the memory unit that include polling, interrupt, direct memory access (DMA) and input/ output processor (IOP).

An IOP is a processor (controller) dedicated to I/O operations and it does not need to be configured entirely by the master CPU. Moreover, it can fetch its own instructions and communicate with the peripherals.

II. PROBLEM STATEMENT

General-purpose processors (GPPs) have one profound problem that is the imbalance between the I/O and processing capabilities. Complicated operations can be performed by such processors faster than loading or storing results. Therefore, plenty of these processors' tasks are I/O bounded which result in a performance bottleneck and inefficiency [2].

Such drawback can be alleviated by adding up an I/O controller (IP cores) that will relieve the master CPU from I/O tasks. However, the architecture area should be considered. For instance, AEMB processor core consumes about 1500 FPGA slices while a USB 1.1 IP core takes up about 2700 FPGA slices [3]. If we want to make AEMB USB-capable, we need to add up a controller that is almost double the size pf the AEMB processor core [4] and this is where ASH1 comes to address and settle this problem by implementing the I/O operations virtually in software with a minimum hardware support.

III. THE ASH1 IOP

A. Architecture

ASH1 is a special-purpose processor designed for I/O operations (currently, it can carry out USB 1.1 operations). It is implemented using a stack-based architecture in which most of the operations can be carried out on the top of the stack (TOS).

A block diagram of ASH1 is shown in Fig 2 in the next page. It consists of an 8-bit data stack of 8 elements, a control unit, a 11-bit program counter, an arithmetic/ logic unit (ALU), a cyclic redundancy checksum (CRC) unit, an 8-bit input port, two 8-bit output ports, a wishbone bus interface unit, a register file of 16 8-bit registers which can be used as a configuration registers for ASH1 and master CPU to use as well as a temporary storage, two 2048 bytes FIFO buffers to store incoming and outgoing data payload, and three 8-bit flag registers. These components are interconnected among each other via two buses; an 8-bit data bus (dbus) and an 11-bit address bus (abus) [5].

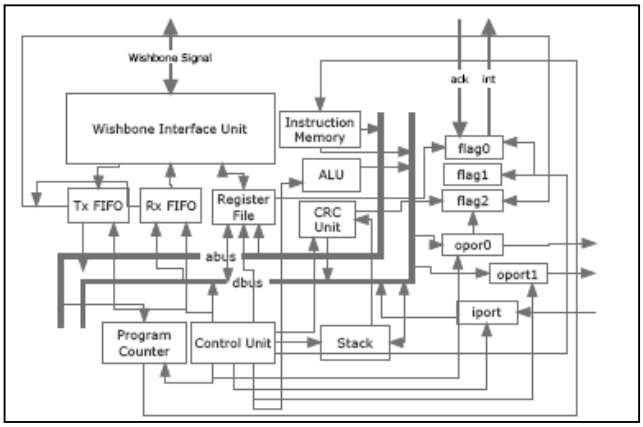

Fig. 2 ASH1 Architecture

B. ASH1 Data Stack

Data stack represents the main axis upon which ASH1 operates. The stack-based architecture was chosen because of the fact that it is a reduced-operand set architecture which makes the processor as well as its instruction set simpler in design and smaller in size [6]. Procedure calls can be implemented efficiently, as the working parameters are always on the stack, thus reducing subroutines overhead [7].

ASH1 data stack consists of 8 elements of 8-bit wide each. ALU instructions can access the top 4 elements. I/O instructions access the TOS. In case of a USB packet transmission, the LSB of the TOS is sent over the output port with the Non-return to Zero, Inverted (NRZI) differential encoding. In the case of a USB packet reception, the MSB of the TOS is loaded with the incoming bit.

C. Communication Protocol

For any I/O operations, the software involved can be subdivided into three levels:

1) High-level (Application) Software, which makes use of the data being exchanged for various purposes.

2) Medium-level (Framing) Software, which is responsible for creating the frames/ packet with which the I/O protocols conform.

3) Low-level (Interface Driving) Software, which is responsible for driving and communicating with the interface units (PHYs) that link up the I/O devices (Peripherals).

As an IOP, ASH1 is designed to execute the framing and interface driving software. The high-level software is the task of the main processing unit (master CPU) with which ASH1 would co-exist. Therefore, they should be able to exchange data and communicate with each other. This achieved via the wishbone bus [8] and a set of interrupt signals. The wishbone datasheet for ASH1 is shown in table 1.

TABLE I
WISHBONE DATASHEET FOR ASH1

WISHBONE DATASHEET for ASH1		
Description	Specification	
General Description	8-bit slave input and output port	
Supported Cycles	Slave READ/WRITE	
Data Port Size	8-bit	
Data Port Granularity	8-bit	
Data Port, Max Operand Size	8-bit	
Data Transfer Ordering	N/A	
Data Transfer Sequencing	Undefined	
	Signal Name	**WISHBONE Equivalent**
	clk	CLK_I
	reset	RST_I
	strb_wb	STB_I
	we_wb	WE_I

	ack_wb	ACK_0
	data_i_wb	DAT_I()
	data_o_wb	DATA_O()
	addr_wb	ADR_I

Three entities act as a message centre between ASH1 and the master CPU that are:

1) Tx FIFO: The data in this buffer is written by the master CPU only and read by ASH1 only.

2) Rx FIFO: The data in this buffer is written by ASH1 only and read by the master CPU only.

3) Register File: whose memory cells can be accessed (read/ write) by both ASH1 and the master CPU.

Tx and Rx FIFO buffers require no protocol as the data integrity is safeguarded (read or write is only from one party). However, the register file needs a specific protocol agreed between ASH1 and the master CPU, for instance, ASH1 can read all the registers but write to a specific ones only and the same goes with the master CPU.

D. Instruction Set

ASH1's seventeen instructions are 14-bit wide. All instructions take one cycle except one instruction (CRC). There are nine categories of instructions: literal, call, conditional jump, jump, ALU, memory access, I/O, CRC and bit manipulation.

Literals are 8-bit. All target addresses for call, conditional jump and jump are relative 8-bit wide addresses with an additional sign bit which allows to jump within ±255 address locations around the call, conditional jump or jump instruction.

ALU instructions can access the top 4 elements of the data stack on which it can perform AND, OR, XOR, shift to the right, increment and decrement operations.

Memory access instructions include fetching/ storing from/into the register file as well as storing into the Rx FIFO and fetching from the Tx FIFO.

I/O instructions deal with the top of the stack to drive two output ports and one input port with the suitable encoding that can be Non-return to Zero, Level (NRZL), NRZI or NRZI differential encoding.

Aside from the I/O instructions that can be used to encode outgoing USB packets and decode incoming USB packets, CRC instructions can be used to generate and check USB CRC-5 and CRC-16 fields.

Bit manipulation instructions are used to set/ reset flags useful for ASH1 operation and for generating interrupt signals to the master CPU. This category also helps in dealing with single bit input signals that might be used to detect an error state or a specific condition.

E. USB-Specific Operations

As it was mentioned before, ASH1 is built to execute framing and interface driving software codes. Therefore its instructions were optimized to meet this objective. At this point of time, ASH1 instructions have been design to accommodate for USB operations and as the following:

1) *USB Protocol:* This includes, for example, acknowledgement procedure, error detection procedure, bit stuffing / destuffing, retransmission time, time between packets and specifying the appropriate packet to send. All these rules can be implemented in software with the jump, conditional jump and ALU instruction categories.
2) *USB Packets Composing/ Decomposing:* The master CPU provides ASH1 with the necessary data components like target address, data payload through the wishbone bus. Other packet fields are built up by ASH1like SYNC and CRC field through the use of literal, CRC, ALU and memory access instruction categories.
3) *USB Packets Transmission and Reception:* ASH1 I/O ports are hardware-equipped with NRZI differential encoding/ decoding used for USB packets over the physical line. However it is the programmer's responsibility to identify the start and end of a packet through the use of I/O, ALU, bit manipulation, call, jump and conditional jump instruction categories.

F. Size and Performance

ASH1 has been FPGA proven on Cyclone II, EP2C35F672C6 device using Altera DE2 board with the maximum clock frequency present on-board (50 MHz). It consumes 1400 FPGA slices and has a maximum running frequency of 90 MHz.

Table 2 summarizes ASH1's architecture speed and area versus various USB IP Cores [3], [9], [10]. Fig. 3 is a normalized scatter plot comparing their

performances. In terms of resource utilization, it's apparent that ASH1 utilizes the least. In terms of operating frequency, ASH1 functions at a relatively high frequency. Despite this high frequency, ASH1 is currently able to perform low speed USB and probably full speed USB as well. USB 1.1 has a data rate of 1.5 Mbit/s for low speed and 12 Mbits/s for full speed [11], thus with a maximum clock frequency of 90 MHz, ASH1 can have up to 60 instructions for low speed and up to 7 instruction for full speed to manipulate a single bit for its transmission or reception which is considered considerably flexible especially for the low speed data rate. However for USB 2.0 and USB 3.0 data rates, ASH1 has to be running on a much higher frequency than the current one.

TABLE II
ASH1 ARCHITECTURE AREA AND SPEED ON DIFFERENT FPGA FAMILIES VERSUS SOME USB IP CORES

FPGA Family	ASH1 Area (Slices)	ASH1 Speed (MHz)	IP Core	IP Core's Area (Slices)	IP Core's Speed (MHz)
Cyclone II	1281	90	USB 1.1	2617	48
Cyclone III	1306	115.5	USB OTG	4471	105
Stratix III	784	123	USB 2.0	1889	60

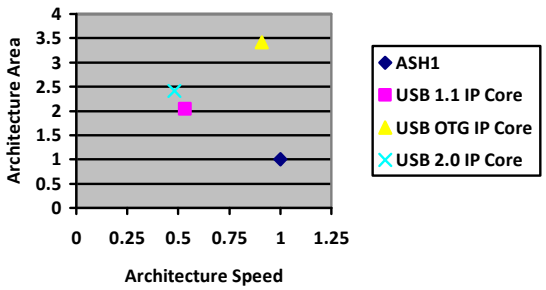

Fig. 3 Normalized scatter plot ASH1 and various IP cores architecture area and speed.

IV. FUTURE WORK

ASH1 is still under development to include other I/O operations. Some points to be considered for our future work: including MAC Ethernet operations, achieving higher clock rate thereby being able to perform USB 2.0 and USB 3.0 operations, and developing an assembler for ASH1 instruction set.

More details about ASH1 development can be found at Aeste Works (M) Sdn Bhd website/ code blog: http://www.aeste.my/

V. CONCLUSION

With its rich instruction set, size and speed, ASH1 makes a good substitute for most of the present communication controllers (IP cores) whose considerable size is a major drawback. The only requirement needed with ASH1 is a profound knowledge of the I/O protocol being used along with excellent programming techniques that would make ASH1 run efficiently.

ASH1 is currently able to perform USB 1.1 operations and future work is planned to make it MAC Ethernet capable and USB2 compatible.

REFERENCES

[1] D. Nasib S. Gill and J.B Dixit, Digital Design and Computer Organisation. Firewall Media, 2008.

[2] Bob Walsh, TEK Microsystems. FPGAs Edge Out GPPs for Advanced Signal Processing Apps. Retrieved October 21, 2011, from COTS Journal Web site: http://www.cotsjournalonline.com/articles/print_article/100527

[3] 1.1 Host and Function IP core :: Overview. (2010, February 22). Retrieved June 25, 2011, from OpenCores Web Site: http://opencores.org/project,usbhostslave

[4] Aeste Works (M) Sdn Bhd. (n.d.). aemb | AESTE. Retrieved December 5, 2011, from AESTE | engineering elegance: http://www.aeste.my/tags/aemb

[5] Nakano, K. and Ito, Y. "Processor, Assembler, and Compiler Design Education Using an FPGA," 2008. Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on. pp. 723-728.

[6] A. Burutarchanai, P. Nanthanavoot, C. Aporntewan, and P. Chongstitvatana, "A stack-based processor for resource efficient embedded systems," Proc. of IEEE TENCON 2004, Thailand, 21-24 November 2004.

[7] Koopman, Philip J. Stack Computers: the new wave: Mountain View Press, 1989.

[8] Wishbone :: OpenCores. Retrieved October 21, 2011, from OpenCores Web site: http://opencores.org/opencores,wishbone

[9] CAST, Inc. (n.d.). USBHS-OTG-SD. Retrieved December 5, 2011, from Semiconductor IP Cores and Electronic Platform IP from CAST, Inc.: http://www.cast-inc.com/ip-cores/interfaces/usbhs-otg-sd/index.html

[10] HiTech Global, LLC. (n.d.). USB 2.0 Device IP Core. Retrieved December 5, 2011, from IP cores, FPGA Evaluation boards and desing services: http://www.hitechglobal.com/IPCores/usbdevice.htm

[11] Compaq, Intel, Microsoft,NEC. Universal Serial Bus Specification. Revision 1.1. s.l. : Compaq Computer Corporation,Intel Corporation, Microsoft Corporation, NEC Corporation, 23 September 1998.