MIPS_SystemC Reference Manual

Generated by Doxygen 1.2.17

Sat Feb 14 14:54:23 2004

Contents

1 MIPS_SystemC **1**

1.1 Introduction 1

1.2 Compiling 1

1.3 Running 1

2 MIPS_SystemC Hierarchical Index **3**

2.1 MIPS_SystemC Class Hierarchy 3

3 MIPS_SystemC Compound Index **5**

3.1 MIPS_SystemC Compound List 5

4 MIPS_SystemC File Index **7**

4.1 MIPS_SystemC File List 7

5 MIPS_SystemC Class Documentation **9**

5.1 add Struct Reference 9

5.2 alu Struct Reference 10

5.3 and Struct Reference 11

5.4 andgate Struct Reference 12

5.5 control Struct Reference 13

5.6 decode Struct Reference 14

5.7 dmem Class Reference 15

5.8 ext Struct Reference 16

5.9 hazard Struct Reference 17

5.10 imem Class Reference 18

5.11 mips Struct Reference 19

5.12 mux< T > Class Template Reference 21

5.13 orgate Struct Reference 22

5.14 reg_exe_mem_t Struct Reference 23

5.15 reg_id_exe_t Struct Reference 24
5.16 reg_if_id_t Struct Reference 25
5.17 reg_mem_wb_t Struct Reference 26
5.18 regfile Class Reference 27
5.19 registo Struct Reference 28
5.20 regT< T > Class Template Reference 29
5.21 shiftl2 Struct Reference 30

6 MIPS_SystemC File Documentation **31**
6.1 main.cpp File Reference 31
6.2 mipsaux.h File Reference 32

Chapter 1

MIPS_SystemC

1.1 Introduction

`MIPS_SystemC` is a MIPS simulator that uses SystemC to model a MIPS pipelined architecture. The architecture is identical to the one presented in "Computer Organization and Design" by Hennessy&Patterson. `MIPS_SystemC` has a graphical user interface that allows control of program execution and displays some of the signal values inside the architecture.

1.2 Compiling

To compile `MIPS_SystemC` you need SystemC 2.0.1 and Qt version 2.x installed in your system.

You must also define the environment variables `QTDIR` and `SYSTEMC`. The correct value of these variables depend on the instalation of Qt and SystemC. As an example, `QTDIR` and `SYSTEMC` may be initialised, using the bash shell, by issuing the following commands:

```
export QTDIR=/usr/local/qt2
```

```
export SYSTEMC=/usr/local/systemc-2.0.1
```

If those libraries are installed run `make`.

1.3 Running

To run `MIPS_SystemC` execute:

```
MIPS_SystemC
```

The initial contents of instruction memory and data memory can be changed by editing files `instmem.dat` and `datamem.dat`, respectively. These files are formatted as lists of hexadecimal numbers.

Chapter 2

MIPS_SystemC Hierarchical Index

2.1 MIPS_SystemC Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

add . . . 9
alu . . . 10
and . . . 11
andgate . . . 12
control . . . 13
decode . . . 14
dmem . . . 15
ext . . . 16
hazard . . . 17
imem . . . 18
mips . . . 19
mux< T > . . . 21
orgate . . . 22
reg_exe_mem_t . . . 23
reg_id_exe_t . . . 24
reg_if_id_t . . . 25
reg_mem_wb_t . . . 26
regfile . . . 27
registo . . . 28
regT< T > . . . 29
shiftl2 . . . 30

Chapter 3

MIPS_SystemC Compound Index

3.1 MIPS_SystemC Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

add (Adder module) . . . 9
alu (ALU module) . . . 10
and (`and` module) . . . 11
andgate (`andgate` module) . . . 12
control (Control module) . . . 13
decode (Decode module) . . . 14
dmem (Dmem module) . . . 15
ext (`ext` module) . . . 16
hazard (Hazard module) . . . 17
imem (Imem module) . . . 18
mips (MIPS module) . . . 19
mux< **T** > (Mux module) . . . 21
orgate (`orgate` module) . . . 22
reg_exe_mem_t (Reg_exe_mem_t module) . . . 23
reg_id_exe_t (Reg_id_exe_t module) . . . 24
reg_if_id_t (Reg_if_id_t module) . . . 25
reg_mem_wb_t (Reg_mem_wb_t module) . . . 26
regfile (Regfile module) . . . 27
registo (Registo module) . . . 28
regT< **T** > (RegT template) . . . 29
shiftl2 (`shiftl2` module) . . . 30

Chapter 4

MIPS_SystemC File Index

4.1 MIPS_SystemC File List

Here is a list of all documented files with brief descriptions:

main.cpp (Defines `sc_main` function where all SystemC programs start) 31
mipsaux.h (Auxiliary functions related to MIPS) 32

Chapter 5

MIPS_SystemC Class Documentation

5.1 add Struct Reference

adder module.

Inherits sc_module.

Public Methods

- void **calc** ()
  callback function of module `add`.

5.1.1 Detailed Description

adder module.

adder module adds two sc_uint<32> numbers without overflow or carry

- input ports
  - `sc_uint<32> op1` - first operand
  - `sc_uint<32> op2` - second operand
- output ports
  - `sc_uint<32> res` - result

The documentation for this struct was generated from the following files:

- add.h
- add.cpp

5.2 alu Struct Reference

ALU module.

Inherits sc_module.

Public Methods

- void **calc** ()

  `alu` *module callback function.*

5.2.1 Detailed Description

ALU module.

ALU module models the integer ALU of the MIPS. It can perform 6 operations: add, subtract, and, or, set on equal

- input ports
  - `sc_uint<32> din1` - first operand
  - `sc_uint<32> din2` - second operand
  - `sc_uint<3> op` - selects operation
- output ports
  - `sc_unint<32> dout` - result
  - `bool zero` - active if result = 0

The documentation for this struct was generated from the following files:

- alu.h
- alu.cpp

5.3 and Struct Reference

`and` module.

Inherits sc_module.

Public Methods

- void **entry** ()

  `and` *module callback function.*

5.3.1 Detailed Description

`and` module.

`and` module models a two input and gate

- input ports
  - `bool din1` - first input
  - `bool din2` - second input
- output ports
  - `bool dout` - result

The documentation for this struct was generated from the following files:

- and.h
- and.cpp

5.4 andgate Struct Reference

`andgate` module.

Inherits sc_module.

Public Methods

- void **entry** ()

  `andgate` *module callback function.*

5.4.1 Detailed Description

`andgate` module.

`andgate` module models a two input and gate

- input ports
  - `bool din1` - first input
  - `bool din2` - second input
- output ports
  - `bool dout` - result

The documentation for this struct was generated from the following files:

- gates.h
- gates.cpp

5.5 control Struct Reference

Control module.

Inherits sc_module.

Public Methods

- void **entry** ()

  `control` *module callback function.*

5.5.1 Detailed Description

Control module.

Control module models the control unit of MIPS.

- input ports
  - `sc_uint<6>` `opcode` - instruction opcode field
  - `sc_uint<6>` `funct` - instruction funct field
- output ports
  - `bool` `RegDst` - selects if rd or rt is written
  - `bool` `RegWrite` - enables writing in Register file
  - `bool` `MemRead` - enables reading from Memory
  - `bool` `MemWrite` - enables writing to Memory
  - `bool` `MemtoReg` - Value to write in register comes from memory
  - `sc_uint<6>` `ALUOp` - selects ALU operation
  - `bool` `ALUSrc` - selects ALU second operand
  - `bool` `Branch` - active if instruction is beq

The documentation for this struct was generated from the following files:

- control.h
- control.cpp

5.6 decode Struct Reference

Decode module.

Inherits sc_module.

Public Methods

- void **entry** ()

  `decode` *module callback function.*

5.6.1 Detailed Description

Decode module.

Decode module splits instruction in its fields.

- input ports
  - `sc_uint<32> inst` - instruction
- output ports
  - `sc_uint<5> rs` - instruction rs field
  - `sc_uint<5> rd` - instruction rd field
  - `sc_uint<5> rt` - instruction rt field
  - `sc_uint<16> imm` - instruction imm field
  - `sc_uint<6> opcode` - instruction opcode field
  - `sc_uint<5> funct` - instruction funct field
  - `sc_uint<6> shamt` - instruction shamt field

The documentation for this struct was generated from the following files:

- decode.h
- decode.cpp

5.7 dmem Class Reference

Dmem module.

Inherits sc_module.

Public Methods

- void **read_mem** ()

  callback of the assynchronous behaviour of module `dmem`

- void **write_mem** ()

  callback of the synchronous behaviour of module `dmem`

- void **dump** ()

  Writes the contents of memory to `stdout`.

5.7.1 Detailed Description

Dmem module.

Dmem module models the data memory of MIPS. Synchronous on writes, assynchronous on reads.

- input ports
  - `sc_uint<32> addr` - address
  - `sc_uint<32> din` - input data
  - `bool wr` - write enable
  - `bool rd` - read enable
  - `bool clk` - clock
- output ports
  - `sc_uint<32> dout` - output data

The documentation for this class was generated from the following files:

- dmem.h
- dmem.cpp

5.8 ext Struct Reference

`ext` module.

Inherits sc_module.

Public Methods

- void **entry** ()

  sign extends din from 16 bits to 32 bits.

5.8.1 Detailed Description

`ext` module.

`ext` module sign extends a sc_uint<16> value to a sc_uint<32>.

- input ports
  - `sc_uint<16> din` - input
- output ports
  - `sc_uint<16> dout` - output

The documentation for this struct was generated from the following files:

- ext.h
- ext.cpp

5.9 hazard Struct Reference

hazard module.

Inherits sc_module.

Public Methods

- void **detect_hazard** ()

  Callback for the hazrd detection of `hazard` *module.*

5.9.1 Detailed Description

hazard module.

hazard module is the hazard detection unit.

- input ports
  - `sc_uint<5>` `reg1` - first register being read
  - `sc_uint<5>` `reg2` - second register being read
  - `sc_uint<5>` `WriteReg_exe` - register to be written (EXE)
  - `sc_uint<5>` `WriteReg_mem` - register to be written (MEM)
  - `bool` `RegWrite_exe` - control signal of writing registers (EXE)
  - `bool` `RegWrite_mem` - control signal of writing registers (MEM)
- output ports
  - `bool` `enable_pc` - enables PC update
  - `bool` `enable_ifid` - enables IF/ID update
  - `bool` `reset_idexe` - resets IF/EXE

The documentation for this struct was generated from the following files:

- hazard.h
- hazard.cpp

5.10 imem Class Reference

Imem module.

Inherits sc_module.

Public Methods

- void **entry** ()

  `imem` *module callback function.*

5.10.1 Detailed Description

Imem module.

Imem module models the instruction memory of MIPS. Assynchronous on reads, no writes allowed.

- input ports
  - `sc_uint<32>` `addr` - address
- output ports
  - `sc_uint<32>` `inst` - instruction

The documentation for this class was generated from the following files:

- imem.h
- imem.cpp

5.11 mips Struct Reference

MIPS module.

Inherits sc_module.

Public Methods

- void **buildArchitecture** ()

  Instantiates the pipeline registers and calls other functions to buils stage specific components.

- void **buildIF** ()

  builds IF stage components

- void **buildID** ()

  builds ID stage components

- void **buildEXE** ()

  builds EXE stage components

- void **buildMEM** ()

  builds MEM stage components

- void **buildWB** ()

  builds WB stage components

5.11.1 Detailed Description

MIPS module.

MIPS module is the main module of the MIPS simulator. Instruction memory, register file, ALU, data memory, etc are instatiated and interconnected inside this module.

- input ports

  - `bool reset` - reset
  - `bool clk` - clock

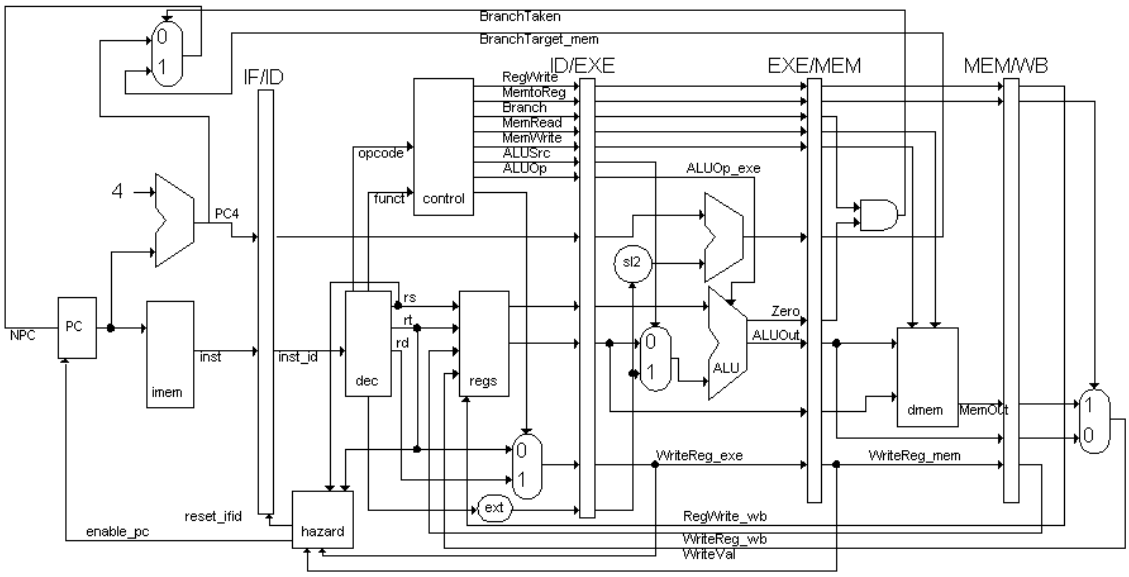

Figure 5.1: architecture of mips

The documentation for this struct was generated from the following files:

- mips.h
- mips.cpp

5.12 mux< T > Class Template Reference

Mux module.

Inherits sc_module.

5.12.1 Detailed Description

template<class T> class mux< T >

Mux module.

Mux module models a generic 2:1 multiplexor of template type T. Implementation based on a template class.

- input ports
  - `T din0` - input
  - `T din1` - input
  - `bool sel` - select
- output ports
  - `T dout` - output

The documentation for this class was generated from the following file:

- mux.h

5.13 orgate Struct Reference

`orgate` module.

Inherits sc_module.

Public Methods

- void **entry** ()

  `orgate` *module callback function.*

5.13.1 Detailed Description

`orgate` module.

`orgate` module models a two input or gate

- input ports
  - `bool din1` - first input
  - `bool din2` - second input
- output ports
  - `bool dout` - result

The documentation for this struct was generated from the following files:

- gates.h
- gates.cpp

5.14 reg_exe_mem_t Struct Reference

reg_exe_mem_t module.

Inherits sc_module.

5.14.1 Detailed Description

reg_exe_mem_t module.

reg_exe_mem_t module is the EXE/MEM pipeline register.

The documentation for this struct was generated from the following file:

- reg_exe_mem.h

5.15 reg_id_exe_t Struct Reference

reg_id_exe_t module.

Inherits sc_module.

5.15.1 Detailed Description

reg_id_exe_t module.

reg_id_exe_t module is the ID/EXE pipeline register.

The documentation for this struct was generated from the following file:

- reg_id_exe.h

5.16 reg_if_id_t Struct Reference

reg_if_id_t module.

Inherits sc_module.

5.16.1 Detailed Description

reg_if_id_t module.

reg_if_id_t module is the IF/ID pipeline register.

The documentation for this struct was generated from the following file:

- reg_if_id.h

5.17 reg_mem_wb_t Struct Reference

reg_mem_wb_t module.

Inherits sc_module.

5.17.1 Detailed Description

reg_mem_wb_t module.

reg_mem_wb_t module is the MEM/WB pipeline register.

The documentation for this struct was generated from the following file:

- reg_mem_wb.h

5.18 regfile Class Reference

Regfile module.

Inherits sc_module.

Public Methods

- void **init_regs** ()

  instantiates the 32 registers and initializes values to 0.

- void **dump** ()

  Used for debugging.

- void **regfile_access** ()

  Callback for behaviour of `regfile` *module.*

5.18.1 Detailed Description

Regfile module.

Regfile module models the integer register bank of MIPS. Synchronous on writes, assinchronous on read.

- input ports
  - `sc_uint<5> reg1` - id1 of register to read
  - `sc_uint<5> reg2` - id2 of register to read
  - `sc_uint<5> regwrite` - id of register to write
  - `sc_uint<32> datawr` - value to write
  - `bool wr` - write enable
  - `bool clk` - clock
  - `bool reset` - reset
- output ports
  - `sc_uint<32> data1` - value of register id1
  - `sc_uint<32> data2` - value of register id2

The documentation for this class was generated from the following files:

- regfile.h
- regfile.cpp

5.19 registo Struct Reference

Registo module.

Inherits sc_module.

Public Methods

- void **entry** ()

  `registo` *module callback function.*

5.19.1 Detailed Description

Registo module.

Registo module implements a 32 bit register. Synchronous on writes, assinchronous on reset.

- input ports
  - `sc_uint<32> din` - input
  - `bool reset` - reset
  - `bool clk` - clock
- output ports
  - `sc_uint<32> dout` - output

The documentation for this struct was generated from the following files:

- reg.h
- reg.cpp

5.20 regT< T > Class Template Reference

regT template.

Inherits sc_module.

5.20.1 Detailed Description

template<class T> class regT< T >

regT template.

regT template implements a variable width register. The type of data is selected by the template class T. Synchronous on writes and resets.

- input ports
  - `T din` - input
  - `bool reset` - reset
  - `bool enable` - enable
  - `bool clk` - clock
- output ports
  - `T dout` - output

The documentation for this class was generated from the following file:

- regT.h

5.21 shiftl2 Struct Reference

`shiftl2` module.

Inherits sc_module.

Public Methods

- void **entry** ()

  `shifl2` *module callback function.*

5.21.1 Detailed Description

`shiftl2` module.

`shiftl2` module shifts a sc_uint<32> two bits to the left.

- input ports
  - `sc_uint<32>` `din` - input
- output ports
  - `sc_uint<32>` `dout` - output

The documentation for this struct was generated from the following files:

- shiftl2.h
- shiftl2.cpp

Chapter 6

MIPS_SystemC File Documentation

6.1 main.cpp File Reference

defines `sc_main` function where all SystemC programs start.

Functions

- int **sc_main** (int argc, char *argv[])

  `MIPS_SystemC` `sc_main`, *instantiates the* **mips** *module and creates the Qt application that manages the user interface.*

6.1.1 Detailed Description

defines `sc_main` function where all SystemC programs start.

6.2 mipsaux.h File Reference

auxiliary functions related to MIPS.

Functions

- void **disassemble** (unsigned cod, char *decod)

  disassemble determines the assembly instruction from the machine code.

6.2.1 Detailed Description

auxiliary functions related to MIPS.

6.2.2 Function Documentation

6.2.2.1 void disassemble (unsigned *cod*, char * *decod*)

disassemble determines the assembly instruction from the machine code.

disassemble is a function that given the machine code `cod` determines the corresponding MIPS native assembly language instruction and store it in the `decod` string.

Index

add, 9
    calc, 9
alu, 10
    calc, 10
and, 11
    entry, 11
andgate, 12
    entry, 12

buildArchitecture
    mips, 19
buildEXE
    mips, 19
buildID
    mips, 19
buildIF
    mips, 19
buildMEM
    mips, 19
buildWB
    mips, 19

calc
    add, 9
    alu, 10
control, 13
    entry, 13

decode, 14
    entry, 14
detect_hazard
    hazard, 17
disassemble
    mipsaux.h, 32
dmem, 15
    dump, 15
    read_mem, 15
    write_mem, 15
dump
    dmem, 15
    regfile, 27

entry
    and, 11
    andgate, 12
    control, 13
    decode, 14
    ext, 16
    imem, 18
    orgate, 22
    registo, 28
    shiftl2, 30
ext, 16
    entry, 16

hazard, 17
    detect_hazard, 17

imem, 18
    entry, 18
init_regs
    regfile, 27

main.cpp, 31
    sc_main, 31
mips, 19
    buildArchitecture, 19
    buildEXE, 19
    buildID, 19
    buildIF, 19
    buildMEM, 19
    buildWB, 19
mipsaux.h, 32
    disassemble, 32
mux, 21

orgate, 22
    entry, 22

read_mem
    dmem, 15
reg_exe_mem_t, 23
reg_id_exe_t, 24
reg_if_id_t, 25
reg_mem_wb_t, 26
regfile, 27
    dump, 27
    init_regs, 27
    regfile_access, 27
regfile_access
    regfile, 27

registo, 28
    entry, 28
regT, 29

sc_main
    main.cpp, 31
shiftl2, 30
    entry, 30

write_mem
    dmem, 15