T48 µController Integration Manual

Author: Arnim Läuger

arniml@opencores.org

Rev 0.3

October 31, 2005

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
0.1	19-Jun-2005	A. Läuger	First Draft
0.2	12-Sep-2005	A. Läuger	Added design hierarchy, memory integration, I/O interfaces and sample systems.
0.3	31-Oct-2005	A. Läuger	Description of Wishbone Master, added index.

Contents

INTRODUCTION.....1

ARCHITECTURE.....2

OPERATION.....4

CLOCKS.....5

PORT LIST.....7

MEMORY INTEGRATION.....9

I/O INTERFACES.....10

1

Introduction

The T48 µController core is an implementation of the MCS-48 microcontroller family architecture. While being a controller core for SoC, it also aims for code-compatability and cycle-accuracy so that it can be used as a drop-in replacement for any MCS-48 controller.

The core can be configured to better suit the requirements and characteristics of the integrating system. On the other hand, nearly the full functionality of a stock 8048/8049 is available. This flexibility is achieved by separating system aspects from the core's functionality. Among others, this includes memory sizes, memory implementation and clock generation.

For reference and to enable quick setup, this core is accompanied by several sample systems. They demonstrate how the configuration features can be utilized to tailor the core to one's needs.

The T48 µController project is maintained at

http://www.opencores.org/projects.cgi/web/t48/overview

Updates of the core can be obtained via the project pages.

2

Architecture

This section describes the architecture of the block. A block diagram should be included describing the top level of the design.

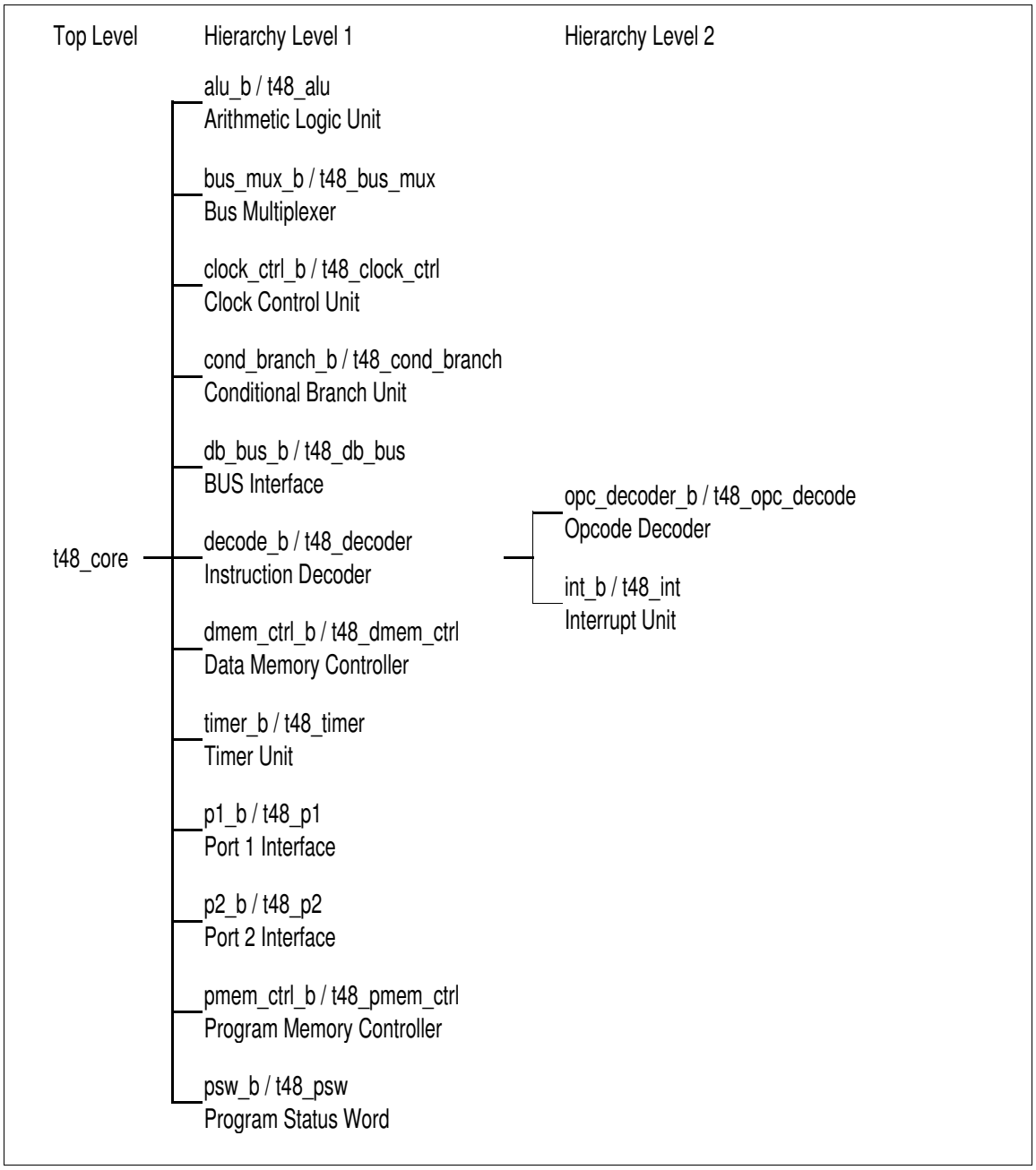

Figure 1: T48 µController Hierarchy

3

Operation

This section describes the operation of the core. Specific sequences, such as startup sequences, as well as the modes and states of the block should be described.

4

Clocks

The T48 µController core operates on one single clock. However, due to the characteristics of the MCS-48 clocking system, this main clock is derived by a clock divider. This section explains the details.

Name	Source	Rates (MHz)		Description
		Max	Min	
xtal_i	Input Pad	11	-	Clock input from external crystal / clock generation circuit.
clk_i	Clock former	f_{xtal_i}	$f_{xtal\_i}/3$	Main system clock. Synchronously enabled by en_clk_i.

Table 1: List of clocks

The clocking system of the MCS-48 family establishes a circuit that divides the incoming external clock on XTAL by 3 to generate the base clock for all system operations. The T48 µController core needs to mimic this scheme because most of the control signals like psen_n_o, rd_n_o etc. are generated with the clock on xtal_i. This is all done inside the clock_ctrl module, therefore it is the only module which requires the external clock on xtal_i.

All other logic of the T48 µController operates with the main system clock applied to clk_i. It is the responsibility of the system to provide a suitable clock waveform at this input. The core supports this task by providing the xtal3_o output which indicates that the next rising edge of xtal_i is the third in a row. So most of the dividing is already prepared inside the core (namely clock_ctrl). What is left to the integrating system is the final clock shaping.

There are two methods to generate the required clock at clk_i:

1. Use clock enable input en_clk_i
   All modules that operate on clk_i also use en_clk_i as a synchronous clock enable. It is therefore possible to apply the external clock (connected to xtal_i) to clk_i as well, while the divider output xtal3_o is connected to en_clk_i.
   This scenario is the more simple one and should work with any FPGA technology.

2. Shape external XTAL clock by clock gating
   In case your technology provides valid clock gating circuitry, you can gate the external XTAL, thus generating a divided clock at clk_i. Use xtal3_o as the clock gate enable signal. As clk_i is already supplied with the required clock, the synchronous clock enable at en_clk_i has to be tied constantly to 1.
   This option is the most elegant one as it will result in reduced area (synchronous clock enables optimized away from each flip-flop) and reduced power consumption (flip-flops are only clocked every third clock). However, dedicated clock gating support from the underlying technology is required to safely gate the incoming clock without glitches.

These two approaches can be further enhanced with other logic. E.g. the Wishbone master will suppress valid impulses of the main clock when it needs to infer wait states during peripheral access.

Here, the paradigm applies as before: The main clock is formed by clk_i plus en_clk_i. With solution 1), the Wishbone master gates xtal3_o before it is applied to en_clk_i. With solution 2), the Wishbone master gates xtal3_o before it is applied to the clock gate. en_clk_i is still tied to 1 in this case.

5

Port List

This section specifies the I/O ports of the T48 µController.

Port	W	Dir	Description
T48 Interface			
xtal_i	1	In	Clock from external crystal/clock generation circuit.
reset_i	1	In	Asynchronous reset input.
t0_i	1	In	Test 0 input.
t0_o	1	Out	Test 0 output (derived clock output).
t0_dir_o	1	Out	Direction selector for T0 pad. 0 ... T0 is operated in input direction 1 ... T0 is in output mode
int_n_i	1	In	Interrupt input. (Active low)
ea_i	1	In	External Access input which forces all program memory fetches to reference external memory.
rd_n_o	1	Out	Output strobe activated during a BUS read. (Active low)
psen_n_o	1	Out	Program Store Enable. This output occurs only during a fetch to external program memory. (Active low)
wr_n_o	1	Out	Output strobe during a BUS write. (Active low) Used as a write strobe to external data memory.
ale_o	1	Out	Address Latch Enable. This signal occurs once during each cycle. The negative edge of ALE strobes address into external data and program memory.
db_i	8	In	Data Bus or general purpose input/output bus. Read while rd_n_o is active, written while wr_n_o active. Contains the 8 low order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of PSEN'. Also contains the address and data during an external RAM data store instruction, under control of ALE, RD' and WR'.
db_o	8	Out	

Port	W	Dir	Description
db_dir_o	1	Out	Direction of DB pads 0 ... DB[7..0] are operated in input direction 1 ... DB[7..0] are in output mode
t1_i	1	In	Test 1 input
p2_i	8	In	8-bit general purpose input/output port.
p2_0	8	Out	P2[3..0] contain the four high order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for 8243.
p2_low_imp_o	1	Out	Low impedance output driver enable for Port 2.
p1_i	8	In	8-bit general purpose input/output port.
p1_o	8	Out	
p1_low_imp_o	1	Out	Low impedance output driver enable for Port 1.
prog_n_o	1	Out	Output strobe for 8243 I/O expander.
Core Interface			
clk_i	1	In	Main core clock.
en_clk_i	1	In	Clock enable.
xtal3_o	1	Out	Indication of third XTAL clock state.
dmem_addr_o	8	Out	Data Memory address.
dmem_we_o	1	Out	Data Memory write enable.
dmem_data_i	8	In	Data Memory data input.
dmem_data_o	8	Out	Data Memory data output.
pmem_addr_o	12	Out	Program Memory address.
pmem_data_i	8	In	Program Memory data input.

Table 2: List of IO ports

6

Memory Integration

The typical configuration of the T48 µController contains one ROM and one RAM module used for the Program Memory and the Data Memory, respectively. Both components have the same characteristics in that they are synchronous memories clocked by the global system clock clk_i. This clock samples the address, data and control inputs of the memories while the output is formed by a combinational read path from the memory array.

Maximum memory sizes are constrained by the architecture of the MCS-48 family. The Data Memory can contain up to 256 bytes and the Program Memory up to 4096 bytes. The minimum size for the Data Memory is 32 bytes, whereas the members of the MCS-48 family contain at least 64 bytes. Implementation of a Program Memory component is optional. It is in the responsibility of the integrator to choose suitable memory sizes.

The T48 µController's interface to the Data Memory consists of the ports dmem_addr_o, dmem_we_o, dmem_data_i and dmem_data_o. The Program Memory is interfaced via ports pmem_addr_o and pmem_data_i. Refer also to Table 2.

Apart from the memories' interface signals, the port ea_i has to be considered for when integrating the Program Memory. Logic controlling ea_i has to implement the following scheme:

1. Port ea_i is set to '1' whenever the Program Memory is disabled globally. This implements the behavior of the MCS-48 EA pin.
2. Port ea_i is set to '1' whenever an access to a Program Memory location is announced by pmem_address_o that is beyond the implemented ROM size. This implements the automatic Program Memory extension of the MCS-48 family.
3. Port ea_i is set to '0' in all other situations.

For more details on generating ea_i control refer to the sample systems that come with the source code release.

7

I/O Interfaces

The MCS-48 family microcontrollers contain three types of I/O interfaces.

1. Two pseudo-bidirectional general purpose I/O ports called Port 1 and Port 2.
2. One bidirectional port called BUS.
3. Two test inputs called T1 and T2.

Each of the bidirectional port is implemented as two unidirectional buses at the T48 µController interface together with a control signal. For BUS, the signal db_dir_o indicates, when set to '1', that all bits of BUS are operated in output mode.

The situation at Port 1 and Port 2 is a bit more complex. MCS-48 controllers implement open-drain type output drivers with pull-up resistors. This behavior can easily be built in FPGA devices with tri-state drivers where the output enable control for each pin is derived from the state of the respective data bit. In addition, a high level is driven actively when the port register is written to by the CPU. This ensures a proper transition from low to high in contrast to loading the parasitic capacitances at the pin with the pull-up resistor. To enable this behavior, dedicated control signals are available that indictate when Port 1 or Port 2 outputs should be driven actively.

Figure 2 shows a sample circuit for a bidirectional implementation of Port 1 and Port 2.

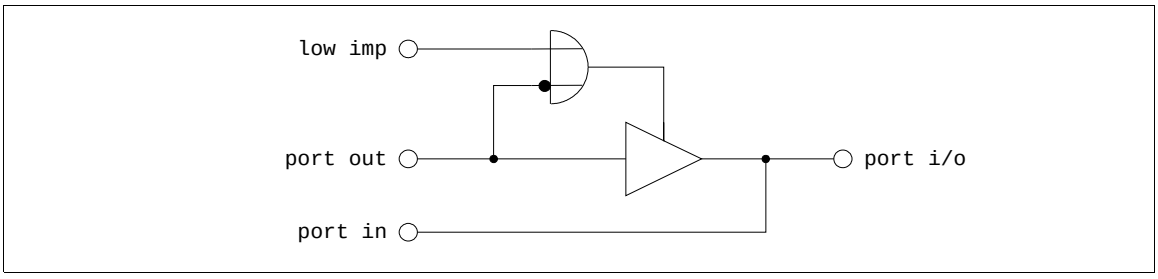

Figure 2: Pseudo-Bidirectional Port Circuit

Appendix A

Sample Systems

Included in the release of the T48 µController project, several sample systems are available. Systems building an MCS-48 compatible chip have a two-level hierarchical structure. The lower level (marked by the “notri” infix) instantiates the T48 µController core and attaches the memories to the core. This level provides the undirectional interface ports towards the system top level. Here, the interfaces are combined to bidirectional buses by tri-state drivers. Chapter I/O Interfaces describes the characteristics of these drivers.

The following sample systems are available:

Name	RAM Size	ROM Size	Remark
t8039	128	None	8039HL-alike top level
t8048	64	1024	8048H-alike top level
t8050_wb	256	4096	8050AH-alike top level with Wishbone Interface

Appendix B

Wishbone Master

The Wishbone master is an optional module that can be attached to the T48 µController core and enables interfacing to Wishbone compatible peripherals. Characteristics are as follows:

- Data bus 8 bit
- Address bus 24 bit
- Standard read/write cycles with wait states

The current implementation of the Wishbone master module requires exclusive access to the BUS interface of the T48 µController. Refer to the t8050_wb sample system for information on how the Wishbone master module is connected to BUS. All MOVX read and write operations generate Wishbone bus cycles at the specified address. This address is built as follows:

Wishbone address = adr2 & adr1 & address of MOXV

Address components adr1 and adr2 are specified via the configuration range of the module. The following Table 1 summarizes the access scheme.

adr_i	MOVX Address	Description	
1	000h	Read/write adr1	Configuration Range
	001h	Read/write adr2	
0	0XXh	Wishbone cycle @ 0XXh	Wishbone Range

Table 3: Wishbone Master Access Matrix

The range selection input adr_i is controlled by P2.4.

Index

Index

8039HL 11
8048H 11
8050AH 11
B
BUS 10, 12
C
Clock
  divider 5
  gating 6
  system 5
E
EA 9
G
general purpose I/O 10
M
Memory
  Data 9
  Program 9
MOVX 12
O
open-drain 10
P
peripherals 12
Ports
  Port 1 10
  Port 2 10
pull-up 10
S
Sample System
  t8039 11
  t8048 11
  t8050_wb 11
SoC 1
T
T1 10
T2 10
tri-state 11
W
Wishbone Master **12**
X
XTAL 5