Gisselquist
Technology, LLC

OPENARTY SPECIFICATION

Dan Gisselquist, Ph.D.
dgisselq (at) opencores.org

September 18, 2016

Revision History

Rev.	Date	Author	Description
0.0	6/20/2016	Gisselquist	First Draft

Contents

Figures

Figure Page

Tables

Preface

Dan Gisselquist, Ph.D.

1.

Introduction

The goals of this project include:

1. Use entirely open interfaces

   This means not using the Memory Interface Generator (MIG), the Xilinx CoreGen IP, etc. Further, I wish to use all of Arty's on–board hardware: Flash, DDR3-SDRAM, Ethernet, and everything else at their full and fastest speed(s). For example, the flash will need to be clocked at 82 MHz, not the 50 MHz I've clocked it at before. The memory should also be able to support pipelined 32–bit interactions over the Wishbone bus at a 162 MHz clock. Finally, the Ethernet controller should be supported by a DMA capable interface that can drive the ethernet at its full 100Mbps rate.

2. Run using a 162.5 MHz clock, if for no other reason than to gain the experience of building logic that can run that fast.[1]

3. Modify the ZipCPU to support an MMU and a data cache, and perhaps even a floating point unit.

4. The default configuration will also include three Pmods: a USBUART, an SDCard, and the GPS Pmod.

I intend to demonstrate this project with a couple programs:

1. A very simple program that runs automatically upon startup that can be used to select from among multiple configurations.

2. NTP Server

3. A ZipOS that can actually load and run programs from the SD Card

[1] The original goal was to run at 200 MHz. However, the memory controller cannot run faster than 83 MHz. If we run it at 81.25 MHz and double that clock to get our logic clock, that now places us at 162.5 MHz. 200 MHz is . . . too fast for DDR3 transfers using the Artix–7 chip on the Arty.

2.

Architecture

3.

Software

3.1 Directory Structure

3.2 Zip CPU Tool Chain

3.3 Bench Test Software

3.4 Host Software

- `readflash`: As I am loathe to remove anything from a device that came factory installed, the `readflash` program reads the original installed configuration from the flash and dumps it to a file.
- `wbregs`: This program offers a capability very similar to the PEEK and POKE capability Apple user's may remember from before the days of Macintosh. `wbregs <address>` will read from the Wishbone bus the value at the given address. Likewise `wbregs <address> <value>` will write the given value into the given address. While both address and value have the semantics of numbers acceptable to `strtoul()`, the address can also be a named address. Supported names can be found in `regdefs.cpp`, and their register mapping in `regdefs.h`.
- `ziprun`:
- `zipload`:

3.5 Zip CPU Programs

- `ntpserver`:
- `goldenstart`:

3.6 ZipOS

3.6.1 System Calls

- `int wait(unsigned event_mask, int timeout)`

- `int clear(unsigned event_mask, int timeout)`
- `void post(unsigned event_mask)`
- `void yield(void)`
- `int read(int fid, void *buf, int len)`
- `int write(int fid, void *buf, int len)`
- `unsigned time(void)`
- `void *malloc(void)`
- `void free(void *buf)`

3.6.2 Scheduler

4.

Operation

5.

Registers

There are several address regions on the S6 SoC, as shown in Tbl. 5.1.

Binary Address	Base	Size(W)	Purpose
0000 0000 0000 0000 0001 000x xxxx	0x00000100	32	Peripheral I/O Control
0000 0000 0000 0000 0001 0010 0yyx	0x00000120	8	Debug scope control
0000 0000 0000 0000 0001 0010 10xx	0x00000128	4	RTC control
0000 0000 0000 0000 0001 0010 11xx	0x0000012c	4	SDCard controller
0000 0000 0000 0000 0001 0011 00xx	0x00000130	4	GPS Clock loop control
0000 0000 0000 0000 0001 0011 01xx	0x00000134	4	OLEDrgb control
0000 0000 0000 0000 0001 0011 1xxx	0x00000138	8	Network packet interface
0000 0000 0000 0000 0001 0100 0xxx	0x00000140	8	GPS Testbench
0000 0000 0000 0000 0001 0100 1xxx	0x00000148	8	*Unused*
0000 0000 0000 0000 0001 0101 xxxx	0x00000150	16	*Unused*
0000 0000 0000 0000 0001 011x xxxx	0x00000160	32	*Unused*
0000 0000 0000 0000 0001 100x xxxx	0x00000180	32	*Unused*
0000 0000 0000 0000 0001 101x xxxx	0x000001a0	32	Ethernet configuration registers
0000 0000 0000 0000 0001 110x xxxx	0x000001c0	32	Extended Flash Control Port
0000 0000 0000 0000 0001 111x xxxx	0x000001e0	32	ICAPE2 Configuration Port
0000 0000 0000 0000 10xx xxxx xxxx	0x00000800	1k	Ethernet TX Buffer
0000 0000 0000 0000 11xx xxxx xxxx	0x00000c00	1k	Ethernet RX Buffer
0000 0000 0000 1xxx xxxx xxxx xxxx	0x00008000	32k	On-chip Block RAM
0000 01xx xxxx xxxx xxxx xxxx xxxx	0x00400000	4M	QuadSPI Flash
0100 1000 0000 0000 0000 0000 0000	0x0400000		Configuration Start
0100 1000 0000 0000 0000 0000 0000	0x0480000		CPU Reset Address
01xx xxxx xxxx xxxx xxxx xxxx xxxx	0x04000000	64M	DDR3 SDRAM
1000 0000 0000 0000 0000 0000 000x	0x08000000	2	ZipCPU debug control port—only visible to debug WB master

Table 5.1: Address Regions

5.1 Peripheral I/O Control

Tbl. 5.3 shows the addresses of various I/O peripherals included as part of the SoC. We'll walk

Base	Size(W)	Purpose
0x0c0000000	1	Primary Zip PIC
0x0c0000001	1	Watchdog Timer
0x0c0000002	1	Bus Watchdog Timer
0x0c0000003	1	Alternate Zip PIC
0x0c0000004	1	ZipTimer-A
0x0c0000005	1	ZipTimer-B
0x0c0000006	1	ZipTimer-C
0x0c0000007	1	ZipJiffies
0x0c0000008	1	Master task counter
0x0c0000009	1	Master prefetch stall counter
0x0c000000a	1	Master memory stall counter
0x0c000000b	1	Master instruction counter
0x0c000000c	1	User task counter
0x0c000000d	1	User prefetch stall counter
0x0c000000e	1	User memory stall counter
0x0c000000f	1	User instruction counter
0x0c0000010	1	DMA command register
0x0c0000011	1	DMA length
0x0c0000012	1	DMA source address
0x0c0000013	1	DMA destination address
0x0c0000040	1	*Reserved for MMU context register*
0x0c0000080	32	*Reserved for MMU TLB*

Table 5.2: ZipSystem Addresses

Name	Address	Width	Access	Description
VERSION	0x0100	32	R	Build date
PIC	0x0101	32	R/W	Bus Interrupt Controller
BUSERR	0x0102	32	R	Last Bus Error Address
PWRCOUNT	0x0103	32	R	Ticks since startup
BTNSW	0x0104	32	R/W	Button/Switch controller
LEDCTRL	0x0105	32	R/W	LED Controller
AUXSETUP	0x0106	29	R/W	Auxilliary UART config
GPSSETUP	0x0107	29	R/W	GPS UART config
CLR-LEDx	0x0108-b	32	R/W	Color LED controller
RTCDATE	0x010c	32	R/W	BCD Calendar Date
GPIO	0x010d	32	R/W	GPIO controller
UARTRX	0x010e	32	R/W	Aux UART receive byte
UARTTX	0x010f	32	R/W	Aux UART transmit byte
GPSRX	0x0110	32	R/W	GPS UART receive byte
GPSTX	0x0111	32	R/W	GPS UART transmit byte

Table 5.3: I/O Peripheral Registers

through each of these peripherals in turn, describing how they work.

5.1.1 Interrupt Controller

The OpenArty design maintains three interrupt controllers. Two of them are found within the ZipSystem, and the third is located on the bus itself. Of these, the primary interrupt controller is located in the ZipSystem. This interrupt controller accepts, as interrupt inputs, the outputs of both the auxilliary interrupt controller as well as the bus interrupt controller. Hence, even though the CPU only supports a single interrupt line, by using these three interrupt controllers many more interrupts can be supported.

The primary interrupt controller handles interrupts from the sources listed in Tbl. 5.4. These interrupts are listed together with the mask that would need to be used when referencing them to the interrupt controller. In a similar fashion, the auxilliary interrupt controller accepts inputs from the sources listed in Tbl. 5.5. Finally, the bus interrupt controller handles the interrupts from the sources listed in Tbl. 5.6.

Name	Bit Mask	DMAC ID	Description
SYS_DMAC	0x0001		The DMA controller is idle.
SYS_JIF	0x0002	1	A Jiffies timer has expired.
SYS_TMC	0x0004	2	Timer C has timed out.
SYS_TMB	0x0008	3	Timer C has timed out.
SYS_TMA	0x0010	4	Timer C has timed out.
SYS_AUX	0x0020	5	The auxilliary interrupt controller sends an interrupt
SYS_EXT	0x0040	6	A Bus interrupt has tripped.
SYS_PPS	0x0080	7	An interrupt marking the top of the second
SYS_GPSRX	0x0100	8	A character has been received via GPS
SYS_NETRX	0x0200	9	A packet has been received via the network
SYS_NETTX	0x0400	10	The network controller is idle, having sent its last packet
SYS_UARTRX	0x0800	11	A character has been received via the UART
SYS_UARTTX	0x1000	12	The transmit UART is idle, and ready for its next character.
SYS_SDCARD	0x2000	13	The SD-Card controller has become idle
SYS_BUTTON	0x4000	14	A Button has been pressed.

Table 5.4: Primary System Interrupts

Name	Bit Mask	DMAC ID	Description
AUX_UIC	0x0001	16	The user instruction counter has overflowed.
AUX_UPC	0x0002	17	The user prefetch stall counter has overflowed.
AUX_UOC	0x0004	18	The user ops stall counter has overflowed.
AUX_UTC	0x0008	19	The user clock tick counter has overflowed.
AUX_MIC	0x0010	20	The supervisor instruction counter has overflowed.
AUX_MPC	0x0020	21	The supervisor prefetch stall counter has overflowed.
AUX_MOC	0x0040	22	The supervisor ops stall counter has overflowed.
AUX_MTC	0x0080	23	The supervisor clock tick counter has overflowed.
AUX_SWITCH	0x0100	24	A switch has changed state
AUX_FLASH	0x0200	25	The flash controller has completed a write/erase cycle
AUX_SCOPE	0x0400	26	The Scope has completed its collection
AUX_RTC	0x0800	27	An alarm or timer has taken place (assuming the RTC is installed, and includes both alarm or timer)
AUX_GPIO	0x1000	28	The GPIO input lines have changed values.
AUX_OLED	0x2000	29	The OLED driver is idle

Table 5.5: Auxilliary System Interrupts

Name	Bit Mask	Description
BUS_BUTTON	0x0001	A Button has been pressed.
BUS_SWITCH	0x0002	The Scope has completed its collection
BUS_PPS	0x0004	Top of the second
BUS_RTC	0x0008	An alarm or timer has taken place (assuming the RTC is installed, and includes both alarm or timer)
BUS_NETRX	0x0010	A packet has been received via the network
BUS_NETTX	0x0020	The network controller is idle, having sent its last packet
BUS_UARTRX	0x0040	A character has been received via the UART
BUS_UARTTX	0x0080	The transmit UART is idle, and ready for its next character.
BUS_GPIO	0x0100	The GPIO input lines have changed values.
BUS_FLASH	0x0200	The flash device has finished either its erase or write cycle, and is ready for its next command. (Alternate config only.)
BUS_SCOPE	0x0400	A scope has completed collecting.
BUS_GPSRX	0x0800	A character has been received via GPS
BUS_SDCARD	0x1000	The SD-Card controller has become idle
BUS_OLED	0x2000	The OLED interface has become idle
BUS_ZIP	0x4000	True if the ZipCPU has come to a halt

Table 5.6: Bus Interrupts

5.1.2 Last Bus Error Address

5.1.3 General Purpose I/O

5.1.4 UART Data Register

5.2 Debugging Scopes

5.3 Internal Configuration Access Port

5.4 Real–Time Clock

5.5 On-Chip Block RAM

5.6 Flash Memory

Name	Address	Width	Access	Description
ewreg	0x0180	32	R	Erase/write control and status
status	0x0181	8	R/W	Bus Interrupt Controller
nvconf	0x0182	16	R	Last Bus Error Address
vconf	0x0183	8	R	Ticks since startup
evonc	0x0184	8	R/W	Button/Switch controller
lock	0x0185	8	R/W	LED Controller
flagstatus	0x0186	8	R/W	Auxilliary UART config
clear	0x0187	8	R/W	Clear status on write
Device ID	0x0188--0x018c	5x32	R	Device ID
asyncOTP	0x18e	32	W	Asynch Read OTP. Write starts the ASynch read, 0xff returned until complete
OTP	0x0190--0x19f	16x32	R/W	OTP Memory

Table 5.7: Flash control registers

6.

Wishbone Datasheet

The master and slave interfaces have been simplified with the following requirement: the `STB` line is not allowed to be high unless the `CYC` line is high. In this fashion, a slave may often be able to ignore `CYC` and only act on the presence of `STB`, knowing that `CYC` must be active at the same time.

7.

Clocks

Name	Source	Rates (MHz)		Description
		Max	Min	
i_clk_100mhz	Ext	100		100 MHz Crystal Oscillator
Future s_clk	PLL	152	166	Internal Logic, Wishbone Clock
s_clk	PLL	83.33	75.76	DDR3 SDRAM Controller Clock
mem_clk_200mhz		200 MHz		MIG Reference clock for PHASERs
ddr3_ck_*x*	DDR	166.67	303	DDR3 Command Clock
o_qspi_sck	DDR	95		QSPI Flash clock
o_sd_clk	Logic	50	0.100	SD–Card clock
o_oled_sck	Logic	166		OLED SPI clock
o_eth_mdclk	Logic	25	2.5	Ethernet MDIO controller clock

Table 7.1: OpenArty clocks

8.

I/O Ports

Table. 8.1 lists the various I/O ports associated with OpenArty.

Port	Width	Direction	Description
i_clk_100mhz	1	Input	Clock
o_qspi_cs_n	1	Output	Quad SPI Flash chip select
o_qspi_sck	1	Output	Quad SPI Flash clock
io_qspi_dat	4	Input/Output	Four-wire SPI flash data bus
i_btn	4	Input	Inputs from the two on-board push-buttons
i_sw	4	Input	Inputs from the two on-board push-buttons
o_led	4	Output	Outputs controlling the four on-board LED's
o_clr_led0	3	Output	
o_clr_led1	3	Output	
o_clr_led2	3	Output	
o_clr_led3	3	Output	
i_uart_rx	1	Input	UART receive input
o_uart_tx	1	Output	UART transmit output
i_aux_rx	1	Input	Auxiliary/Pmod UART receive input
o_aux_tx	1	Output	Auxiliary/Pmod UART transmit output
i_aux_rts	1	Input	Auxiliary/Pmod UART receive input
o_aux_cts	1	Output	Auxiliary/Pmod UART transmit output
i_gps_rx	1	Input	GPS/Pmod UART receive input
o_gps_tx	1	Output	GPS/Pmod UART transmit output
i_gps_pps	1	Input	GPS Part-per-second (PPS) signal
i_gps_3df	1	Input	GPS
o_oled_cs_n	1	Output	
o_oled_sck	1	Output	
o_oled_mosi	1	Output	
i_oled_miso	1	Input	
o_oled_reset	1	Output	
o_oled_dc	1	Output	
o_oled_en	1	Output	
o_oled_pmen	1	Output	
o_sd_sck	1	Output	SD Clock
i_sd_cd	1	Input	Card Detect
i_sd_wp	1	Input	Write Protect
io_cmd	1	In/Output	SD Bi-directional command wire
io_sd	4	In/Output	SD Bi-directional data lines
o_cls_cs_n	1	Output	CLS Display chip select
o_cls_sck	1	Output	CLS Display clock
o_cls_mosi	1	Output	CLS Display MOSI
i_cls_miso	1	Input	CLS Display MISO

Table 8.1: List of IO ports