Single Cycled Simple Risc Processor Design in VHDL

Generated by Doxygen 1.8.5

Mon Dec 30 2013 11:02:23

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Design Unit Index

2.1 Design Unit Hierarchy

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

Chapter 3

Design Unit Index

3.1 Design Unit List

Here is a list of all design unit members with links to the Entities they belong to:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Namespace Documentation

5.1 fileRead Namespace Reference

This function converts a char to std_logic.

5.1.1 Detailed Description

This function converts a char to std_logic.

Chapter 6

Class Documentation

6.1 fileRead Package Body Reference

Package >> fileRead

Functions

- **std_logic** to_std_logic_from_char(**data: in character**)
- **std_logic_vector** to_std_logic_vector(**data: in string**)

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

- fileRead.vhdl

6.2 ALU_BU Architecture Reference

OFU is the architectural description of the EX Unit.

Signals

- E **boolean:=false**

  a boolean signal storing the boolean value of equality operation
- GT **boolean:=false**

  a boolean signal storing the boolean value of greater-than operation
- A **std_logic_vector(31 downto 0)**

  The first operand in the ALU.
- B **std_logic_vector(31 downto 0)**

  The second operand in the ALU.

6.2.1 Detailed Description

OFU is the architectural description of the EX Unit.

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

- EXUnit.vhdl

6.3 CU Architecture Reference

CU is the architectural description of the Control Unit.

Processes

- PROCESS_0(**op_code** , **I**)

  Extracting the opcode from the 32-bit instruction.

Signals

- op_code **std_logic_vector(4 downto 0)**

  A vector containing 5 bits of the instruction marking the instruction type.
- I **std_logic**

  The immediate bit.

6.3.1 Detailed Description

CU is the architectural description of the Control Unit.

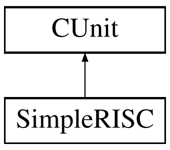
The documentation for this class was generated from the following file:

- CUnit.vhdl

6.4 CUnit Entity Reference

This is the unit that generates the control signals.

Inheritance diagram for CUnit:



Entities

- CU architecture

  CU is the architectural description of the Control Unit.

Libraries

- ieee

Use Clauses

- ieee.std_logic_1164.all

Ports

- Instruction **in std_logic_vector(31 downto 0)**

  The 32-bit instruction coming from the Instruction fetch Unit.
- isMov **out boolean**

  boolean which states whether the instruction is 'mov'
- isSt **out boolean**

  boolean which states whether the instruction is 'st'
- isLd **out boolean**

  boolean which states whether the instruction is 'ld'
- isBeq **out boolean**

  boolean which states whether the instruction is 'beq'
- isBgt **out boolean**

  boolean which states whether the instruction is 'bgt'
- isImmediate **out boolean**

  boolean which the instruction has an immediate as the arguments.
- isWb **out boolean**

  boolean whether something has to be written into the register file.
- isUBranch **out boolean**

  boolean whether the statement is a direct branching one(b, ret, call)
- isRet **out boolean**

  boolean which states whether the instruction is 'ret'
- isCall **out boolean**

  boolean which states whether the instruction is 'call'
- aluS **out std_logic_vector(2 downto 0)**

  This vector stores all the states with each 3-bit value corresponding to one of the instructions.

6.4.1 Detailed Description

This is the unit that generates the control signals.

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

- CUnit.vhdl

6.5 DM Architecture Reference

DM is the architecture of the Memory Unit.

Processes

- PROCESS_2( )

Signals

- memory **std_logic_vector(32767 downto 0)**

  This is a RAM of 32768 bits that is 4096 bytes.

6.5.1 Detailed Description

DM is the architecture of the Memory Unit.

6.5.2 Member Function Documentation

6.5.2.1 PROCESS_2 ( ) [Process]

If there is a store instruction then this process stores the value of op2 in the required memory location If there is a load instruction then the value of required memory location is loaded into ldResult signal

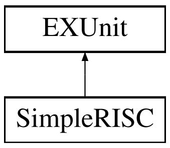
The documentation for this class was generated from the following file:

- MAUnit.vhdl

6.6 EXUnit Entity Reference

This is the unit that implements the ALU and branch Unit.

Inheritance diagram for EXUnit:



Entities

- ALU_BU architecture

  OFU is the architectural description of the EX Unit.

Libraries

- ieee

Use Clauses

- ieee.std_logic_1164.all
- ieee.numeric_std.all

Ports

- op1 **in std_logic_vector(31 downto 0)**

  Operand 1.
- op2 **in std_logic_vector(31 downto 0)**

  Operand 2.
- immediate **in std_logic_vector(31 downto 0)**

  The value of the immediate converted from 27 to 32 bit.
- aluS **in std_logic_vector(2 downto 0)**

  This vector marks the type of instruction.
- aluR **out std_logic_vector(31 downto 0)**

  This store the result of the arithmetic operation.
- isMov **in boolean**

  boolean which states whether the instruction is 'mov'

- isBeq **in boolean**

  boolean which states whether the instruction is 'beq'
- isBgt **in boolean**

  boolean which states whether the instruction is 'bgt'
- isUBranch **in boolean**

  boolean which states whether the instruction is a direct branching statement.
- isImmediate **in boolean**

  boolean which states whether the instruction has an immediate.
- isBranchTaken **out boolean**

  boolean whether the branch is taken.

6.6.1 Detailed Description

This is the unit that implements the ALU and branch Unit.

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

- EXUnit.vhdl

6.7 fileRead Package Reference

This function converts a char to std_logic.

Package Body >> fileRead

Functions

- **std_logic** to_std_logic_from_char**(data: in character)**
- **std_logic_vector** to_std_logic_vector**(data: in string)**

  This function converts a string to a vector.

Libraries

- IEEE

  Whether the instruction is ret,call,b,beq or bgt, this boolean specifies whether the branch is actually taken.

Use Clauses

- IEEE.std_logic_1164.all

6.7.1 Detailed Description

This function converts a char to std_logic.

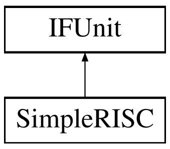
The documentation for this class was generated from the following file:

- fileRead.vhdl

6.8 IFUnit Entity Reference

This is the unit that implements the Instruction fetch Unit.

Inheritance diagram for IFUnit:



Entities

- IM architecture

  IM is the architectural description of the Instruction Fetch Unit.

Libraries

- ieee
- std

Use Clauses

- ieee.std_logic_1164.all
- std.textio.all
- ieee.numeric_std.all
- work.fileRead.all

  This import contains the manually defined function to take input from a file.

Ports

- PC **in std_logic_vector(31 downto 0)**

  The program counter.
- instruction **out std_logic_vector(31 downto 0)**

  The 32-bit instruction corresponding to the program counter.

6.8.1 Detailed Description

This is the unit that implements the Instruction fetch Unit.

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

- IFUnit.vhdl

6.9 IM Architecture Reference

IM is the architectural description of the Instruction Fetch Unit.

Processes

- PROCESS_1(**PC**)

  Executes only when the program counter is changed.

Types

- **memorydefarray(integerrange<>)ofstd_logic_vector(31 downto 0)**

  A custom data type for the memory.

Signals

- memory **memorydef (0 to 499)**

  The memory is a 500-length array of 32-bit vectors,ie, it can handle upto 500 instructions.
- maxCount **integer**

  Marks the maximum Program counter.

6.9.1 Detailed Description

IM is the architectural description of the Instruction Fetch Unit.

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

- IFUnit.vhdl

6.10 main Architecture Reference

Components

- IFUnit

  Instruciton Fetch Unit.
- CUnit

  Constrol Unit.
- OFUnit

  Operand Fetch Unit and Register Write Unit.
- EXUnit

  Execute Unit.
- MAUnit

  Memory access unit.

Signals

- clk **std_logic:=' 0 '**

  Clock Signal initialed with 0.
- PC **std_logic_vector(31 downto 0):=X" FFFFFFFC "**

  Program Counter initialized with -4.
- instruction **std_logic_vector(31 downto 0)**

  The instruction signal.
- isMov **boolean:=false**

  boolean for mov statement

- isSt **boolean:=false**

  boolean for store statement
- isLd **boolean:=false**

  boolean for load statement
- isBeq **boolean:=false**

  boolean for branch if equal statement
- isBgt **boolean:=false**

  boolean for branch if greater statement
- isImmediate **boolean:=false**

  boolean for the statements whre second operand is an immediate
- isWb **boolean:=false**

  boolean for the statements involving writing into register
- isUbranch **boolean:=false**

  boolean for call, ret, b, bgt, beq instructions
- isBranchTaken **boolean:=false**

  boolean which is true when branch is taken by call, ret, b, bgt, beq instructions
- isRet **boolean:=false**

  boolean for ret instruction
- isCall **boolean:=false**

  boolean for call instruction
- aluS **std_logic_vector(2 downto 0)**

  ALU Signals generated by the Control Unit for ALU to perform adequate operation.
- aluR **std_logic_vector(31 downto 0):=X" 00000000 "**

  Result of the operation by ALU.
- ldR **std_logic_vector(31 downto 0):=X" 00000000 "**

  value read from register file
- immediate **std_logic_vector(31 downto 0):=X" 00000000 "**

  immediate computer after applying modifiers and bit extension
- branchTarget **std_logic_vector(31 downto 0):=X" 00000000 "**

  brach target computed after adding offset to the program counter
- op1 **std_logic_vector(31 downto 0):=X" 00000000 "**

  Fisrt Operand.
- op2 **std_logic_vector(31 downto 0):=X" 00000000 "**

  Second Operand.

Instantiations

- iif **IFUnit**

  maps the signals to the ports of IFUnit
- icu **CUnit**

  maps the signals to the ports of CUnit
- iof **OFUnit**

  maps the signals to the ports of OFUnit
- iex **EXUnit**

  maps the signals to the ports of EXUnit
- ima **MAUnit**

  maps the signals to the ports of MAUnit

6.10.1 Detailed Description

This is main assembly of all the components of the prcessor. This is also responsible for generation of clock signal which is sent to all the other components. Further, program counter is also updated in this architecure description only.

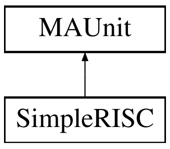
The documentation for this class was generated from the following file:

- SimpleRISC.vhdl

6.11 MAUnit Entity Reference

This is the unit that implements a RAM.

Inheritance diagram for MAUnit:



Entities

- DM architecture

    DM is the architecture of the Memory Unit.

Libraries

- ieee

Use Clauses

- ieee.std_logic_1164.all
- ieee.numeric_std.all

Ports

- clk **in std_logic**

    clk is the CLOCK coming the the unit
- op2 **in std_logic_vector(31 downto 0)**

    The value to stored in case of a store instruction.
- aluR **in std_logic_vector(31 downto 0)**

    aluR stores the address of the memory location to be accessed
- isSt **in boolean**

    a boolean signal generated by Control Unit to determine if this is a store instruction
- isLd **in boolean**

    a boolean signal generated by Control Unit to determine if this is a load instruction
- ldResult **out std_logic_vector(31 downto 0)**

    output signal which contains the value stored in the memory location accessed in case of a load instruction

6.11.1 Detailed Description

This is the unit that implements a RAM.

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

- MAUnit.vhdl

6.12 OFU Architecture Reference

OFU is the architectural description of the operand fetch unit and register file.

Processes

- PROCESS_3( )

Types

- **regvecarray(15 downto 0)ofstd_logic_vector(31 downto 0)**

  A custom data type which is a 16 x 32 array or vector of bits.

Signals

- reg **regvec**

  reg is the register file of data type regvec that is it has 16 32-bit vector fields
- temp **std_logic_vector(31 downto 0)**

  temp is the signal which is used for signed extension of offset, offset is Instruction(26 downto 0)

6.12.1 Detailed Description

OFU is the architectural description of the operand fetch unit and register file.

6.12.2 Member Function Documentation

6.12.2.1 PROCESS_3 ( ) [Process]

First of all immediate is computed after sign extesnion, if 17th bit of the instruction is 1 then sign extension is not done and if the 18th bit is 1 then the 16-bit immediate is shifted left by 16 bits to get the modified immidiate. The branch targeet is computer as $<$ program counter + 4 $*$ offset $>$ offset is stored as Instruction(15 downto 0). Frst operand is computer as reg(rs1) where rs1 is stored in Instruction(21 downto 18). Second operand is either reg(rs2) where rs1 is stored in Instruction(17 downto 14) or it is reg(rd) in case of a store instruction where rd is stored as Instruction(25 downto 22).

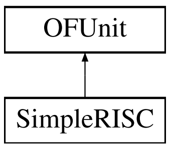
The documentation for this class was generated from the following file:

- OFUnit.vhdl

6.13 OFUnit Entity Reference

This the Operand Fetch Unit which also contains the register file implementation.

Inheritance diagram for OFUnit:



Entities

- OFU architecture

  OFU is the architectural description of the operand fetch unit and register file.

Libraries

- ieee

Use Clauses

- ieee.std_logic_1164.all
- ieee.numeric_std.all

Ports

- clk **in std_logic**

  CLOCK for the unit.
- Instruction **in std_logic_vector(31 downto 0)**

  Complete 32 bit instruction code.
- PC **in std_logic_vector(31 downto 0)**

  Program Counter.
- aluR **in std_logic_vector(31 downto 0)**

  ALU Result that is to be written in register in case of an alu instrcution.
- ldR **in std_logic_vector(31 downto 0)**

  Load Result that is to be written in register in case of a load instrcution.
- isSt **in boolean**

  boolean for store instruction
- isLd **in boolean**

  boolean for load instruction
- isWb **in boolean**

  boolean for writeback that is its true when something is to be written in the register file
- isRet **in boolean**

  boolean which states whether the instruction is 'ret'
- isCall **in boolean**

  boolean which states whether the instruction is 'call'
- immediate **out std_logic_vector(31 downto 0)**

  32 bit immediate that has to be computed
- branchTarget **out std_logic_vector(31 downto 0)**

This is the branch target which is used in case of a branch or call instruction.

- op1 **out std_logic_vector(31 downto 0)**

  First pperand.

- op2 **out std_logic_vector(31 downto 0)**

  Second Operand.

6.13.1 Detailed Description

This the Operand Fetch Unit which also contains the register file implementation.

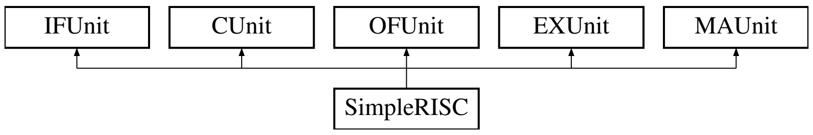
The documentation for this class was generated from the following file:

- OFUnit.vhdl

6.14 SimpleRISC Entity Reference

This is empty shell for the main compilation of all the other components in the processor.

Inheritance diagram for SimpleRISC:



Entities

- main architecture

Libraries

- ieee

Use Clauses

- ieee.std_logic_1164.all
- ieee.numeric_std.all

6.14.1 Detailed Description

This is empty shell for the main compilation of all the other components in the processor.

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

- SimpleRISC.vhdl

Chapter 7

File Documentation

7.1 CUnit.vhdl File Reference

This file for the implementation of a Control Unit for the processor.

Entities

- CUnit entity

  This is the unit that generates the control signals.
- CU architecture

  CU is the architectural description of the Control Unit.

7.1.1 Detailed Description

This file for the implementation of a Control Unit for the processor.

Author

Kunal Singhal and Swapnil Palash

7.2 EXUnit.vhdl File Reference

This file is for the Arithmetic and logic Unit of the processor.

Entities

- EXUnit entity

  This is the unit that implements the ALU and branch Unit.
- ALU_BU architecture

  OFU is the architectural description of the EX Unit.

7.2.1 Detailed Description

This file is for the Arithmetic and logic Unit of the processor.

Author

Kunal Singhal and Swapnil Palash

7.3 fileRead.vhdl File Reference

This file for the implementation of file reading function.

Entities

- fileRead package

  This function converts a char to std_logic.
- fileRead package body

7.3.1 Detailed Description

This file for the implementation of file reading function.

Author

Kunal Singhal and Swapnil Palash

7.4 IFUnit.vhdl File Reference

This file for the implementation of a Instruction Fetch Unit for the processor.

Entities

- IFUnit entity

  This is the unit that implements the Instruction fetch Unit.
- IM architecture

  IM is the architectural description of the Instruction Fetch Unit.

7.4.1 Detailed Description

This file for the implementation of a Instruction Fetch Unit for the processor.

Author

Kunal Singhal and Swapnil Palash

7.5 MAUnit.vhdl File Reference

This file for the implementation of a memory for the processor.

Entities

- MAUnit entity

  This is the unit that implements a RAM.
- DM architecture

  DM is the architecture of the Memory Unit.

7.5.1 Detailed Description

This file for the implementation of a memory for the processor.

Author

Kunal Singhal and Swapnil Palash

7.6 OFUnit.vhdl File Reference

Operand Fetch unit as well as register read and write operations implemented.

Entities

- OFUnit entity

  This the Operand Fetch Unit which also contains the register file implementation.
- OFU architecture

  OFU is the architectural description of the operand fetch unit and register file.

7.6.1 Detailed Description

Operand Fetch unit as well as register read and write operations implemented.

Author

Kunal Singhal and Swapnil Palash

7.7 SimpleRISC.vhdl File Reference

This is compilation of all the components of the processor.

Entities

- SimpleRISC entity

  This is empty shell for the main compilation of all the other components in the processor.
- main architecture

7.7.1 Detailed Description

This is compilation of all the components of the processor.

Author

Kunal Singhal and Swapnil Palash

Index