STORM SoC – System on Chip
by Stephan Nolting

System Components

Proprietary Notice

Table of content

1. Introduction

The STORM SoC provides a complete system on chip, which is build around the STORM Core Processor. This data sheet provides information about the used system components.

Most of the components were created by myself, some additional controller were downloaded from opencores.org. These cores feature an additional data sheet (component/doc folder).

2. System Components

- Not all of the listed modules are used in a specific implementation
- Most of the components, which are connected to the Wishbone bus, can be byte-accessed, even if this feature is not relevant within the STORM SoC due to the static 32-bit interface
- All modules are Wishbone compatible and use therefore the same signal namings for the bus interface. These bus interface signals are not listed in the component's description, only special (non-Wishbone) signals are mentioned.
- Most components, which where not created by me, feature an additional data sheet. See the component's doc folder.
- IO devices are automatically protected and can only be accessed in privileged modes.
- Due to the fabric-internal address-comparator and -mappings, the register order of the relative memory address map is not always equal to the register order in the system memory map. Use only the addresses / IO locations specified in the system memory map.

2.1 STORM Core Processor

Author: Stephan Nolting
File name of component's top entity: STORM_TOP.vhd

The STORM Core Processor is the heart of every implementation of the STORM SoC. It's sources are not included within the SoC package and must be downloaded separately at:
http://www.opencores.com/project,storm_core.

All IO/Memory transactions are controlled by the STORM Core and only by the STORM Core – there is no DMA implemented. It is equipped with separated, full-associative cache units for instructions and data. A Wishbone compatible bus unit connects these units to rest of the system.

The core itself is compatible to ARM's famous RISC controller family (→ ARM7 & ARM9). The opcodes, the functionality as well as the programmer's model are nearly eqaul to the ARM.

See the STORM Core data sheet in the core's doc folder for more information.

Configuration

Generic	Generic type	Function
I_CACHE_PAGES	natural	Number of pages in I-Cache
I_CACHE_PAGE_SIZE	natural	I-Cache page size (# of 32-bit words)
D_CACHE_PAGES	natural	Number of pages in D-Cache
D_CACHE_PAGE_SIZE	natural	D-Cache page size (# of 32-bit words)
TIME_OUT_VAL	natural	Maximum Wishbone bus cycle length
BOOT_VECTOR	std_logic_vector(31:0)	Boot vector address
IO_UC_BEGIN	std_logic_vector(31:0)	First address of not cache-able IO area
IO_UC_END	std_logic_vector(31:0)	Last address of not cache-able IO are

Interface

Port signal	Signal size	Direction	Function
CORE_CLK_I	1 bit	Input	Core clock signal, triggering on rising edge
RST_I	1 bit	Input	System rest, high-active, sync to rising edge of core clock
IO_PORT_O	16 bit	Output	Direct system output port
IO_PORT_I	16 bit	Input	Direct system input port
WB_ADR_O	32 bit	Output	Wishbone bus address, word-boundary → bits[1..0] = “00”
WB_CTI_O	3 bit	Output	Wishbone bus cycle type
WB_TGD_O	7 bit	Output	Wishbone bus cycle tag
WB_SEL_O	4 bit	Output	Wishbone bus byte select, always set to “1111”
WB_WE_O	1 bit	Output	Wishbone bus write enable
WB_DATA_O	32 bit	Output	Wishbone bus data output
WB_DATA_I	32 bit	Input	Wishbone bus data input
WB_STB_O	1 bit	Output	Wishbone bus valid transfer
WB_CYC_O	1 bit	Output	Wishbone bus valid cycle
WB_ACK_I	1 bit	Input	Wishbone bus acknowledge signal
WB_ERR_I	1 bit	Input	Wishbone bus abnormal cycle termination
WB_HALT_I	1 bit	Input	Wishbone bus halt
IRQ_I	1 bit	Input	Interrupt request
FIQ_I	1 bit	Input	Fast interrupt request

See the Wishbone specification in the core's doc folder for more information.

2.2 Internal SRAM

Author: Stephan Nolting
File name of component's top entity: MEMORY.vhd

This memory component is the basic module for the internal data/program memory. Set the OUTPUT_GATE generic “true”, if you are using an or-based Wishbone data read-back. Enabling this feature might cause problems for the synthesis tool to map the memory to dedicated memory components.
The memory can be initialized with a program code or data segment when needed. Place the program/data in the “INIT MEMORY IMAGE” labeled signal initialization (this feature should only be used for simulation / debugging).

Interface / Configuration

Generic	Generic type	Function
MEM_SIZE	natural	Memory size in cells (=32 bit entries)
LOG2_MEM_SIZE	natural	Log2 of memory size (= log2(MEM_SIZE))
OUTPUT_GATE	boolean	Use and-gates for data output

Relative address map

Relative address	Area (byte)	R/W	Function
0x00000000 ... MEM_SIZE-4	MEM_SIZE*4	R/W	Free-to-use memory cells, each 4 bytes wide

2.3 Boot ROM

Author: Stephan Nolting
File name of component's top entity: BOOT_ROM_FILE.vhd

This memory component is the basic module for the internal boot rom memory. Set the OUTPUT_GATE generic “true”, if you are using an or-based Wishbone data read-back. Enabling this feature might cause problems for the synthesis tool to map the memory to dedicated memory components.
The memory can be initialized with a program code or data segment when needed, pre-installed bootloaders for several development boards can be selected via the INIT_IMAGE_ID. Ensure, the STORM Core is booting up from the base address of the boot ROM.

Interface / Configuration

Generic	Generic type	Function
MEM_SIZE	natural	Memory size in cells (=32 bit entries)
LOG2_MEM_SIZE	natural	Log2 of memory size (= log2(MEM_SIZE))
OUTPUT_GATE	boolean	Use and-gates for data output
INIT_IMAGE_ID	string	Name of initialization image

Relative address map

Relative address	R/W	Function
0x00000000 ... MEM_SIZE-4	R	Free-to-use memory cells, each 4 bytes wide

2.4 IO Controller

Author: Stephan Nolting
File name of component's top entity: GP_IO_CTRL.vhd

The IO controller provides a simple IO port. 32 bits are used as outputs, another 32 are used as inputs.
It can be used for directly controlling FPGA pins or for internal system control functions.
The IRQ output will go high for one clock cycle when the status of the inputs has changed.

Interface / Configuration

Special signal	Signal type	Function
GP_IO_O	std_logic_vector(31:0)	Parallel output port
GP_IO_I	std_logic_vector(31:0)	Parallel input
IO_IRQ_O	std_logic	Input status change interrupt

Relative address map

Relative address	R/W	Function
0x00000000	R/W	Output port register
0x00000004	R/W	Input port register

2.5 Seven Segment Controller

Author: Stephan Nolting
File name of component's top entity: SEVEN_SEG_CTRL.vhd

Up to four (d = 0..3) high or low-active seven segment display can be controlled with this module. To display hexadecimal numbers, the corresponding value can be written to the DATA register (offset = 0). To display other symbols, you can write to the CTRL register (offset = 4) to directly control the display segments. Hex-coded DATA will be decoded and the decoded value is automatically written to the CTRL register.

Display control lines (d is the led-display index):

```
A-segment(d) <= HEX_O(d+0)
B-segment(d) <= HEX_O(d+1)
C-segment(d) <= HEX_O(d+2)
D-segment(d) <= HEX_O(d+3)
E-segment(d) <= HEX_O(d+4)
F-segment(d) <= HEX_O(d+5)
G-segment(d) <= HEX_O(d+6)
```

Display segments:

```
 AAAAA
F     B
F     B
F     B
 GGGGG
E     C
E     C
E     C
 DDDDD
```

```
display(d) <= DATA_REGISTER(d*4+3 downto d*4+0)
display(d) <= CTRL_REGISTER(d*7+6 downto d*7+0)
display(d) <= HEX_O(d*7+6 downto d*7+0)
```

Interface / Configuration

Generic	Generic type	Function
HIGH_ACTIVE_OUTPUT	boolean	Connected LEDs are high-active

Special signal	Signal type	Function
HEX_O	std_logic_vector(27:0)	Control signals for 4 seven segment displays

Relative address map

Relative address	R/W	Function
0x00000000	R/W	Hex DATA register
0x00000004	R/W	Segment CTRL register

2.6 Timer

Author: Stephan Nolting
File name of component's top entity: TIMER.vhd

This timer can be used for any timing application. Whenever the counter reg value reached the threshold value, an interrupt (when enabled) can be generated. The counter can then be automatically reset if bit 1 of the control register is set. A prescaler value different from 0 will prescale the frequency of the counter increment. The interrupt output signal will become high for one clock cycle, when the counter register value reached the threshold value (and interrupt enable bit is set).

Interface / Configuration

Special signal	Signal type	Function
INT_O	std_logic	Threshold interrupt, one clock cycle high

Relative address map

Relative address	R/W	Function
0x00000000	R/W	Counter register
0x00000004	R/W	Threshold value register
0x00000008	R/W	Configuration register
0x0000000C	R/W	Scratch register

Configuration Register

Bit(s)	R/W	Function
31..16	R/W	Prescaler value
15..3	R/W	*unused*
2	R/W	Interrupt enable
1	R/W	Auto reset after threshold reached
0	R/W	Timer enable

2.7 Vector Interrupt Controller

Author: Stephan Nolting
File name of component's top entity: VIC.vhd

The vectorized interrupt controller is mostly compatible to the one used e.g. in LPC ARM controller.
Up to 16 interrupt sources can be configured using vectorized interrupt service routine addresses. Another 16 interrupt request lines can be mapped to one interrupt service routine. See the data sheet of an LPC ARM controller for more information about the VIC.

Relative address map

Relative address	R/W	Function
0x00000000	R	Masked IRQ request status
0x00000004	R	Masked FIQ request status
0x00000008	R	Unmasked interrupt requests
0x0000000C	R/W	Interrupt lines type select, '0' = IRQ, '1' = FIQ
0x00000010	R/W	Interrupt request lines enable
0x00000014	W	Clear interrupt request line enable
0x00000018	W	Trigger interrupt request line by software
0x0000001C	W	Clear software interrupt request line
0x00000020	R/W	Bit 0: Protected mode enable → access only in privileged modes possible
0x00000030	R/W	Interrupt service routine address / interrupt acknowledge
0x00000034	R/W	Interrupt service routine address for unvectorized interupts
0x00000038	R/W	High level / rising edge ('0') or low level / falling edge trigger
0x0000003C	R/W	Level triggered ('0') or edge triggered ('1') interrupt
0x00000040 ... 0x0000007C	R/W	Interrupt service routine address for corresponding interrupt channel
0x00000080 ... 0x000000BC	R/W	Source select (bits 4:0) and enable (bit 5) for corresponding interrupt channel

2.8 Mini UART

Author: Philippe Carton (opencores)
Modified by: Stephan Nolting
File name of component's top entity: INT_CTRL.vhd

This is a simple UART, created by Philippe Carton. It is capable of transmitting, receiving data via the RS232 port. The system was modified to support 32-bit Wishbone bus access.
Since it does not support any handshake or FIFO system, it is very suitable for simple operations.

Interface / Configuration

Generic	Generic type	Function
BRDIVISON	integer	Baud divider value = f_clk/(4*baudrate)

Special signal	Signal type	Function
IntTx_O	std_logic	Waiting for byte interrupt
IntRx_O	std_logic	Byte received interrupt
BR_CLK_I	std_logic	Clock used for baud generator
TxD_PAD_O	std_logic	Transmitter output
RxD_PAD_I	std_logic	Receiver input

Relative address map

Relative address	R/W	Function
0x00000000	R/W	UART data register Bits [7:0]: Received / transmitted character
0x00000004	R	UART status register Bit 0: Ready to send when '1' Bit 1: Byte received when '1'

See the data sheet in the component's doc folder for more information.

2.9 SPI Controller

Author: Simon Srot (opencores)
File name of component's top entity: spi_top.v

This SPI controller provides a high-speed SPI port with 8 low-active slave select signals.

Relative address map

Relative address	R/W	Function
0x00000000	R/W	Data receive / transmit register 0
0x00000004	R/W	Data receive / transmit register 1
0x00000008	R/W	Data receive / transmit register 2
0x0000000C	R/W	Data receive / transmit register 3
0x00000010	R/W	Control and status register
0x00000014	R/W	Clock divider register
0x00000018	R/W	Slave select register

See the data sheet in the component's doc folder for more information.

2.10 I²C Controller

Author: Richard Herveille (opencores)
File name of component's top entity: i2c_master_top.vhd

The I2C controller implements an interface for on-board communication via I²C.
This device was originally created for an eight bit bus system, so only the lowest 8 bits of the data bus are relevant, all other bits are read as zero. Writing data on bits 31 downto 8 does not perform any operation.

Relative address map

Relative address	R/W	Function
0x00000000	R/W	Clock divider register, low byte
0x00000004	R/W	Clock divider register, high byte
0x00000008	R/W	Control register
0x0000000C	R/W	Receive / transmit data register
0x00000010	R/W	Command / status register

See the data sheet in the component's doc folder for more information.

2.11 PS-2 Keyboard Controller

Author: Daniel Quinter (opencores)
File name of component's top entity: ps2_wb.vhd

Via this controller, a ps-2 compatible keyboard can be connected with the STORM SoC.

Relative address map

Relative address	R/W	Function
0x00000000	R/W	Receive / transmit data register
0x00000004	R/W	Status / control register

2.12 External Memory Controller

Author: Rudolf Usselman (opencores)
File name of component's top entity: mc_top.v

This extreme powerful controller provides dedicated access to external memory devices.

Relative address map

Relative address	R/W	Function
0x00000000	R/W	Main configuration register
0x00000004	R	Power-On configuration register
0x00000008	R/W	Base address mask register
0x00000010 0x00000014 . . . 0x00000048 0x0000004C	R/W	Chip select configuration register 0 Timing specification register 0 ... Chip select configuration register 7 Timing specification register 7

See the data sheet in the component's doc folder for more information.

2.13 Reset Protector

Author: Stephan Nolting
File name of component's top entity: RST_PROTECT.vhd

The reset protector is not connected to the Wishbone bus system. It is responsible for generating a valid system reset out of external reset requests. Only when a valid external reset is applied to the EXT_RST_I input for at least 3*CLK_SPEED clock cycles, the SYS_RST_O pin is taken high for CLK_SPEED/10000 cycles.

Interface / Configuration

Generic	Generic type	Function
CLK_SPEED	natural	System clock speed in Hz
LOW_ACT_RST	booelan	EXT_RST_I is low active when set to “true”

Special signal	Signal type	Function
MAIN_CLK_I	std_logic	System clock
EXT_RST_I	std_logic	External reset request
SYS_RST_O	std_logic	System reset, high active

2.14 System PLL

Author: Altera Quartus II - Megawizzard
File name of component's top entity: SYSTEM_PLL.vhd

This component was generated by the Altera Megawizzard tool. The phase locked loop is used to create the internal and external clock results, as well as a system reset (only active during warm-up of PLL).

Interface

Special signal	Signal type	Function
inclk0	std_logic	External clock input
c0	std_logic	System clock
c1	std_logic	Memory clock = system clock / 2
c2	std_logic	Memory clock, -3ns phase shifted
locked	std_logic	Clocks are stable when '1'

3. Source Files

All hardware sources can be found in the component's rtl director.
The top entity of the corresponding component is listed first and is highlighted.

STORM Core Processor:
- storm_core/trunk/rtl/**STORM_TOP.vhd**
- storm_core/trunk/rtl/BUS_UNIT.vhd
- storm_core/trunk/rtl/CACHE.vhd
- storm_core/trunk/rtl/CORE_PKG.vhd
- storm_core/trunk/rtl/CORE.vhd
- storm_core/trunk/rtl/OPCODE_DECODER.vhd
- storm_core/trunk/rtl/FLOW_CTRL.vhd
- storm_core/trunk/rtl/MC_SYS.vhd
- storm_core/trunk/rtl/REG_FILE.vhd
- storm_core/trunk/rtl/OPERAND_UNIT.vhd
- storm_core/trunk/rtl/MS_UNIT.vhd
- storm_core/trunk/rtl/MULTIPLY_UNIT.vhd
- storm_core/trunk/rtl/BARREL_SHIFTER.vhd
- storm_core/trunk/rtl/ALU.vhd
- storm_core/trunk/rtl/LOAD_STORE_UNIT.vhd
- storm_core/trunk/rtl/WB_UNIT.vhd

Boot ROM:
- storm_soc/trunk/components/boot_rom/rtl/**BOOT_ROM_FILE.vhd**

I²C controller:
- storm_soc/trunk/components/i2ccontroller/rtl/vhdl/**i2c_master_top.vhd**
- storm_soc/trunk/components/i2ccontroller/rtl/vhdl/i2c_master_byte_ctrl.vhd
- storm_soc/trunk/components/i2ccontroller/rtl/vhdl/i2c_master_bit_ctrl.vhd

IO controller:
- storm_soc/trunk/components/io controller/rtl/**GP_IO_CTRL.vhd**

Memory controller:
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/**mc_top.v**
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_adr_sel.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_cs_rf.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_defines.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_dp.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_incn_r.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_mem_if.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_obct.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_obct_top.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_rd_fifo.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_refresh.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_rf.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_timing.v
- storm_soc/trunk/components/mem_ctrl/rtl/verilog/mc_wb_if.v

miniUART:
- storm_soc/trunk/components/miniuart/rtl/vhdl/**MINI_UART.vhd**
- storm_soc/trunk/components/miniuart/rtl/vhdl/rxunit.vhd
- storm_soc/trunk/components/miniuart/rtl/vhdl/txunit.vhd
- storm_soc/trunk/components/miniuart/rtl/vhdl/utils.vhd

PS-2 controller:
- storm_soc/trunk/components/ps2core/rtl/vhdl/**ps2_wb.vhd**
- storm_soc/trunk/components/ps2core/rtl/vhdl/ps2.vhd

Reset protector:
- storm_soc/trunk/components/reset protector/rtl/**RST_PROTECT.vhd**

PS-2 Seven segment controller:
- storm_soc/trunk/components/seven segment controller/rtl/**SEVEN_SEG_CTRL.vhd**

SPI controller:
- storm_soc/trunk/components/spi controller/rtl/verilog/**spi_top.v**
- storm_soc/trunk/components/spi controller/rtl/verilog/spi_clgen.v
- storm_soc/trunk/components/spi controller/rtl/verilog/spi_defines.v
- storm_soc/trunk/components/spi controller/rtl/verilog/spi_shift.v
- storm_soc/trunk/components/spi controller/rtl/verilog/timescale.v

SPI controller:
- storm_soc/trunk/components/sram memory/rtl/**MEMORY.vhd**

System timer:
- storm_soc/trunk/components/timer/rtl/**TIMER.vhd**

Vectorized interrupt controller:
- storm_soc/trunk/components/vector interrupt controller/rtl/**VIC.vhd**