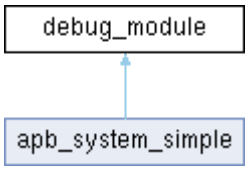
debug_module.vhd File Reference

Entities

debug_module	entity
debug_module.behavioral	architecture

debug_module Entity Reference

Inheritance diagram for debug_module:



Entities

debug_module.behavioral architecture

Libraries

ieee

Use Clauses

std_logic_1164
numeric_std
nanorv_pkg **Package** ***<nanorv_pkg>***

Ports

PCLK	**in**	**std_logic**
PRSTn	**in**	**std_logic**
DEBUG_IN	**in**	**debug_inputs**
wr_value	**in**	**std_logic_vector (31 downto 0)**
rd_addr	**in**	**std_logic_vector (JTAG_REG_ADDR_WIDTH - 1 downto 0)**
wr_ena	**in**	**std_logic**
wr_addr	**in**	**std_logic_vector (JTAG_REG_ADDR_WIDTH - 1 downto 0)**
rd_value	**out**	**std_logic_vector (31 downto 0)**
DEBUG_OUT	**out**	**debug_outputs**

Detailed Description

Debug module This module is used to connect the CPU to the JTAG interface. It contains five registers files (JTAG_USER_DATA_REG): control, status, memory address, write data and read data. The control register (JTAG_CTRL_REG : Address of the control register) is used to control the state of the module. The status register (JTAG_STATUS_REG : Address of the status register) is used to store the status of the module. The memory address register (JTAG_MEMADDR_REG : Address of the memory register) is used to store the address of the memory. The write data register (JTAG_WRDATA_REG : Address of the write data register) is used to store the data to be written to the memory. The read data register (JTAG_RDDATA_REG : Address of the read data register) is used to store the data read from the memory. The module with this registers files can store or load the values of the instruction register, the program counter and the memory interface. It also contains a state machine that is used to control the state of the module, if it is in Go mode, in Stop mode, or in Busy mode. The module is connected to the CPU through the debug_inputs and debug_outputs interfaces. The

module is connected to the JTAG interface through the JTAG_USER_DATA_REG register file. The module is used to control the state of the CPU and to read and write the values of the program counter, the instruction register and the memory interface. The module is used to control the state of the CPU and to read and write the values of the program counter, the instruction register and the memory interface.

Member Data Documentation

◆ DEBUG_IN

DEBUG_IN in debug_inputs

Input signals for the debug and they comes from the CPU.

◆ DEBUG_OUT

DEBUG_OUT out debug_outputs

Output signals for the debug and they goes in input to the CPU.

◆ ieee

ieee

◆ nanorv_pkg

nanorv_pkg

◆ numeric_std

numeric_std

◆ PCLK

PCLK in std_logic

Clock signal.

◆ PRSTn

PRSTn in std_logic

Reset signal active on the falling edge.

◆ rd_addr

rd_addr in std_logic_vector (JTAG_REG_ADDR_WIDTH - 1 downto 0)

Address of the register to be read.

◆ rd_value

rd_value out std_logic_vector (31 downto 0)

Value read from the register file.

◆ std_logic_1164

std_logic_1164

◆ wr_addr

wr_addr in std_logic_vector (JTAG_REG_ADDR_WIDTH - 1 downto 0)

Address of the register to be written.

◆ wr_ena

wr_ena in std_logic

Write enable signal.

◆ wr_value

wr_value in std_logic_vector (31 downto 0)

Value to be written to the register file.

The documentation for this design unit was generated from the following file:

- C:/tirocinio/nanorv/source/**debug_module.vhd**

debug_module.behavioral Architecture Reference

Architecture >> debug_module::behavioral

Processes

FSM_go_update	(PCLK , PRSTn)
FSM_handle_go	(go_current_state , jtag_go , DEBUG_IN **.fetch**)
FSM_memory_update	(PCLK , PRSTn)
FSM_memory_operation	(mem_current_state , read_mem , write_mem , actual_go)
FSM_reg_file_update	(PCLK , PRSTn)
FSM_reg_file	(reg_current_state , reg_wr , reg_rd , actual_go)
jtag_wr	(PCLK , PRSTn)
jtag_rd	(PCLK)

Use Clauses

nanorv_pkg **Package** *<nanorv_pkg>*

Types

jtag_regfile_t	(0 to JTAG_REG_NUMBER - 1) std_logic_vector (31 downto 0)
state_GO	(GO , STOP , BUSY , FETCH_BUSY , RESET , P_STOP)
state_MEM	(IDLE , MEM_BUSY , DONE)
state_REG	(REG_IDLE , REG_BUSY , REG_DONE)

Signals

actual_go	std_logic
jtag_mem_busy	std_logic
JTAG_USER_DATA_REG	jtag_regfile_t
go_current_state	state_GO
go_next_state	state_GO
mem_current_state	state_MEM
mem_next_state	state_MEM
reg_current_state	state_REG
reg_next_state	state_REG
reg_busy_flag	std_logic

Aliases

mem_addr	std_logic_vector (31 downto 0) is JTAG_USER_DATA_REG (JTAG
wr_data	std_logic_vector (31 downto 0) is JTAG_USER_DATA_REG (JTA
rd_data	std_logic_vector (31 downto 0) is JTAG_USER_DATA_REG (JTA
status	std_logic_vector (31 downto 0) is JTAG_USER_DATA_REG (JT
rstn	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_
jtag_go	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTA
read_pc	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_
write_pc	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_
reg_addr	std_logic_vector (4 downto 0) is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_REGADDR_BIT_H downto JTAG_CTRL_
reg_rd	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_C
reg_wr	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_C
mem_size	std_logic_vector (1 downto 0) is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_MEMSIZE_BIT_H downto JTAG_CTRL
read_mem	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_C
write_mem	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_C
read_unsigned	std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_

Member Function/Procedure/Process Documentation

◆ FSM_go_update()

FSM_go_update (**PCLK** ,
PRSTn)

The process is used to update the state of the module (GO, RESET, BUSY, FETCH_BUSY, STOP and P_STOP)

◆ FSM_handle_go()

```
FSM_handle_go ( go_current_state ,
                jtag_go ,
                DEBUG_IN .fetch  )
```

The process is used to handle the state of the debug module depending on the fetch of the CPU, jtag_go signal and the current state of the module.

◆ FSM_memory_operation()

```
FSM_memory_operation ( mem_current_state ,
                       read_mem ,
                       write_mem ,
                       actual_go  )
```

The process is used to handle the state of the debug module during the operation with the memory of the CPU.

◆ FSM_memory_update()

```
FSM_memory_update ( PCLK ,
                    PRSTn  )
```

The process is used to update the state of the memory module (IDLE, MEM_BUSY and DONE)

◆ FSM_reg_file()

```
FSM_reg_file ( reg_current_state ,
               reg_wr ,
               reg_rd ,
               actual_go  )
```

The process is used to handle the state of the debug module during the operation with the register file of the CPU.

◆ FSM_reg_file_update()

```
FSM_reg_file_update ( PCLK ,
                      PRSTn  )
```

The process is used to update the state of the register file (REG_IDLE, REG_BUSY and REG_DONE)

◆ jtag_rd()

```
jtag_rd ( PCLK  )
```

JTAG Read process The JTAG read process is used to read the values from the JTAG_USER_DATA_REG register file and send them to the CPU.

◆ jtag_wr()

```
jtag_wr ( PCLK ,
          PRSTn  )
```

The JTAG write process is used to write the values from the JTAG interface to the JTAG_USER_DATA_REG register file. The JTAG connection is used to connect the CPU to the JTAG interface. It uses the JTAG_USER_DATA_REG register file to write or read the values from JTAG to the instruction register, the program counter and the memory interface of the CPU. Update JTAG status register every clock cycle, if a reset event comes the register are settet to zero. Remember that the first two registers are only reading.

Member Data Documentation

actual_go

actual_go std_logic

go_current_state

go_current_state state_GO

Start with the Reset state.

The FSM_GO is used to control the state of the module, it has six states: The GO state is the normal operation state. The STOP state is used to stop the module. The BUSY state is used to indicate that the module is busy. The FETCH_BUSY state is used to indicate that the module is busy fetching data. The RESET state is used to reset the module. The P_STOP state is used to stop the module and put it in a low power state.

go_next_state

go_next_state state_GO

Start with the Reset state.

jtag_go

jtag_go std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_GO_BIT)

JTAG go signal, used to control the state of the module (could be Go and Stop);.

jtag_mem_busy

jtag_mem_busy std_logic

jtag_regfile_t

jtag_regfile_t (0 to JTAG_REG_NUMBER - 1) std_logic_vector (31 downto 0)

JTAG_USER_DATA_REG

JTAG_USER_DATA_REG jtag_regfile_t

JTAG user data registers, used to store the values of the registers.

mem_addr

mem_addr std_logic_vector (31 downto 0) is JTAG_USER_DATA_REG (JTAG_MEMADDR_REG)

JTAG Memory address register, used to store the address of the memory to be read or written;.

mem_current_state

mem_current_state state_MEM

Start with the IDLE state.

The FSM_MEM is used to control the state of the memory, it has three states: The IDLE state is the normal operation state. The MEM_BUSY state is used to indicate that the memory is busy. The DONE state is used to indicate that the memory has finished the operation.

mem_next_state

mem_next_state state_MEM

Start with the IDLE state.

◆ mem_size

mem_size std_logic_vector (1 downto 0) is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_MEMSIZE_BIT_H downto JTAG_CTRL_MEMSIZE_BIT_L)

JTAG memory size signal, used to store the size of the memory to be read or written;.

◆ nanorv_pkg

nanorv_pkg

◆ rd_data

rd_data std_logic_vector (31 downto 0) is JTAG_USER_DATA_REG (JTAG_RDDATA_REG)

JTAG read data register, used to store the data read from the memory;.

◆ read_mem

read_mem std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_RDMEM_BIT)

JTAG read memory signal, is the enable signal used to read the value of the memory;.

◆ read_pc

read_pc std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_RDPC_BIT)

JTAG read program counter signal, is the enable signal used to read the value of the program counter;.

◆ read_unsigned

read_unsigned std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_RDMEM_UNS_BIT)

JTAG read unsigned memory signal, is the enable signal used to read the value of the memory as an unsigned value;.

◆ reg_addr

reg_addr std_logic_vector (4 downto 0) is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_REGADDR_BIT_H downto JTAG_CTRL_REGADDR_BIT_L)

JTAG register address signal, used to store the address of the register to be read or written;.

◆ reg_busy_flag

reg_busy_flag std_logic

◆ reg_current_state

reg_current_state state_REG

Start with the REG_IDLE state.

The FSM_REG is used to control the state of the register file, it has three states: The REG_IDLE state is the normal operation state. The REG_BUSY state is used to indicate that the register file is busy. The REG_DONE state is used to indicate that the register file has finished the operation.

◆ reg_next_state

reg_next_state state_REG

Start with the REG_IDLE state.

◆ reg_rd

reg_rd std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_RDREG_BIT)

JTAG register read signal, is the enable signal used to read the value of the register file;.

◆ reg_wr

reg_wr std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_WRREG_BIT)

JTAG register write signal, is the enable signal used to write the value of the register file;.

◆ rstn

rstn std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_RSTN_BIT)

JTAG control register, used to control the state of the module and to read and write the values of the program counter, the instruction register and the memory interface. The JTAG control register has the following bits:JTAG reset signal, used to reset the module and it's active onthe falling edge;.

◆ state_GO

state_GO (GO , STOP , BUSY , FETCH_BUSY , RESET , P_STOP)

The FSMs are used to handle the state of the module during the operation with the memory and the register file of the CPU.

◆ state_MEM

state_MEM (IDLE , MEM_BUSY , DONE)

◆ state_REG

state_REG (REG_IDLE , REG_BUSY , REG_DONE)

◆ status

status std_logic_vector (31 downto 0) is JTAG_USER_DATA_REG (JTAG_STATUS_REG)

JTAG status register, used to store the status of the module;.

◆ wr_data

wr_data std_logic_vector (31 downto 0) is JTAG_USER_DATA_REG (JTAG_WRDATA_REG)

JTAG write data register, used to store the data to be written to the memory;.

◆ write_mem

write_mem std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_WRMEM_BIT)

JTAG write memory signal, is the enable signal used to write the value of the memory;.

◆ write_pc

write_pc std_logic is JTAG_USER_DATA_REG (JTAG_CTRL_REG) (JTAG_CTRL_WRPC_BIT)

JTAG write program counter signal, is the enable signal used to write the value of the program counter;.

The documentation for this design unit was generated from the following file:

- C:/tirocinio/nanorv/source/**debug_module.vhd**