Config_register_set Module

Given Code:

```
module config_register_set #(
   parameter REG_WIDTH = 8,          // Width of each register (e.g., 8 bits)
   parameter NUM_REGS = 4)           // Number of configuration registers
(  input wire clk,                  // Clock signal
   input wire rst_n,                // Active low reset
   input wire [REG_WIDTH-1:0] wr_data,// Data to write
   input wire [1:0] wr_addr,        // Write address (2 bits for 4 registers)
   input wire write_enable,         // Write enable signal
   output reg [REG_WIDTH-1:0] rd_data // Data to read);

   // Internal register array
   reg [REG_WIDTH-1:0] reg_array [NUM_REGS-1:0];

   // Declare integer variable outside the always block
   integer i;

   // Write operation
   always @(posedge clk or negedge rst_n)
    begin
      if (!rst_n)
        //begin
         // Reset all registers to zero
         for (i = 0; i < NUM_REGS; i = i + 1)
            //begin
            reg_array[i] <= {REG_WIDTH{8'b0}};
         //end
      //end
        else if (write_enable)
        begin
         // Write data to specified register address
         reg_array[wr_addr] <= wr_data;
      end
   end

   // Read operation (combinational)
   always @(*) begin
      rd_data = reg_array[wr_addr];
   end

endmodule
```

- The module is configurable set of register that provides an interface to write and read data from multiple registers
- Parameter we can adjust the size of Number of registers and the size of the each register.

Module parameters

Reg_width: Defines the bit width of each register in this code default is set to 8bits
But this can be configured based on system requirments(8bit)
Num_regs: Specifies the Number of Configuration registers(4 registers)

Register1: Address=2’b00

1	0	1	0	0	1	0	1

Register2: Address=2’b01

0	1	0	1	1	0	1	0

Register3: Address=2’b10

1	1	1	1	1	1	1	1

Register4: Address=2’b11

0	0	0	0	1	1	1	1

PORTS:

CLK: Clock Controls the Timing for Synchronous Operations Such as Writing to registers

rst_n: If Active low reset Asserted this reset initializes all registers to zero

write_enable: When Write enable is high it enables the write Operation to the register Specified by Wr_addr

wr_data: Write data (input). The data to be written into the specified register.

rd_data: Holds the data read from the registers addressed by wr_addr

wr_addr: Write address (input, 2 bits). Specifies the address (0 to 3) of the register where the data should be written. Since there are 4 registers, only 2 bits are needed to represent the address.

Internal Register(array)

- **Register Array (reg_array)**: The internal storage for register values, defined as an array of NUM_REGS elements, each with a width of REG_WIDTH bits. This array holds the configuration values for the module.

Operation:

Reset Behavior

Upon a low level on rst_n (active-low reset), all registers within reg_array are initialized to zero. This is managed within a for loop, ensuring that each register in the array is zeroed on reset. This is critical for initializing all registers to a known state before use.

When rst_n is low, all registers are cleared to zero.

Write Operation

The write operation is controlled by both the clk signal and the write_enable flag. On the rising edge of clk, if write_enable is asserted high, the data in wr_data is stored into the register addressed by wr_addr. This allows controlled data entry into specific registers.

When write_enable is high on the clock's rising edge, the module writes wr_data into the register located at wr_addr.

Read Operation

The read operation is purely combinational, meaning it does not depend on the clock. The output rd_data continuously reflects the value in the register addressed by wr_addr. This immediate reflection allows faster access to the register data for downstream modules.

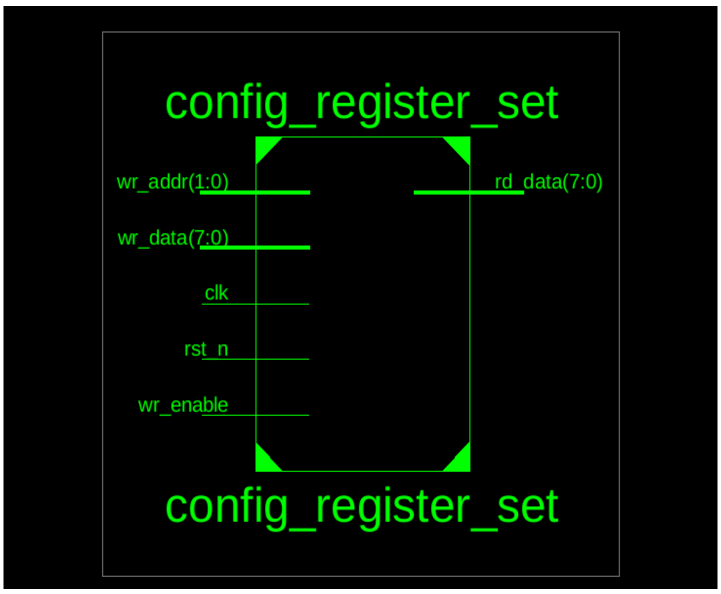
The output rd_data continuously shows the contents of the register addressed by wr_addr, allowing real-time access to stored data.

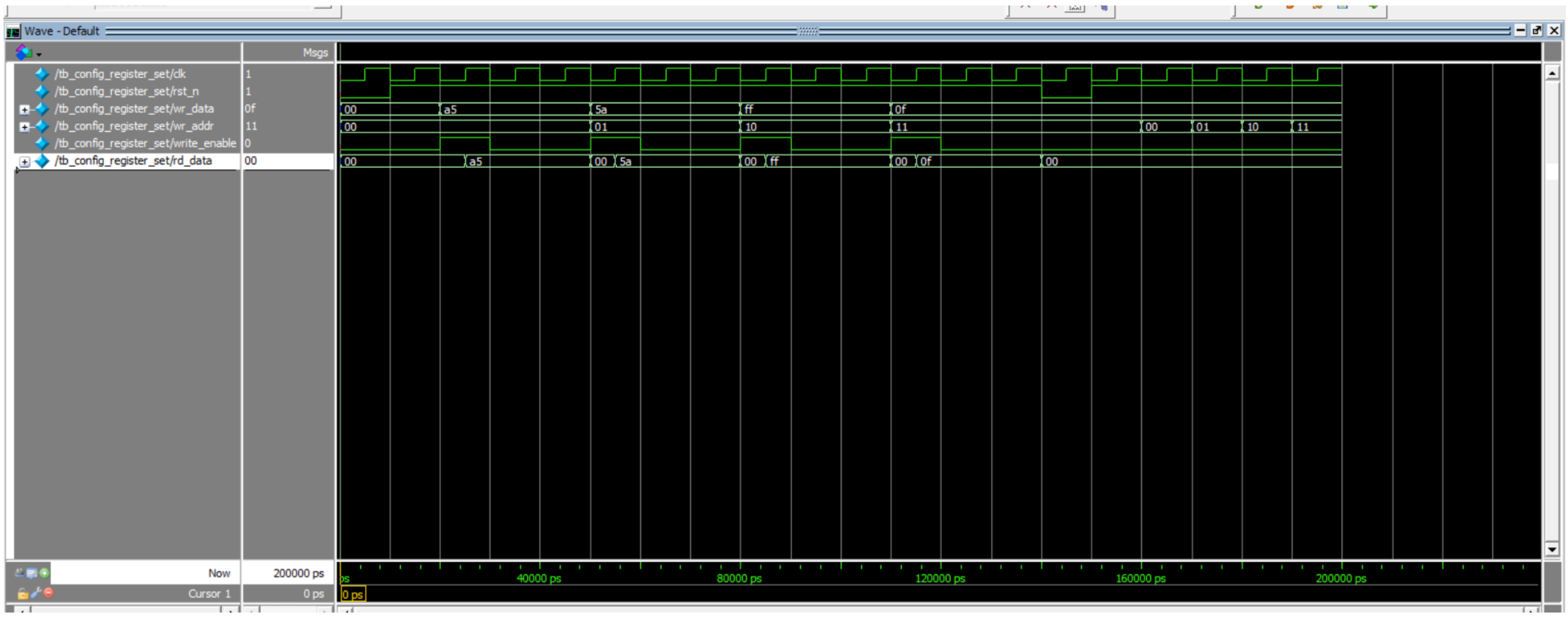
Used In:

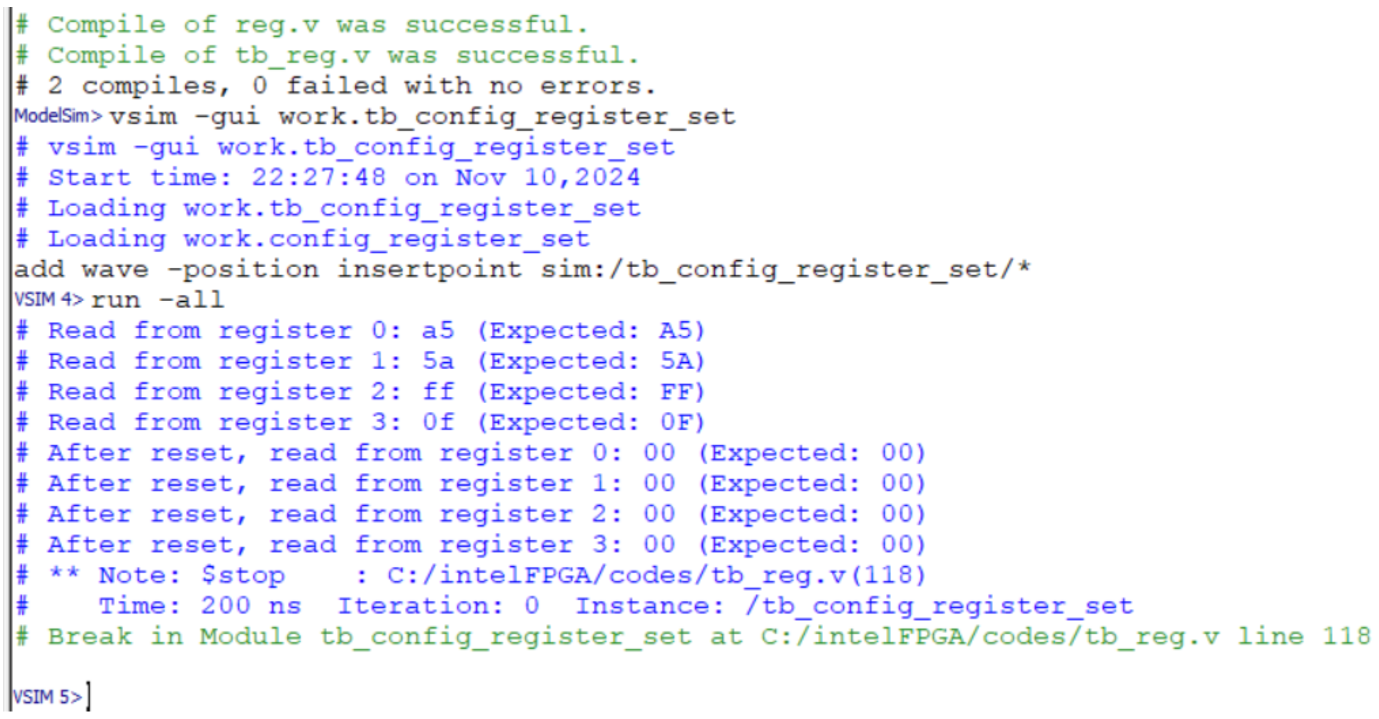
This config_register_set module can be used in various applications such as:

- Microcontroller configurations
- Memory-mapped register banks
- Configuration control in embedded systems

Results:




Wave - Default
Msgs
/tb_config_register_set/clk 1
/tb_config_register_set/rst_n 1
/tb_config_register_set/wr_data 0f
/tb_config_register_set/wr_addr 11
/tb_config_register_set/write_enable 0
/tb_config_register_set/rd_data 00
00 a5 5a ff 0f
00 01 10 11 00 01 10 11
00 a5 00 5a 00 ff 00 0f 00
Now 200000 ps
Cursor 1 0 ps
40000 ps
80000 ps
120000 ps
160000 ps
200000 ps



```
# Compile of reg.v was successful.
# Compile of tb_reg.v was successful.
# 2 compiles, 0 failed with no errors.
ModelSim> vsim -gui work.tb_config_register_set
# vsim -gui work.tb_config_register_set
# Start time: 22:27:48 on Nov 10,2024
# Loading work.tb_config_register_set
# Loading work.config_register_set
add wave -position insertpoint sim:/tb_config_register_set/*
VSIM 4> run -all
# Read from register 0: a5 (Expected: A5)
# Read from register 1: 5a (Expected: 5A)
# Read from register 2: ff (Expected: FF)
# Read from register 3: 0f (Expected: 0F)
# After reset, read from register 0: 00 (Expected: 00)
# After reset, read from register 1: 00 (Expected: 00)
# After reset, read from register 2: 00 (Expected: 00)
# After reset, read from register 3: 00 (Expected: 00)
# ** Note: $stop    : C:/intelFPGA/codes/tb_reg.v(118)
#    Time: 200 ns  Iteration: 0  Instance: /tb_config_register_set
# Break in Module tb_config_register_set at C:/intelFPGA/codes/tb_reg.v line 118

VSIM 5>
```

Summary

The config_register_set module is a flexible, parameterized Verilog module for configuration management in digital designs. It supports reset, controlled writing, and real-time read access for efficient configuration storage, making it ideal for digital designs requiring a structured register storage solution.

====================================END====================================