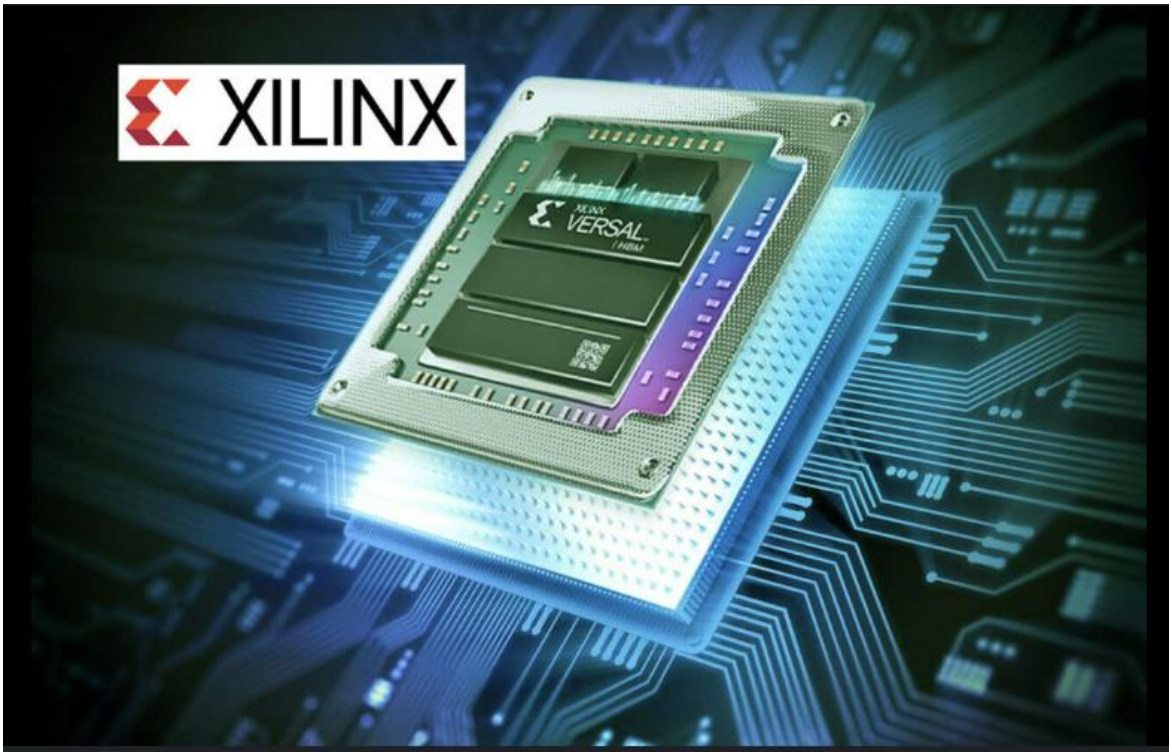
California State University, Northridge

College of Engineering & Computer Science

Electrical and Computer Engineering Department

ECE 526 Verilog and SystemVerilog Final Project Report

Daniel Celnik



Spring 2025

Instructor: Orod Haghighiara

I hereby attest that this lab report is entirely my own work. I have not copied either code or text from anyone, nor have I allowed or will I allow anyone to copy my work.

Name (printed) Daniel Glnik

Name (signed) [redacted] Date 5/10/25

INTRODUCTION

Double Data Rate Type 3 (DDR3) is a type of Synchronous Dynamic Random-Access Memory (SDRAM) found in consumer and industrial computers. Its notable features include high throughput in the Gb/s range, burst-oriented access, and a pipelined architecture allowing for simultaneous reads and writes. The goal of modeling DDR3 behavior in this report will be to understand various memory system design considerations such as latency, burst operations, and First-In First-Out (FIFO) queueing. This project additionally aims to prepare an ECE 526 student for field-programmable gate array (FPGA) and system-on-chip (SoC) integration via the realistic simulation of memory interfaces.

This project implements a behavioral DDR3 model simulating latency and burst responses. As the design process continued, I added iterative upgrades to functionality that are referred to as "versions" throughout this report. These include a burst-capable memory controller finite state machine (FSM) for read/write access, a register-based front-end interface to model an Advanced eXtensible Interface (AXI)-lite bus protocol style control scheme, and an automated stress testbench that verifies hundreds of read/writes. The project will support a 4-word bust length, latency measurement, pseudo-random pattern generation using XOR operators, CPU-style register access, and exhaustive pass/fail tracking. The end result is a modular, realistic memory system featuring a fully automated simulation with success/failure reporting that can be expanded to support AXI-lite or custom bus systems.

DESIGN METHODOLOGY AND TEST PLAN

The design is composed of the memory module ddr3_module, the memory interface/controller ddr3_interface, and the top module ddr3_full_system_tb that instantiates both and simulates the design. The corresponding code for version 1 can be found in Appendices A-C respectively. The ddr3_model and ddr3_interface feature the following input and output port configuration:

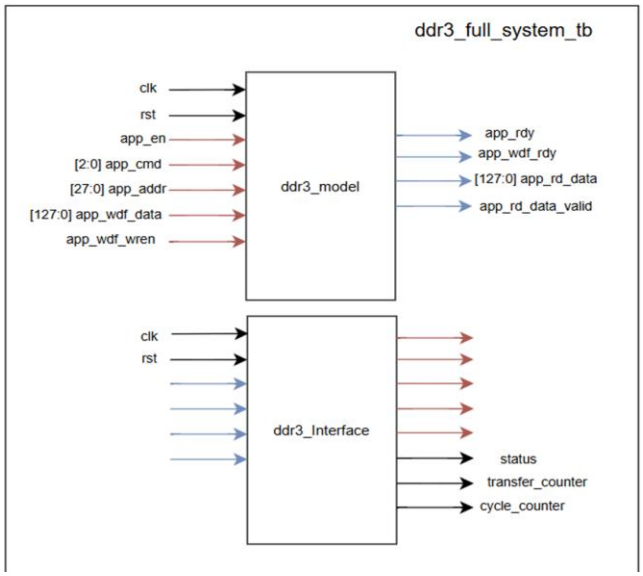


Figure 1 - A full schematic of Version 1

Note that ddr3_model and ddr3_interface are cross coupled; the interface generates control signals for the memory module whose outputs are fed back into the interface for data processing and comparison. The ddr3_module features reset, read, and write capabilities all enabled via control signals from ddr3_interface. Since this project is solely intended for simulation, the latency for each read cycle is hardcoded as a

parameter in ddr3_module.v and measured by the memory controller. The ddr3_interface features a cyclic FSM with the following states: IDLE, where the expected value is generated by XORing the address with a fixed value, WRITE, where the interface signals the ddr3_module to write data, WRITE_WAIT which waits a clock cycle for the written values to stabilize, READ, where the interface signals the ddr3_module to read data, READ_WAIT, where the interface waits for the ddr3_module to return the data given its inherent latency, and COMPARE, where the controller compares the read data with the expected data it calculated in the idle state.

In the next iteration, version 2, the system is updated with burst read/write support, enabling multiple writes and reads per command to model a realistic memory system. To implement this, the ddr3_model must be updated with a FIFO to handle the queuing of bursts. The 28-bit wide 16-element deep FIFO features a head pointing to the next read response to process, a tail pointing to where the next new read request will be written, and a counter to keep track of how many entries are currently in the queue to prevent overflow or underflow. When a read is issued by the controller, the address is saved to the FIFO, a countdown timer is set for that address, the FIFO's tail advances one entry, and the FIFO counter is incremented. Each clock cycle, the memory module checks if the FIFO's head has a nonzero timer. If it does, the module decrements the timer value, otherwise it returns the memory value at the current address and flags that the memory read is valid for one cycle before advancing the FIFO's head and decrimenting the FIFO's counter. A diagram of the FIFO is shown below for conceptualization purposes:

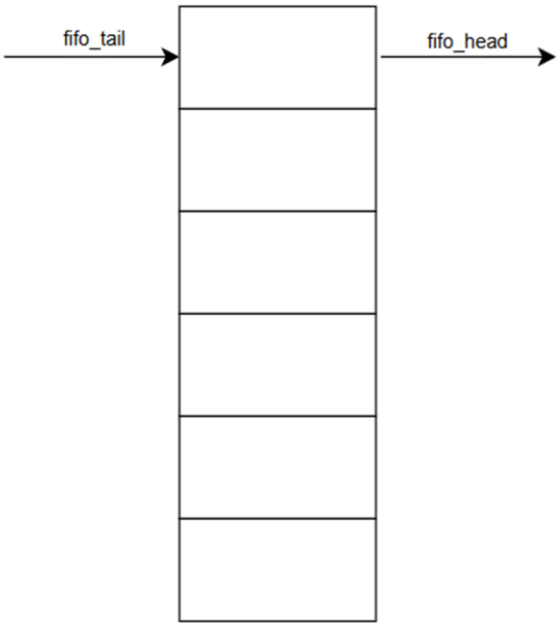


Figure 2 - A diagram of the FIFO featured in Version 2's ddr3_module

The ddr3_interface is also updated with new states reflecting the burst read/write behavior. The new states are entitled: IDLE, whose function has not changed, WRITE, which signals the ddr3_module to write to a single address, WRITE_BURST, which generates the next data values and signals the memory module to write them to the next parameterized number of addresses, WAIT_BEFORE_READ, which waits 3 cycles to ensure the memory module has time to accept the writes, READ, which reads the first signal in the burst, and READ_BURST, which reads from the remaining addresses in the burst and tracks the incoming valid flags. READ_BURST

additionally compares each read piece of data to the expected data calculated in the IDLE and WRITE_BURST states. If all the comparisons return true, the READ_BURST state will increment the base address and add latency to the cycle_counter before returning the FSM to its IDLE state. The total latency is then added at the end of the simulation and displayed in the command window. The code for version 2 can be found in Appendices E-G.

In version 3, we add a register-based front-end to model an AXI-lite style control scheme. No changes are made to the ddr3_module; only the interface and top module are updated to support read and write commands via two new registers in ddr3_interface_regctrl, read_trigger and write_trigger. The interface's FSM is additionally updated with the following states: idle, which is the same as the idle stage from version 2, issue_write where the corresponding control signals are enabled to write to a memory address, issue_write_burst where the interface finishes writing out the burst, wait_before_read where the system waits to ensure all written values have stabilized, issue_read, where the corresponding control signals are enabled to read from the first address in the burst, issue_read_burst where the interface finishes reading from the other addresses in the burst and compares the read values with the expected values calculated in the idle stage, and complete, where the testbench displays that the test is complete and returns to idle. Unlike the previous version where all states were iterated through once in order, this FSM has a branch in idle that determines whether the next state is issue_read or issue_write based on the values stored in read_trigger and write_trigger. The top module is then simplified; instead of manually setting control signals for each read and write, the testbench needs only to include assignments to read_trigger and write_trigger to trigger the system's behavior, simulating a CPU issuing read and write commands. Since we only need to verify the proper functionality of this new behavior, the testbench will be relatively lightweight covering just a single burst. The code for version 3 can be found in Appendices I-K.

Finally, version 4 modifies the testbench to include an exhaustive test iterating over 64 bursts corresponding to 256 addresses. The memory and interface need not be modified since the purpose of this version is to implement a stress test over many words; this can simply be implemented via a for loop starting at address 0x00 to 0xff incrementing by 4 each iteration. To ensure that the data written is unique to each address and more complex in nature than simply using the address index as the data value, I implemented the following data calculation: a 96-bit fixed value is XORed with the address being written to, while this is concatenated with the 32-bit address XORed with itself left shifted by 2 bits. This ensures that the data written to each address appears more random as opposed to simply incrementing the data value with the address being written to. The code for version 4's testbench can be found in Appendix M.

ANALYSIS OF RESULTS

In this section, we will iterate through each version's simulation output and waveform plot to verify their functionality. Starting with version 1, we check the output log found in Appendix D and observe that the testbench displays the write, read request, and read itself for addresses 0x00 through 0x1f before displaying that all transfers returned their expected value with a total latency of 256 cycles, or 8 cycles per address. Note that a latency cycle value of 5 is hardcoded into ddr3_module.v for simulation purposes only. The module takes 5 clock cycles to read the data and 3 additional cycles to move through the state machine for a total of 8 simulated cycles. For deployment on an FPGA in the real world, this would be calculated using an interrupt or timer. The corresponding waveform for the first 16 transactions can be seen below:

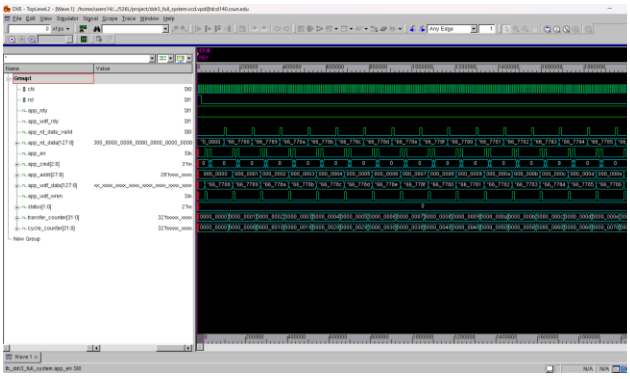


Figure 3 - Waveform data for writes and reads to the first 16 addresses in Version 1

The following explanation corresponds to a single address spanning 8 clock cycles. We observe the app enable (app_en) and app write data FIFO - write enable (app_wdf_wren) signals being driven high while the app command (app_cmd) signal is low to initiate the write. After a short delay, app_en and app_cmd are driven high to initiate the read; the read data valid flag app_rd_data_valid is then driven high for a cycle to indicate that the read was successful and returned the expected data.

For version 2, we check the log in Appendix H and observe that the testbench now writes to four addresses, reads from and compares the values stored in those four addresses, and advances to the next four addresses, repeating this process up to address 0x1f before displaying that all transfers returned their expected values with an average latency of 6.5 cycles as opposed to 8 cycles in version 1. This reduction in latency occurs since there are now less states to move through in the burst-pipelined FSM, however the first address's read still takes the full 5 cycles just as in version 1. The following waveform graph displays the first 16 of these transactions similar to the previous version:

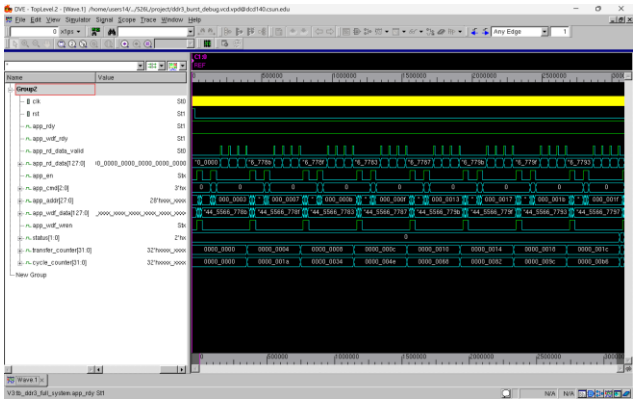


Figure 4 - Waveform data for burst-supported system in Version 2

We observe the cycle_counter increments by 0x1a (or decimal 26) each burst. Distributing the 26 cycles of latency among

the 4 transmitted words yields the calculated average latency of 6.5 cycles. At the start of the testbench, we observe app_en and app_wdf_wren asserted high while app_cmd is low to initiate the write sequence. We subsequently see the app address (app_addr) register increment quickly as it writes data for each word in the burst back-to-back before app_cmd and app_en are asserted high to signal the ddr3_module to read data at the specified addresses in app_addr. After a short delay, the app_rd_data_valid flag is raised for each successfully verified word. This process repeats until we reach the final address of 0x1f.

In version 3, we check the log in Appendix L and observe the relatively short testbench that simply writes a burst using the new registered front-end, reads it back, and compares it to the calculated data to verify transmission accuracy. Note that the interface was named ddr3_interface.v in previous versions, however its name has now been modified to ddr3_interface_regctrl.v to reflect the new registered front-end. The following waveforms correspond to the testbench for version 3:

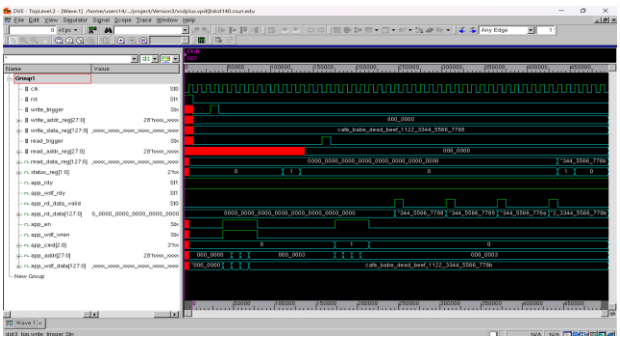


Figure 5 - Waveform data for the registered front-end system writing and reading a single word in Version 3

From the waveform plot, we observe the same write control signals from the previous versions change to signal the ddr3_model to initiate its writing sequence after the new register write_trigger is set high for a clock cycle. A similar process occurs for the read pipeline once read_trigger is set high for a clock cycle. Finally, we observe the app_rd_data_valid flag being asserted every 5 clock cycles as the value in the app read data (app_rd_data) changes simultaneously. This version features a new front-end that significantly reduces the complexity required to initiate read and write bursts, as the trigger registers branch the state machine accordingly. In the following final version, we attempt to utilize this simplified front-end and perform an exhaustive stress test.

Version 4 features no new additions to ddr3_model nor ddr3_interface_regctrl. We check Appendix N for the corresponding log file and observe that the new testbench indeed iterates over 256 total addresses (64 total bursts) with perfect accuracy. The following waveforms correspond to the complete simulation:

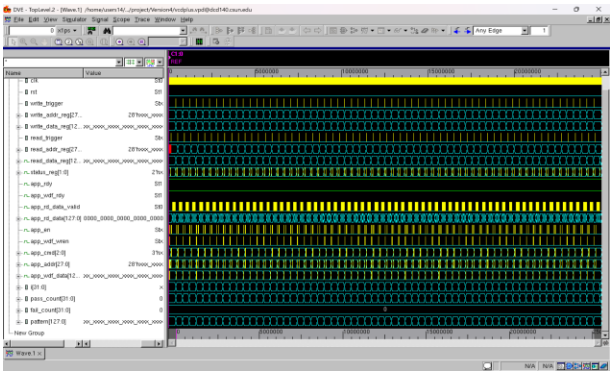


Figure 6 - Waveforms depicting the exhaustive 30-microsecond simulation iterating over 256 addresses in Version 4

The following waveforms correspond to a zoomed image of the first two burst transactions from version 4 for verification purposes:

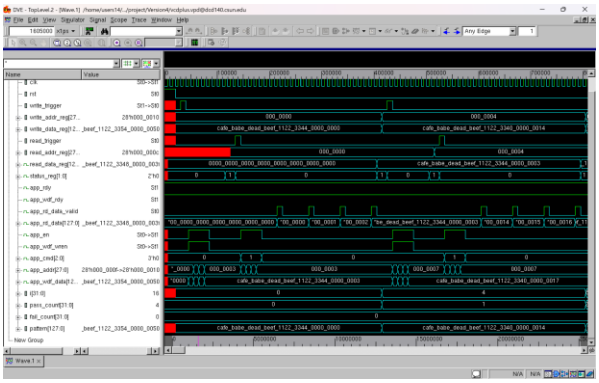


Figure 7 - Waveforms depicting the first two burst transactions in Version 4

We can now observe the same control signal expression in this version compared to version 3, albeit with another burst and the addition of registers i, pass_count, fail_count, and pattern. Register "i" holds the value of the first address in the current burst, pass_count holds the number of successful bursts, fail_count holds the number of failed bursts, while register "pattern" holds the value of the data to be written to the ddr3_module. We observe that between the first and second burst, the viewable app_wdf_data value is equal to the pattern value plus three. The first word written to the ddr3_module will have the value specified in "pattern" while the last word written will have the value specified in app_wdf_data. Note that when comparing the pattern between the first two bursts, the last 9 bits are visibly different as a result of the XOR operations and concatenations. The next waveform will depict the final two bursts from version 4's testbench:

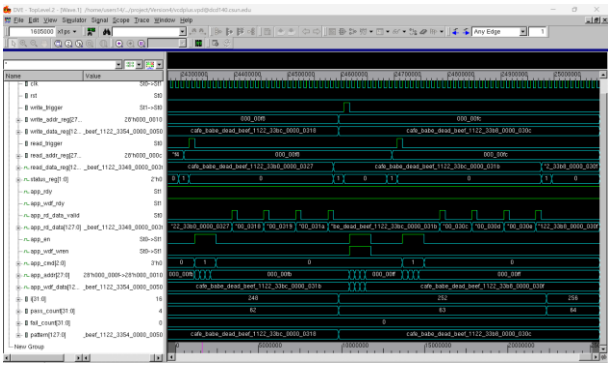


Figure 8 - Waveforms depicting the last two burst transactions in Version 4

From the above waveform plot, we observe pass_count reach 64, app_addr reach 0xff, and the variation in the last 9 bits of "pattern" once again. This verifies that our exhaustive testbench correctly wrote and read from all 256 addresses and concludes the results section of this report.

CONCLUSION

In this project, I was able to design a modular, expandable, and realistic memory system with a fully automated simulation testbench and success/failure reporting. We explored key features such as 4-word burst support, latency measurement, pseudo-random pattern generation using the XOR logical operator, CPU-style register access, and exhaustive pass/fail tracking. Each iteration of the project brought the model closer to a realistic memory system, concluding with a stress test to verify system functionality and stability. From the implementation of this project, I gained valuable experience designing FIFO systems, configuring and branching finite

state machines, and modularizing system design to support iterative upgrades

REFERENCES

[1] JEDEC Solid State Technology Association, "DDR3 SDRAM Specification (JESD79-3F)," 2010.
[2] ARM Ltd., "AMBA AXI and ACE Protocol Specification," 2011.
[3] P. P. Chu, FPGA Prototyping by Verilog Examples: Xilinx Spartan-3 Version, Wiley, 2008.
[4] Doulos Ltd., SystemVerilog for Design and Verification, training material, 2015.
[5] Xilinx, Memory Interface Solutions User Guide (UG086), 2019.

APPENDIX
Appendix A - Version 1 ddr3_model.v

```
// ddr3_model.v - Custom Behavioral DDR3 Simulation Model
`timescale 1ns / 1ps

module ddr3_model (
    input wire clk,
    input wire rst,

    input wire app_en,
    input wire [2:0] app_cmd,         // 3'b000: write, 3'b001: read
    input wire [27:0] app_addr,
    input wire [127:0] app_wdf_data,
    input wire app_wdf_wren,

    output reg app_rdy,
    output reg app_wdf_rdy,
    output reg [127:0] app_rd_data,
    output reg app_rd_data_valid
);

    // Simulated memory (128-bit wide, 512 locations)
    reg [127:0] mem_array [0:511];

    // Delay tracking
    parameter LATENCY_CYCLES = 5;
    reg [4:0] delay_counter;
    reg [127:0] read_buffer;
    reg [27:0] read_addr;
    reg pending_read;

    initial begin
        app_rdy = 1;
        app_wdf_rdy = 1;
        app_rd_data = 0;
        app_rd_data_valid = 0;
        delay_counter = 0;
        pending_read = 0;
    end
```

```
   always @(posedge clk) begin
      if (rst) begin
         app_rd_data_valid <= 0;
         delay_counter <= 0;
         pending_read <= 0;
      end else begin
         app_rd_data_valid <= 0;

         if (app_en && app_cmd == 3'b000 && app_wdf_wren) begin
            // WRITE
            mem_array[app_addr] <= app_wdf_data;
            $display("[DDR3_MODEL] WRITE to addr 0x%h = 0x%h", app_addr,
app_wdf_data);
         end
         else if (app_en && app_cmd == 3'b001) begin
            // READ: register request
            read_addr <= app_addr;
            pending_read <= 1;
            delay_counter <= 0;
            $display("[DDR3_MODEL] READ request for addr 0x%h", app_addr);
         end

         // READ pipeline
         if (pending_read) begin
            delay_counter <= delay_counter + 1;
            if (delay_counter == LATENCY_CYCLES) begin //simulates real-world delay
               app_rd_data <= mem_array[read_addr];
               app_rd_data_valid <= 1;
               $display("[DDR3_MODEL] READ response from addr 0x%h = 0x%h", read_addr,
mem_array[read_addr]);
               pending_read <= 0;
            end
         end
      end
   end
endmodule
```

Appendix B - Version 1 ddr3_interface.v

```
// ddr3_interface.v - Basic version (non-burst) with single-word read/write and verification
module ddr3_interface (
    input wire clk,
    input wire rst,

    input wire app_rdy,
    input wire app_wdf_rdy,
    input wire app_rd_data_valid,
    input wire [127:0] app_rd_data,

    output reg app_en,
    output reg [2:0] app_cmd,
    output reg [27:0] app_addr,
    output reg [127:0] app_wdf_data,
    output reg app_wdf_wren,

    output reg [1:0] status,
    output reg [31:0] transfer_counter,
    output reg [31:0] cycle_counter
);

    reg [2:0] state;
    parameter S_IDLE = 3'd0,
              S_WRITE = 3'd1,
              S_WRITE_WAIT = 3'd2,
              S_READ = 3'd3,
              S_READ_WAIT = 3'd4,
              S_COMPARE = 3'd5;

    parameter MAX_ADDR = 28'd32;

    reg [27:0] addr;
    reg [127:0] expected_data;
    reg [127:0] read_data;
    reg [31:0] latency_counter;
    reg measuring;

    always @(posedge clk) begin
```

```
if (rst) begin
    state <= S_IDLE;
    app_en <= 0;
    app_cmd <= 0;
    app_addr <= 0;
    app_wdf_data <= 0;
    app_wdf_wren <= 0;
    addr <= 0;
    transfer_counter <= 0;
    cycle_counter <= 0;
    latency_counter <= 0;
    measuring <= 0;
    status <= 0;
end else begin
    app_en <= 0;
    app_wdf_wren <= 0;

    case (state)
        S_IDLE: begin
            expected_data <= 128'hCAFEBABE_DEADBEEF_11223344_55667788 ^ addr;
            state <= S_WRITE;
        end

        S_WRITE: begin
            if (app_rdy && app_wdf_rdy) begin
                app_en <= 1;
                app_cmd <= 3'b000;
                app_addr <= addr;
                app_wdf_data <= expected_data;
                app_wdf_wren <= 1;
                state <= S_WRITE_WAIT;
            end
        end

        S_WRITE_WAIT: begin
            state <= S_READ;
        end

        S_READ: begin
            if (app_rdy) begin
```

```
                app_en <= 1;
                app_cmd <= 3'b001;
                app_addr <= addr;
                measuring <= 1;
                latency_counter <= 0;
                state <= S_READ_WAIT;
            end
        end

        S_READ_WAIT: begin
            if (measuring)
                latency_counter <= latency_counter + 1;
            if (app_rd_data_valid) begin
                read_data <= app_rd_data;
                measuring <= 0;
                state <= S_COMPARE;
            end
        end

        S_COMPARE: begin
            if (read_data == expected_data) begin
                transfer_counter <= transfer_counter + 1;
                cycle_counter <= cycle_counter + latency_counter;
                if (addr + 1 < MAX_ADDR) begin
                    addr <= addr + 1;
                    state <= S_IDLE;
                end else begin
                    status <= 2'b01;
                    state <= S_IDLE;
                end
            end else begin
                status <= 2'b10;
                state <= S_IDLE;
            end
        end
      endcase
    end
  end
endmodule
```

Appendix C - Version 1 tb_ddr3_full_system.v

```
// tb_ddr3_full_system.v - Full system testbench with controller and DDR model
`timescale 1ns / 1ps

module tb_ddr3_full_system();

    reg clk = 0;
    reg rst = 1;

    wire app_rdy;
    wire app_wdf_rdy;
    wire app_rd_data_valid;
    wire [127:0] app_rd_data;

    wire app_en;
    wire [2:0] app_cmd;
    wire [27:0] app_addr;
    wire [127:0] app_wdf_data;
    wire app_wdf_wren;

    wire [1:0] status;
    wire [31:0] transfer_counter;
    wire [31:0] cycle_counter;

    ddr3_interface ctrl (
        .clk(clk),
        .rst(rst),
        .app_rdy(app_rdy),
        .app_wdf_rdy(app_wdf_rdy),
        .app_rd_data_valid(app_rd_data_valid),
        .app_rd_data(app_rd_data),
        .app_en(app_en),
        .app_cmd(app_cmd),
        .app_addr(app_addr),
        .app_wdf_data(app_wdf_data),
        .app_wdf_wren(app_wdf_wren),
        .status(status),
        .transfer_counter(transfer_counter),
        .cycle_counter(cycle_counter)
```

```
);

ddr3_model mem (
    .clk(clk),
    .rst(rst),
    .app_en(app_en),
    .app_cmd(app_cmd),
    .app_addr(app_addr),
    .app_wdf_data(app_wdf_data),
    .app_wdf_wren(app_wdf_wren),
    .app_rdy(app_rdy),
    .app_wdf_rdy(app_wdf_rdy),
    .app_rd_data(app_rd_data),
    .app_rd_data_valid(app_rd_data_valid)
);

always #5 clk = ~clk;

initial begin
    $display("Starting full DDR3 system simulation...");
    $dumpfile("ddr3_full_system.vcd");
    $dumpvars(0, tb_ddr3_full_system);

    #20 rst = 0;

    wait (status != 2'b00);
    #10;

    if (status == 2'b01) begin
        $display("✅ TEST PASSED: All transfers correct.");
    end else if (status == 2'b10) begin
        $display("❌ TEST FAILED: Data mismatch detected.");
    end

    $display("Transfers: %0d, Total Latency: %0d cycles, Avg Latency: %.2f",
        transfer_counter, cycle_counter,
        (transfer_counter != 0) ? (cycle_counter * 1.0 / transfer_counter) : 0.0);

    #20 $finish;
end
```

```
endmodule
```

Appendix D - Version1.log

```
Command: /home/users14/dec71046/526L/project/./simv -l Version1.log
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP1_Full64; Runtime version U-2023.03-SP1_Full64;  Apr  9
11:06 2025
Starting full DDR3 system simulation...
[DDR3_MODEL] WRITE to addr 0x0000000 = 0xcafebabedeadbeef1122334455667788
[DDR3_MODEL] READ request for addr 0x0000000
[DDR3_MODEL] READ response from addr 0x0000000 =
0xcafebabedeadbeef1122334455667788
[DDR3_MODEL] WRITE to addr 0x0000001 = 0xcafebabedeadbeef1122334455667789
[DDR3_MODEL] READ request for addr 0x0000001
[DDR3_MODEL] READ response from addr 0x0000001 =
0xcafebabedeadbeef1122334455667789
[DDR3_MODEL] WRITE to addr 0x0000002 = 0xcafebabedeadbeef112233445566778a
[DDR3_MODEL] READ request for addr 0x0000002
[DDR3_MODEL] READ response from addr 0x0000002 =
0xcafebabedeadbeef112233445566778a
[DDR3_MODEL] WRITE to addr 0x0000003 = 0xcafebabedeadbeef112233445566778b
[DDR3_MODEL] READ request for addr 0x0000003
[DDR3_MODEL] READ response from addr 0x0000003 =
0xcafebabedeadbeef112233445566778b
[DDR3_MODEL] WRITE to addr 0x0000004 = 0xcafebabedeadbeef112233445566778c
[DDR3_MODEL] READ request for addr 0x0000004
[DDR3_MODEL] READ response from addr 0x0000004 =
0xcafebabedeadbeef112233445566778c
[DDR3_MODEL] WRITE to addr 0x0000005 = 0xcafebabedeadbeef112233445566778d
[DDR3_MODEL] READ request for addr 0x0000005
[DDR3_MODEL] READ response from addr 0x0000005 =
0xcafebabedeadbeef112233445566778d
[DDR3_MODEL] WRITE to addr 0x0000006 = 0xcafebabedeadbeef112233445566778e
[DDR3_MODEL] READ request for addr 0x0000006
[DDR3_MODEL] READ response from addr 0x0000006 =
0xcafebabedeadbeef112233445566778e
[DDR3_MODEL] WRITE to addr 0x0000007 = 0xcafebabedeadbeef112233445566778f
[DDR3_MODEL] READ request for addr 0x0000007
[DDR3_MODEL] READ response from addr 0x0000007 =
0xcafebabedeadbeef112233445566778f
```

[DDR3_MODEL] WRITE to addr 0x0000008 = 0xcafebabedeadbeef1122334455667780
[DDR3_MODEL] READ request for addr 0x0000008
[DDR3_MODEL] READ response from addr 0x0000008 =
0xcafebabedeadbeef1122334455667780
[DDR3_MODEL] WRITE to addr 0x0000009 = 0xcafebabedeadbeef1122334455667781
[DDR3_MODEL] READ request for addr 0x0000009
[DDR3_MODEL] READ response from addr 0x0000009 =
0xcafebabedeadbeef1122334455667781
[DDR3_MODEL] WRITE to addr 0x000000a = 0xcafebabedeadbeef1122334455667782
[DDR3_MODEL] READ request for addr 0x000000a
[DDR3_MODEL] READ response from addr 0x000000a =
0xcafebabedeadbeef1122334455667782
[DDR3_MODEL] WRITE to addr 0x000000b = 0xcafebabedeadbeef1122334455667783
[DDR3_MODEL] READ request for addr 0x000000b
[DDR3_MODEL] READ response from addr 0x000000b =
0xcafebabedeadbeef1122334455667783
[DDR3_MODEL] WRITE to addr 0x000000c = 0xcafebabedeadbeef1122334455667784
[DDR3_MODEL] READ request for addr 0x000000c
[DDR3_MODEL] READ response from addr 0x000000c =
0xcafebabedeadbeef1122334455667784
[DDR3_MODEL] WRITE to addr 0x000000d = 0xcafebabedeadbeef1122334455667785
[DDR3_MODEL] READ request for addr 0x000000d
[DDR3_MODEL] READ response from addr 0x000000d =
0xcafebabedeadbeef1122334455667785
[DDR3_MODEL] WRITE to addr 0x000000e = 0xcafebabedeadbeef1122334455667786
[DDR3_MODEL] READ request for addr 0x000000e
[DDR3_MODEL] READ response from addr 0x000000e =
0xcafebabedeadbeef1122334455667786
[DDR3_MODEL] WRITE to addr 0x000000f = 0xcafebabedeadbeef1122334455667787
[DDR3_MODEL] READ request for addr 0x000000f
[DDR3_MODEL] READ response from addr 0x000000f =
0xcafebabedeadbeef1122334455667787
[DDR3_MODEL] WRITE to addr 0x0000010 = 0xcafebabedeadbeef1122334455667798
[DDR3_MODEL] READ request for addr 0x0000010
[DDR3_MODEL] READ response from addr 0x0000010 =
0xcafebabedeadbeef1122334455667798
[DDR3_MODEL] WRITE to addr 0x0000011 = 0xcafebabedeadbeef1122334455667799
[DDR3_MODEL] READ request for addr 0x0000011
[DDR3_MODEL] READ response from addr 0x0000011 =
0xcafebabedeadbeef1122334455667799

[DDR3_MODEL] WRITE to addr 0x0000012 = 0xcafebabedeadbeef112233445566779a
[DDR3_MODEL] READ request for addr 0x0000012
[DDR3_MODEL] READ response from addr 0x0000012 =
0xcafebabedeadbeef112233445566779a
[DDR3_MODEL] WRITE to addr 0x0000013 = 0xcafebabedeadbeef112233445566779b
[DDR3_MODEL] READ request for addr 0x0000013
[DDR3_MODEL] READ response from addr 0x0000013 =
0xcafebabedeadbeef112233445566779b
[DDR3_MODEL] WRITE to addr 0x0000014 = 0xcafebabedeadbeef112233445566779c
[DDR3_MODEL] READ request for addr 0x0000014
[DDR3_MODEL] READ response from addr 0x0000014 =
0xcafebabedeadbeef112233445566779c
[DDR3_MODEL] WRITE to addr 0x0000015 = 0xcafebabedeadbeef112233445566779d
[DDR3_MODEL] READ request for addr 0x0000015
[DDR3_MODEL] READ response from addr 0x0000015 =
0xcafebabedeadbeef112233445566779d
[DDR3_MODEL] WRITE to addr 0x0000016 = 0xcafebabedeadbeef112233445566779e
[DDR3_MODEL] READ request for addr 0x0000016
[DDR3_MODEL] READ response from addr 0x0000016 =
0xcafebabedeadbeef112233445566779e
[DDR3_MODEL] WRITE to addr 0x0000017 = 0xcafebabedeadbeef112233445566779f
[DDR3_MODEL] READ request for addr 0x0000017
[DDR3_MODEL] READ response from addr 0x0000017 =
0xcafebabedeadbeef112233445566779f
[DDR3_MODEL] WRITE to addr 0x0000018 = 0xcafebabedeadbeef1122334455667790
[DDR3_MODEL] READ request for addr 0x0000018
[DDR3_MODEL] READ response from addr 0x0000018 =
0xcafebabedeadbeef1122334455667790
[DDR3_MODEL] WRITE to addr 0x0000019 = 0xcafebabedeadbeef1122334455667791
[DDR3_MODEL] READ request for addr 0x0000019
[DDR3_MODEL] READ response from addr 0x0000019 =
0xcafebabedeadbeef1122334455667791
[DDR3_MODEL] WRITE to addr 0x000001a = 0xcafebabedeadbeef1122334455667792
[DDR3_MODEL] READ request for addr 0x000001a
[DDR3_MODEL] READ response from addr 0x000001a =
0xcafebabedeadbeef1122334455667792
[DDR3_MODEL] WRITE to addr 0x000001b = 0xcafebabedeadbeef1122334455667793
[DDR3_MODEL] READ request for addr 0x000001b
[DDR3_MODEL] READ response from addr 0x000001b =
0xcafebabedeadbeef1122334455667793

[DDR3_MODEL] WRITE to addr 0x000001c = 0xcafebabedeadbeef1122334455667794
[DDR3_MODEL] READ request for addr 0x000001c
[DDR3_MODEL] READ response from addr 0x000001c =
0xcafebabedeadbeef1122334455667794
[DDR3_MODEL] WRITE to addr 0x000001d = 0xcafebabedeadbeef1122334455667795
[DDR3_MODEL] READ request for addr 0x000001d
[DDR3_MODEL] READ response from addr 0x000001d =
0xcafebabedeadbeef1122334455667795
[DDR3_MODEL] WRITE to addr 0x000001e = 0xcafebabedeadbeef1122334455667796
[DDR3_MODEL] READ request for addr 0x000001e
[DDR3_MODEL] READ response from addr 0x000001e =
0xcafebabedeadbeef1122334455667796
[DDR3_MODEL] WRITE to addr 0x000001f = 0xcafebabedeadbeef1122334455667797
[DDR3_MODEL] READ request for addr 0x000001f
[DDR3_MODEL] READ response from addr 0x000001f =
0xcafebabedeadbeef1122334455667797
✅ TEST PASSED: All transfers correct.
Transfers: 32, Total Latency: 256 cycles, Avg Latency: 8.00
$finish called from file "tb_ddr3_full_system.v", line 77.
$finish at simulation time 4205000
V C S S i m u l a t i o n R e p o r t
Time: 4205000 ps
CPU Time: 0.170 seconds; Data structure size: 0.0Mb
Wed Apr 9 11:06:23 2025

Appendix E - Version 2 ddr3_model.v

```
// ddr3_model.v - Final version with read FIFO queue and correct response timing
`timescale 1ns / 1ps

module ddr3_model (
    input wire clk,
    input wire rst,

    input wire app_en,
    input wire [2:0] app_cmd,
    input wire [27:0] app_addr,
    input wire [127:0] app_wdf_data,
    input wire app_wdf_wren,

    output reg app_rdy,
    output reg app_wdf_rdy,
    output reg [127:0] app_rd_data,
    output reg app_rd_data_valid
);

    reg [127:0] mem_array [0:1023];

    parameter LATENCY_CYCLES = 5;
    parameter FIFO_DEPTH = 16;

    // FIFO for read request queue
    reg [27:0] rd_addr_fifo [0:FIFO_DEPTH-1];
    reg [3:0] rd_timer_fifo [0:FIFO_DEPTH-1];  // Individual delay counters
    reg [3:0] fifo_head, fifo_tail;
    reg [3:0] fifo_count;

    integer i;

    initial begin
        app_rdy = 1;
        app_wdf_rdy = 1;
        app_rd_data = 0;
        app_rd_data_valid = 0;
        fifo_head = 0;
```

```
    fifo_tail = 0;
    fifo_count = 0;
    for (i = 0; i < FIFO_DEPTH; i = i + 1) begin
        rd_timer_fifo[i] = 0;
        rd_addr_fifo[i] = 0;
    end
end

always @(posedge clk) begin
    if (rst) begin
        app_rd_data_valid <= 0;
        app_rd_data <= 0;
        fifo_head <= 0;
        fifo_tail <= 0;
        fifo_count <= 0;
        for (i = 0; i < FIFO_DEPTH; i = i + 1) begin
            rd_timer_fifo[i] <= 0;
            rd_addr_fifo[i] <= 0;
        end
    end else begin
        app_rd_data_valid <= 0;

        // Handle write
        if (app_en && app_cmd == 3'b000 && app_wdf_wren) begin
            mem_array[app_addr] <= app_wdf_data;
        end

        // Queue read requests
        if (app_en && app_cmd == 3'b001 && fifo_count < FIFO_DEPTH) begin
            rd_addr_fifo[fifo_tail] <= app_addr;
            rd_timer_fifo[fifo_tail] <= LATENCY_CYCLES;
            fifo_tail <= fifo_tail + 1;
            fifo_count <= fifo_count + 1;
        end

        // Decrement timers and issue read response if ready
        if (fifo_count > 0) begin
            if (rd_timer_fifo[fifo_head] > 0) begin
                rd_timer_fifo[fifo_head] <= rd_timer_fifo[fifo_head] - 1;
            end else begin
```

```
                app_rd_data <= mem_array[rd_addr_fifo[fifo_head]];
                app_rd_data_valid <= 1;
                fifo_head <= fifo_head + 1;
                fifo_count <= fifo_count - 1;
            end
        end
    end
  end
endmodule
```

Appendix F - Version 2 ddr3_interface.v

```
// ddr3_interface.v - Clean version with polished output for logging
module ddr3_interface (
    input wire clk,
    input wire rst,

    input wire app_rdy,
    input wire app_wdf_rdy,
    input wire app_rd_data_valid,
    input wire [127:0] app_rd_data,

    output reg app_en,
    output reg [2:0] app_cmd,
    output reg [27:0] app_addr,
    output reg [127:0] app_wdf_data,
    output reg app_wdf_wren,

    output reg [1:0] status,
    output reg [31:0] transfer_counter,
    output reg [31:0] cycle_counter
);

    reg [2:0] state;
    parameter S_IDLE = 3'd0,
              S_WRITE = 3'd1,
              S_WRITE_BURST = 3'd2,
              S_WAIT_BEFORE_READ = 3'd3,
              S_READ = 3'd4,
              S_READ_BURST = 3'd5;

    parameter BURST_LENGTH = 4;
    parameter MAX_ADDR = 28'd32;

    reg [27:0] addr;
    reg [127:0] expected_data;
    reg [127:0] compare_expected;
    reg [31:0] latency_counter;
    reg measuring;
```

```
reg [2:0] burst_counter;
reg [2:0] read_req_counter;
reg [2:0] read_resp_counter;
reg [2:0] write_settle_counter;

reg [27:0] next_addr;
reg [127:0] next_data;

always @(posedge clk) begin
    if (rst) begin
        state <= S_IDLE;
        app_en <= 0;
        app_cmd <= 0;
        app_addr <= 0;
        app_wdf_data <= 0;
        app_wdf_wren <= 0;
        addr <= 0;
        transfer_counter <= 0;
        cycle_counter <= 0;
        latency_counter <= 0;
        measuring <= 0;
        status <= 0;
        burst_counter <= 0;
        read_req_counter <= 0;
        read_resp_counter <= 0;
        write_settle_counter <= 0;
    end else begin
        app_en <= 0;
        app_wdf_wren <= 0;
        app_cmd <= 3'b000;

        case (state)

            S_IDLE: begin
                burst_counter <= 0;
                read_req_counter <= 0;
                read_resp_counter <= 0;
                latency_counter <= 0;
                measuring <= 0;
                write_settle_counter <= 0;
```

```
    $display("\\n== New Burst @ Address 0x%08h ==", addr);
    expected_data <= 128'hCAFEBABE_DEADBEEF_11223344_55667788 ^ addr;
    state <= S_WRITE;
end

S_WRITE: begin
    if (app_rdy && app_wdf_rdy) begin
        app_en <= 1;
        app_cmd <= 3'b000;
        app_addr <= addr;
        app_wdf_data <= expected_data;
        app_wdf_wren <= 1;
        burst_counter <= 0;
        write_settle_counter <= 0;
        $display("  [WRITE] Addr: 0x%08h Data: 0x%032h", addr, expected_data);
        state <= S_WRITE_BURST;
    end
end

S_WRITE_BURST: begin
    if (burst_counter < BURST_LENGTH - 1 && app_rdy && app_wdf_rdy) begin
        next_addr = addr + burst_counter + 1;
        next_data = 128'hCAFEBABE_DEADBEEF_11223344_55667788 ^ next_addr;

        app_en <= 1;
        app_cmd <= 3'b000;
        app_addr <= next_addr;
        app_wdf_data <= next_data;
        app_wdf_wren <= 1;

        $display("  [WRITE] Addr: 0x%08h Data: 0x%032h", next_addr, next_data);

        burst_counter <= burst_counter + 1;
    end else if (burst_counter >= BURST_LENGTH - 1) begin
        state <= S_WAIT_BEFORE_READ;
    end
end

S_WAIT_BEFORE_READ: begin
```

```
            if (write_settle_counter < 3)
                write_settle_counter <= write_settle_counter + 1;
            else
                state <= S_READ;
        end

        S_READ: begin
            if (app_rdy) begin
                burst_counter <= 0;
                read_req_counter <= 0;
                read_resp_counter <= 0;
                measuring <= 1;
                latency_counter <= 0;
                $display("  [READ ] Starting burst read @ Address 0x%08h", addr);
                state <= S_READ_BURST;
            end
        end

        S_READ_BURST: begin
            if (measuring)
                latency_counter <= latency_counter + 1;

            if (app_rdy && read_req_counter < BURST_LENGTH) begin
                app_en <= 1;
                app_cmd <= 3'b001;
                app_addr <= addr + read_req_counter;
                read_req_counter <= read_req_counter + 1;
            end

            if (app_rd_data_valid) begin
                compare_expected = 128'hCAFEBABE_DEADBEEF_11223344_55667788 ^
(addr + read_resp_counter);

                if (app_rd_data == compare_expected) begin
                    $display("  [READ ] Addr: 0x%08h OK   Data: 0x%032h", addr +
read_resp_counter, app_rd_data);
                    read_resp_counter <= read_resp_counter + 1;
                    burst_counter <= burst_counter + 1;

                    if (burst_counter + 1 == BURST_LENGTH) begin
```

```
                        measuring <= 0;
                        transfer_counter <= transfer_counter + BURST_LENGTH;
                        cycle_counter <= cycle_counter + latency_counter;
                        addr <= addr + BURST_LENGTH;

                        if (addr + BURST_LENGTH < MAX_ADDR)
                           state <= S_IDLE;
                        else begin
                           status <= 2'b01;
                           state <= S_IDLE;
                        end
                     end
                  end else begin
                     $display("  [READ ] Addr: 0x%08h FAIL Data: 0x%032h (Expected:
0x%032h)", addr + read_resp_counter, app_rd_data, compare_expected);
                     measuring <= 0;
                     status <= 2'b10;
                     state <= S_IDLE;
                  end
               end
            end
         endcase
      end
   end
endmodule
```

Appendix G - Version 2 tb_ddr3_full_system.v

```
`timescale 1ns / 1ps

module tb_ddr3_full_system();

    reg clk = 0;
    reg rst = 1;

    wire app_rdy;
    wire app_wdf_rdy;
    wire app_rd_data_valid;
    wire [127:0] app_rd_data;

    wire app_en;
    wire [2:0] app_cmd;
    wire [27:0] app_addr;
    wire [127:0] app_wdf_data;
    wire app_wdf_wren;

    wire [1:0] status;
    wire [31:0] transfer_counter;
    wire [31:0] cycle_counter;

    ddr3_interface ctrl (
        .clk(clk),
        .rst(rst),
        .app_rdy(app_rdy),
        .app_wdf_rdy(app_wdf_rdy),
        .app_rd_data_valid(app_rd_data_valid),
        .app_rd_data(app_rd_data),
        .app_en(app_en),
        .app_cmd(app_cmd),
        .app_addr(app_addr),
        .app_wdf_data(app_wdf_data),
        .app_wdf_wren(app_wdf_wren),
        .status(status),
        .transfer_counter(transfer_counter),
        .cycle_counter(cycle_counter)
    );
```

```
    ddr3_model mem (
        .clk(clk),
        .rst(rst),
        .app_en(app_en),
        .app_cmd(app_cmd),
        .app_addr(app_addr),
        .app_wdf_data(app_wdf_data),
        .app_wdf_wren(app_wdf_wren),
        .app_rdy(app_rdy),
        .app_wdf_rdy(app_wdf_rdy),
        .app_rd_data(app_rd_data),
        .app_rd_data_valid(app_rd_data_valid)
    );

    always #5 clk = ~clk;

    initial begin
    $display("Starting full DDR3 system simulation with burst support...");
    $dumpfile("ddr3_burst_debug.vcd");
    $dumpvars(0, tb_ddr3_full_system);

    #20 rst = 0;

    while (status == 2'b00)
        @(posedge clk);
    #10;

    if (status == 2'b01) begin
        $display("✅ TEST PASSED: All transfers correct.");
    end else if (status == 2'b10) begin
        $display("❌ TEST FAILED: Data mismatch detected.");
    end

    $display("Transfers: %0d, Total Latency: %0d cycles, Avg Latency: %.2f",
        transfer_counter, cycle_counter,
        (transfer_counter != 0) ? (cycle_counter * 1.0 / transfer_counter) : 0.0);

    #20 $finish;
end
```

```
always @(posedge clk) begin
   if (app_en) begin
      if (app_cmd == 3'b000 && app_wdf_wren)
         $display("[WRITE] Addr: %h Data: %h", app_addr, app_wdf_data);
      else if (app_cmd == 3'b001)
         $display("[READ ] Addr: %h", app_addr);
   end
   if (app_rd_data_valid)
      $display("[RD_VALID] Data: %h", app_rd_data);
end

endmodule
```

Appendix H - Version2.log

```
Command: /home/users14/dec71046/526L/project/./simv -l Version2.log
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP1_Full64; Runtime version U-2023.03-SP1_Full64;  Apr  9
11:02 2025
Starting full DDR3 system simulation with burst support...
\n== New Burst @ Address 0x00000000 ==
  [WRITE] Addr: 0x00000000 Data: 0xcafebabedeadbeef1122334455667788
[WRITE] Addr: 0000000 Data: cafebabedeadbeef1122334455667788
  [WRITE] Addr: 0x00000001 Data: 0xcafebabedeadbeef1122334455667789
[WRITE] Addr: 0000001 Data: cafebabedeadbeef1122334455667789
  [WRITE] Addr: 0x00000002 Data: 0xcafebabedeadbeef112233445566778a
[WRITE] Addr: 0000002 Data: cafebabedeadbeef112233445566778a
  [WRITE] Addr: 0x00000003 Data: 0xcafebabedeadbeef112233445566778b
[WRITE] Addr: 0000003 Data: cafebabedeadbeef112233445566778b
  [READ ] Starting burst read @ Address 0x00000000
[READ ] Addr: 0000000
[READ ] Addr: 0000001
[READ ] Addr: 0000002
[READ ] Addr: 0000003
[RD_VALID] Data: cafebabedeadbeef1122334455667788
  [READ ] Addr: 0x00000000 OK   Data: 0xcafebabedeadbeef1122334455667788
[RD_VALID] Data: cafebabedeadbeef1122334455667789
  [READ ] Addr: 0x00000001 OK   Data: 0xcafebabedeadbeef1122334455667789
[RD_VALID] Data: cafebabedeadbeef112233445566778a
  [READ ] Addr: 0x00000002 OK   Data: 0xcafebabedeadbeef112233445566778a
[RD_VALID] Data: cafebabedeadbeef112233445566778b
  [READ ] Addr: 0x00000003 OK   Data: 0xcafebabedeadbeef112233445566778b
\n== New Burst @ Address 0x00000004 ==
  [WRITE] Addr: 0x00000004 Data: 0xcafebabedeadbeef112233445566778c
[WRITE] Addr: 0000004 Data: cafebabedeadbeef112233445566778c
  [WRITE] Addr: 0x00000005 Data: 0xcafebabedeadbeef112233445566778d
[WRITE] Addr: 0000005 Data: cafebabedeadbeef112233445566778d
  [WRITE] Addr: 0x00000006 Data: 0xcafebabedeadbeef112233445566778e
[WRITE] Addr: 0000006 Data: cafebabedeadbeef112233445566778e
  [WRITE] Addr: 0x00000007 Data: 0xcafebabedeadbeef112233445566778f
[WRITE] Addr: 0000007 Data: cafebabedeadbeef112233445566778f
  [READ ] Starting burst read @ Address 0x00000004
```

[READ] Addr: 0000004
[READ] Addr: 0000005
[READ] Addr: 0000006
[READ] Addr: 0000007
[RD_VALID] Data: cafebabedeadbeef112233445566778c
  [READ] Addr: 0x00000004 OK   Data: 0xcafebabedeadbeef112233445566778c
[RD_VALID] Data: cafebabedeadbeef112233445566778d
  [READ] Addr: 0x00000005 OK   Data: 0xcafebabedeadbeef112233445566778d
[RD_VALID] Data: cafebabedeadbeef112233445566778e
  [READ] Addr: 0x00000006 OK   Data: 0xcafebabedeadbeef112233445566778e
[RD_VALID] Data: cafebabedeadbeef112233445566778f
  [READ] Addr: 0x00000007 OK   Data: 0xcafebabedeadbeef112233445566778f
\n== New Burst @ Address 0x00000008 ==
  [WRITE] Addr: 0x00000008 Data: 0xcafebabedeadbeef1122334455667780
[WRITE] Addr: 0000008 Data: cafebabedeadbeef1122334455667780
  [WRITE] Addr: 0x00000009 Data: 0xcafebabedeadbeef1122334455667781
[WRITE] Addr: 0000009 Data: cafebabedeadbeef1122334455667781
  [WRITE] Addr: 0x0000000a Data: 0xcafebabedeadbeef1122334455667782
[WRITE] Addr: 000000a Data: cafebabedeadbeef1122334455667782
  [WRITE] Addr: 0x0000000b Data: 0xcafebabedeadbeef1122334455667783
[WRITE] Addr: 000000b Data: cafebabedeadbeef1122334455667783
  [READ] Starting burst read @ Address 0x00000008
[READ] Addr: 0000008
[READ] Addr: 0000009
[READ] Addr: 000000a
[READ] Addr: 000000b
[RD_VALID] Data: cafebabedeadbeef1122334455667780
  [READ] Addr: 0x00000008 OK   Data: 0xcafebabedeadbeef1122334455667780
[RD_VALID] Data: cafebabedeadbeef1122334455667781
  [READ] Addr: 0x00000009 OK   Data: 0xcafebabedeadbeef1122334455667781
[RD_VALID] Data: cafebabedeadbeef1122334455667782
  [READ] Addr: 0x0000000a OK   Data: 0xcafebabedeadbeef1122334455667782
[RD_VALID] Data: cafebabedeadbeef1122334455667783
  [READ] Addr: 0x0000000b OK   Data: 0xcafebabedeadbeef1122334455667783
\n== New Burst @ Address 0x0000000c ==
  [WRITE] Addr: 0x0000000c Data: 0xcafebabedeadbeef1122334455667784
[WRITE] Addr: 000000c Data: cafebabedeadbeef1122334455667784
  [WRITE] Addr: 0x0000000d Data: 0xcafebabedeadbeef1122334455667785
[WRITE] Addr: 000000d Data: cafebabedeadbeef1122334455667785
  [WRITE] Addr: 0x0000000e Data: 0xcafebabedeadbeef1122334455667786

[WRITE] Addr: 000000e Data: cafebabedeadbeef1122334455667786
  [WRITE] Addr: 0x0000000f Data: 0xcafebabedeadbeef1122334455667787
[WRITE] Addr: 000000f Data: cafebabedeadbeef1122334455667787
  [READ] Starting burst read @ Address 0x0000000c
[READ] Addr: 000000c
[READ] Addr: 000000d
[READ] Addr: 000000e
[READ] Addr: 000000f
[RD_VALID] Data: cafebabedeadbeef1122334455667784
  [READ] Addr: 0x0000000c OK   Data: 0xcafebabedeadbeef1122334455667784
[RD_VALID] Data: cafebabedeadbeef1122334455667785
  [READ] Addr: 0x0000000d OK   Data: 0xcafebabedeadbeef1122334455667785
[RD_VALID] Data: cafebabedeadbeef1122334455667786
  [READ] Addr: 0x0000000e OK   Data: 0xcafebabedeadbeef1122334455667786
[RD_VALID] Data: cafebabedeadbeef1122334455667787
  [READ] Addr: 0x0000000f OK   Data: 0xcafebabedeadbeef1122334455667787
\n== New Burst @ Address 0x00000010 ==
  [WRITE] Addr: 0x00000010 Data: 0xcafebabedeadbeef1122334455667798
[WRITE] Addr: 0000010 Data: cafebabedeadbeef1122334455667798
  [WRITE] Addr: 0x00000011 Data: 0xcafebabedeadbeef1122334455667799
[WRITE] Addr: 0000011 Data: cafebabedeadbeef1122334455667799
  [WRITE] Addr: 0x00000012 Data: 0xcafebabedeadbeef112233445566779a
[WRITE] Addr: 0000012 Data: cafebabedeadbeef112233445566779a
  [WRITE] Addr: 0x00000013 Data: 0xcafebabedeadbeef112233445566779b
[WRITE] Addr: 0000013 Data: cafebabedeadbeef112233445566779b
  [READ] Starting burst read @ Address 0x00000010
[READ] Addr: 0000010
[READ] Addr: 0000011
[READ] Addr: 0000012
[READ] Addr: 0000013
[RD_VALID] Data: cafebabedeadbeef1122334455667798
  [READ] Addr: 0x00000010 OK   Data: 0xcafebabedeadbeef1122334455667798
[RD_VALID] Data: cafebabedeadbeef1122334455667799
  [READ] Addr: 0x00000011 OK   Data: 0xcafebabedeadbeef1122334455667799
[RD_VALID] Data: cafebabedeadbeef112233445566779a
  [READ] Addr: 0x00000012 OK   Data: 0xcafebabedeadbeef112233445566779a
[RD_VALID] Data: cafebabedeadbeef112233445566779b
  [READ] Addr: 0x00000013 OK   Data: 0xcafebabedeadbeef112233445566779b
\n== New Burst @ Address 0x00000014 ==
  [WRITE] Addr: 0x00000014 Data: 0xcafebabedeadbeef112233445566779c

```
[WRITE] Addr: 0000014 Data: cafebabedeadbeef112233445566779c
  [WRITE] Addr: 0x00000015 Data: 0xcafebabedeadbeef112233445566779d
[WRITE] Addr: 0000015 Data: cafebabedeadbeef112233445566779d
  [WRITE] Addr: 0x00000016 Data: 0xcafebabedeadbeef112233445566779e
[WRITE] Addr: 0000016 Data: cafebabedeadbeef112233445566779e
  [WRITE] Addr: 0x00000017 Data: 0xcafebabedeadbeef112233445566779f
[WRITE] Addr: 0000017 Data: cafebabedeadbeef112233445566779f
  [READ ] Starting burst read @ Address 0x00000014
[READ ] Addr: 0000014
[READ ] Addr: 0000015
[READ ] Addr: 0000016
[READ ] Addr: 0000017
[RD_VALID] Data: cafebabedeadbeef112233445566779c
  [READ ] Addr: 0x00000014 OK   Data: 0xcafebabedeadbeef112233445566779c
[RD_VALID] Data: cafebabedeadbeef112233445566779d
  [READ ] Addr: 0x00000015 OK   Data: 0xcafebabedeadbeef112233445566779d
[RD_VALID] Data: cafebabedeadbeef112233445566779e
  [READ ] Addr: 0x00000016 OK   Data: 0xcafebabedeadbeef112233445566779e
[RD_VALID] Data: cafebabedeadbeef112233445566779f
  [READ ] Addr: 0x00000017 OK   Data: 0xcafebabedeadbeef112233445566779f
\n== New Burst @ Address 0x00000018 ==
  [WRITE] Addr: 0x00000018 Data: 0xcafebabedeadbeef1122334455667790
[WRITE] Addr: 0000018 Data: cafebabedeadbeef1122334455667790
  [WRITE] Addr: 0x00000019 Data: 0xcafebabedeadbeef1122334455667791
[WRITE] Addr: 0000019 Data: cafebabedeadbeef1122334455667791
  [WRITE] Addr: 0x0000001a Data: 0xcafebabedeadbeef1122334455667792
[WRITE] Addr: 000001a Data: cafebabedeadbeef1122334455667792
  [WRITE] Addr: 0x0000001b Data: 0xcafebabedeadbeef1122334455667793
[WRITE] Addr: 000001b Data: cafebabedeadbeef1122334455667793
  [READ ] Starting burst read @ Address 0x00000018
[READ ] Addr: 0000018
[READ ] Addr: 0000019
[READ ] Addr: 000001a
[READ ] Addr: 000001b
[RD_VALID] Data: cafebabedeadbeef1122334455667790
  [READ ] Addr: 0x00000018 OK   Data: 0xcafebabedeadbeef1122334455667790
[RD_VALID] Data: cafebabedeadbeef1122334455667791
  [READ ] Addr: 0x00000019 OK   Data: 0xcafebabedeadbeef1122334455667791
[RD_VALID] Data: cafebabedeadbeef1122334455667792
  [READ ] Addr: 0x0000001a OK   Data: 0xcafebabedeadbeef1122334455667792
```

```
[RD_VALID] Data: cafebabedeadbeef1122334455667793
  [READ ] Addr: 0x0000001b OK   Data: 0xcafebabedeadbeef1122334455667793
\n== New Burst @ Address 0x0000001c ==
  [WRITE] Addr: 0x0000001c Data: 0xcafebabedeadbeef1122334455667794
[WRITE] Addr: 000001c Data: cafebabedeadbeef1122334455667794
  [WRITE] Addr: 0x0000001d Data: 0xcafebabedeadbeef1122334455667795
[WRITE] Addr: 000001d Data: cafebabedeadbeef1122334455667795
  [WRITE] Addr: 0x0000001e Data: 0xcafebabedeadbeef1122334455667796
[WRITE] Addr: 000001e Data: cafebabedeadbeef1122334455667796
  [WRITE] Addr: 0x0000001f Data: 0xcafebabedeadbeef1122334455667797
[WRITE] Addr: 000001f Data: cafebabedeadbeef1122334455667797
  [READ ] Starting burst read @ Address 0x0000001c
[READ ] Addr: 000001c
[READ ] Addr: 000001d
[READ ] Addr: 000001e
[READ ] Addr: 000001f
[RD_VALID] Data: cafebabedeadbeef1122334455667794
  [READ ] Addr: 0x0000001c OK   Data: 0xcafebabedeadbeef1122334455667794
[RD_VALID] Data: cafebabedeadbeef1122334455667795
  [READ ] Addr: 0x0000001d OK   Data: 0xcafebabedeadbeef1122334455667795
[RD_VALID] Data: cafebabedeadbeef1122334455667796
  [READ ] Addr: 0x0000001e OK   Data: 0xcafebabedeadbeef1122334455667796
[RD_VALID] Data: cafebabedeadbeef1122334455667797
  [READ ] Addr: 0x0000001f OK   Data: 0xcafebabedeadbeef1122334455667797
\n== New Burst @ Address 0x00000020 ==
✅ TEST PASSED: All transfers correct.
Transfers: 32, Total Latency: 208 cycles, Avg Latency: 6.50
  [WRITE] Addr: 0x00000020 Data: 0xcafebabedeadbeef11223344556677a8
[WRITE] Addr: 0000020 Data: cafebabedeadbeef11223344556677a8
  [WRITE] Addr: 0x00000021 Data: 0xcafebabedeadbeef11223344556677a9
$finish called from file "tb_ddr3_full_system.v", line 78.
$finish at simulation time             3095000
           V C S   S i m u l a t i o n   R e p o r t
Time: 3095000 ps
CPU Time:      0.180 seconds;       Data structure size:   0.0Mb
Wed Apr  9 11:02:44 2025
```

Appendix I - Version 3 ddr3_model.v

```
`timescale 1ns / 1ps

module ddr3_model (
    input wire clk,
    input wire rst,

    input wire app_en,
    input wire [2:0] app_cmd,
    input wire [27:0] app_addr,
    input wire [127:0] app_wdf_data,
    input wire app_wdf_wren,

    output reg app_rdy,
    output reg app_wdf_rdy,
    output reg [127:0] app_rd_data,
    output reg app_rd_data_valid
);

    reg [127:0] mem_array [0:1023];

    parameter LATENCY_CYCLES = 5;
    parameter FIFO_DEPTH = 16;

    reg [27:0] rd_addr_fifo [0:FIFO_DEPTH-1];
    reg [3:0] rd_timer_fifo [0:FIFO_DEPTH-1];
    reg [3:0] fifo_head, fifo_tail;
    reg [3:0] fifo_count;

    integer i;

    initial begin
        app_rdy = 1;
        app_wdf_rdy = 1;
        app_rd_data = 0;
        app_rd_data_valid = 0;
        fifo_head = 0;
        fifo_tail = 0;
        fifo_count = 0;
```

```
    for (i = 0; i < FIFO_DEPTH; i = i + 1) begin
        rd_timer_fifo[i] = 0;
        rd_addr_fifo[i] = 0;
    end
end

always @(posedge clk) begin
    if (rst) begin
        app_rd_data_valid <= 0;
        app_rd_data <= 0;
        fifo_head <= 0;
        fifo_tail <= 0;
        fifo_count <= 0;
        for (i = 0; i < FIFO_DEPTH; i = i + 1) begin
            rd_timer_fifo[i] <= 0;
            rd_addr_fifo[i] <= 0;
        end
    end else begin
        app_rd_data_valid <= 0;

        if (app_en && app_cmd == 3'b000 && app_wdf_wren) begin
            mem_array[app_addr] <= app_wdf_data;
        end

        if (app_en && app_cmd == 3'b001 && fifo_count < FIFO_DEPTH) begin
            rd_addr_fifo[fifo_tail] <= app_addr;
            rd_timer_fifo[fifo_tail] <= LATENCY_CYCLES;
            fifo_tail <= fifo_tail + 1;
            fifo_count <= fifo_count + 1;
        end

        if (fifo_count > 0) begin
            if (rd_timer_fifo[fifo_head] > 0) begin
                rd_timer_fifo[fifo_head] <= rd_timer_fifo[fifo_head] - 1;
            end else begin
                app_rd_data <= mem_array[rd_addr_fifo[fifo_head]];
                app_rd_data_valid <= 1;
                fifo_head <= fifo_head + 1;
                fifo_count <= fifo_count - 1;
            end
```

```
            end
        end
    end
endmodule
```

Appendix J - Version 3 ddr3_interface_regctrl.v

```
// ddr3_interface_regctrl.v - Register-based burst controller with debug logging
module ddr3_interface_regctrl (
    input wire clk,
    input wire rst,

    input wire write_trigger,
    input wire [27:0] write_addr,
    input wire [127:0] write_data,

    input wire read_trigger,
    input wire [27:0] read_addr,
    output reg [127:0] read_data,
    output reg [1:0] status,

    input wire app_rdy,
    input wire app_wdf_rdy,
    input wire app_rd_data_valid,
    input wire [127:0] app_rd_data,

    output reg app_en,
    output reg [2:0] app_cmd,
    output reg [27:0] app_addr,
    output reg [127:0] app_wdf_data,
    output reg app_wdf_wren
);

    reg [2:0] state;
    parameter IDLE = 3'd0,
              ISSUE_WRITE = 3'd1,
              ISSUE_WRITE_BURST = 3'd2,
              WAIT_BEFORE_READ = 3'd3,
              ISSUE_READ = 3'd4,
              ISSUE_READ_BURST = 3'd5,
              COMPLETE = 3'd6;

    parameter BURST_LENGTH = 4;

    reg [2:0] burst_counter;
```

```
    reg [27:0] base_addr;
    reg [127:0] base_data;
    reg [127:0] expected_data;
    reg [127:0] compare_expected;
    reg [2:0] read_req_counter;
    reg [2:0] read_resp_counter;
    reg [2:0] settle_delay;

    always @(posedge clk) begin
        if (rst) begin
            state <= IDLE;
            app_en <= 0;
            app_cmd <= 0;
            app_addr <= 0;
            app_wdf_data <= 0;
            app_wdf_wren <= 0;
            read_data <= 0;
            status <= 2'b00;
            burst_counter <= 0;
            read_req_counter <= 0;
            read_resp_counter <= 0;
            base_addr <= 0;
            base_data <= 0;
            settle_delay <= 0;
        end else begin
            app_en <= 0;
            app_wdf_wren <= 0;
            app_cmd <= 3'b000;

            case (state)
                IDLE: begin
                    status <= 2'b00;
                    if (write_trigger) begin
                        base_addr <= write_addr;
                        base_data <= write_data;
                        burst_counter <= 0;
                        $display("[REGCTRL] Write trigger received. Starting burst write at addr
0x%08h", write_addr);
                        state <= ISSUE_WRITE;
                    end else if (read_trigger) begin
```

```
                base_addr <= read_addr;
                burst_counter <= 0;
                $display("[REGCTRL] Read trigger received. Starting burst read at addr
0x%08h", read_addr);
                state <= ISSUE_READ;
            end
        end

        ISSUE_WRITE: begin
            if (app_rdy && app_wdf_rdy) begin
                app_en <= 1;
                app_cmd <= 3'b000;
                app_addr <= base_addr;
                app_wdf_data <= base_data;
                app_wdf_wren <= 1;
                burst_counter <= 1;
                $display("[REGCTRL] WRITE 0x%08h = 0x%032h", base_addr, base_data);
                state <= ISSUE_WRITE_BURST;
            end
        end

        ISSUE_WRITE_BURST: begin
            if (burst_counter < BURST_LENGTH && app_rdy && app_wdf_rdy) begin
                app_en <= 1;
                app_cmd <= 3'b000;
                app_addr <= base_addr + burst_counter;
                app_wdf_data <= base_data ^ burst_counter;
                app_wdf_wren <= 1;
                $display("[REGCTRL] WRITE 0x%08h = 0x%032h", base_addr +
burst_counter, base_data ^ burst_counter);
                burst_counter <= burst_counter + 1;
                if (burst_counter == BURST_LENGTH - 1)
                    state <= WAIT_BEFORE_READ;
            end
        end

        WAIT_BEFORE_READ: begin
            settle_delay <= settle_delay + 1;
            if (settle_delay == 3'd3) begin
                settle_delay <= 0;
```

```
                state <= COMPLETE;
                $display("[REGCTRL] Write burst complete.");
                status <= 2'b01;
            end
        end

        ISSUE_READ: begin
            if (app_rdy) begin
                app_en <= 1;
                app_cmd <= 3'b001;
                app_addr <= base_addr;
                read_req_counter <= 1;
                read_resp_counter <= 0;
                burst_counter <= 1;
                $display("[REGCTRL] READ request for addr 0x%08h", base_addr);
                state <= ISSUE_READ_BURST;
            end
        end

        ISSUE_READ_BURST: begin
            if (read_req_counter < BURST_LENGTH && app_rdy) begin
                app_en <= 1;
                app_cmd <= 3'b001;
                app_addr <= base_addr + read_req_counter;
                $display("[REGCTRL] READ request for addr 0x%08h", base_addr +
read_req_counter);
                read_req_counter <= read_req_counter + 1;
            end
            if (app_rd_data_valid) begin
                compare_expected = base_data ^ read_resp_counter;
                if (app_rd_data == compare_expected) begin
                    $display("[REGCTRL] READ OK   Addr: 0x%08h Data: 0x%032h",
base_addr + read_resp_counter, app_rd_data);
                    read_resp_counter <= read_resp_counter + 1;
                    if (read_resp_counter == BURST_LENGTH - 1) begin
                        read_data <= app_rd_data;
                        status <= 2'b01;
                        $display("[REGCTRL] Burst read complete and verified.");
                        state <= COMPLETE;
                    end
```

```
                end else begin
                    $display("[REGCTRL] READ FAIL Addr: 0x%08h Data: 0x%032h Expected:
0x%032h",
                            base_addr + read_resp_counter, app_rd_data, compare_expected);
                    status <= 2'b10;
                    state <= COMPLETE;
                end
            end
        end

        COMPLETE: begin
            $display("[REGCTRL] Returning to IDLE.");
            state <= IDLE;
        end
      endcase
    end
  end
endmodule
```

Appendix K - Version 3 ddr3_top.v

```
// ddr3_top.v - Top-level integration with register-based front-end
`timescale 1ns / 1ps

module ddr3_top();
    reg clk = 0;
    reg rst = 1;

    // Interface registers
    reg        write_trigger;
    reg [27:0] write_addr_reg;
    reg [127:0] write_data_reg;

    reg        read_trigger;
    reg [27:0] read_addr_reg;
    wire [127:0] read_data_reg;
    wire [1:0] status_reg;

    wire app_rdy, app_wdf_rdy, app_rd_data_valid;
    wire [127:0] app_rd_data;
    wire app_en, app_wdf_wren;
    wire [2:0] app_cmd;
    wire [27:0] app_addr;
    wire [127:0] app_wdf_data;

    // Instantiate controller
    ddr3_interface_regctrl controller (
        .clk(clk),
        .rst(rst),
        .write_trigger(write_trigger),
        .write_addr(write_addr_reg),
        .write_data(write_data_reg),
        .read_trigger(read_trigger),
        .read_addr(read_addr_reg),
        .read_data(read_data_reg),
        .status(status_reg),
        .app_rdy(app_rdy),
        .app_wdf_rdy(app_wdf_rdy),
        .app_rd_data_valid(app_rd_data_valid),
```

```
    .app_rd_data(app_rd_data),
    .app_en(app_en),
    .app_cmd(app_cmd),
    .app_addr(app_addr),
    .app_wdf_data(app_wdf_data),
    .app_wdf_wren(app_wdf_wren)
);

// Instantiate memory model
ddr3_model mem (
    .clk(clk),
    .rst(rst),
    .app_en(app_en),
    .app_cmd(app_cmd),
    .app_addr(app_addr),
    .app_wdf_data(app_wdf_data),
    .app_wdf_wren(app_wdf_wren),
    .app_rdy(app_rdy),
    .app_wdf_rdy(app_wdf_rdy),
    .app_rd_data(app_rd_data),
    .app_rd_data_valid(app_rd_data_valid)
);

// Simulate CPU register writes
initial begin
    $display("Starting full DDR3 register interface test...");
    #10 rst = 0;
    write_trigger = 0;
    read_trigger = 0;

    // Burst write to address 0
    write_addr_reg = 28'd0;
    write_data_reg = 128'hCAFEBABE_DEADBEEF_11223344_55667788;
    #20 write_trigger = 1;
    #10 write_trigger = 0;

    // Wait before burst read
    #100;
    read_addr_reg = 28'd0;
    #20 read_trigger = 1;
```

```
        #10 read_trigger = 0;

        // Wait for read to finish
        wait (status_reg != 2'b00);
        #10;
        $display("Final Read Result: %h", read_data_reg);
        $display("Status: %b", status_reg);

        #50 $finish;
    end

    always #5 clk = ~clk;
endmodule
```

Appendix L - Version3.log

```
Command: /home/users14/dec71046/526L/project/Version3/./simv -l Version3.log
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP1_Full64; Runtime version U-2023.03-SP1_Full64;  Apr 16
19:23 2025
Starting full DDR3 register interface test...
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000000
[REGCTRL] WRITE 0x00000000 = 0xcafebabedeadbeef1122334455667788
[REGCTRL] WRITE 0x00000001 = 0xcafebabedeadbeef1122334455667789
[REGCTRL] WRITE 0x00000002 = 0xcafebabedeadbeef112233445566778a
[REGCTRL] WRITE 0x00000003 = 0xcafebabedeadbeef112233445566778b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000000
[REGCTRL] READ request for addr 0x00000000
[REGCTRL] READ request for addr 0x00000001
[REGCTRL] READ request for addr 0x00000002
[REGCTRL] READ request for addr 0x00000003
[REGCTRL] READ OK   Addr: 0x00000000 Data: 0xcafebabedeadbeef1122334455667788
[REGCTRL] READ OK   Addr: 0x00000001 Data: 0xcafebabedeadbeef1122334455667789
[REGCTRL] READ OK   Addr: 0x00000002 Data: 0xcafebabedeadbeef112233445566778a
[REGCTRL] READ OK   Addr: 0x00000003 Data: 0xcafebabedeadbeef112233445566778b
[REGCTRL] Burst read complete and verified.
Final Read Result: cafebabedeadbeef112233445566778b
Status: 01
[REGCTRL] Returning to IDLE.
$finish called from file "ddr3_top.v", line 87.
$finish at simulation time              495000
           V C S   S i m u l a t i o n   R e p o r t
Time: 495000 ps
CPU Time:      0.170 seconds;       Data structure size:   0.0Mb
Wed Apr 16 19:23:47 2025
```

Appendix M - Version 4 ddr3_top.v

```
// ddr3_top.v - Register-based DDR3 testbench with stress test loop
`timescale 1ns / 1ps

module ddr3_top();
    reg clk = 0;
    reg rst = 1;

    reg        write_trigger;
    reg [27:0] write_addr_reg;
    reg [127:0] write_data_reg;

    reg        read_trigger;
    reg [27:0] read_addr_reg;
    wire [127:0] read_data_reg;
    wire [1:0] status_reg;

    wire app_rdy, app_wdf_rdy, app_rd_data_valid;
    wire [127:0] app_rd_data;
    wire app_en, app_wdf_wren;
    wire [2:0] app_cmd;
    wire [27:0] app_addr;
    wire [127:0] app_wdf_data;

    ddr3_interface_regctrl controller (
        .clk(clk),
        .rst(rst),
        .write_trigger(write_trigger),
        .write_addr(write_addr_reg),
        .write_data(write_data_reg),
        .read_trigger(read_trigger),
        .read_addr(read_addr_reg),
        .read_data(read_data_reg),
        .status(status_reg),
        .app_rdy(app_rdy),
        .app_wdf_rdy(app_wdf_rdy),
        .app_rd_data_valid(app_rd_data_valid),
        .app_rd_data(app_rd_data),
        .app_en(app_en),
```

```
        .app_cmd(app_cmd),
        .app_addr(app_addr),
        .app_wdf_data(app_wdf_data),
        .app_wdf_wren(app_wdf_wren)
    );

    ddr3_model mem (
        .clk(clk),
        .rst(rst),
        .app_en(app_en),
        .app_cmd(app_cmd),
        .app_addr(app_addr),
        .app_wdf_data(app_wdf_data),
        .app_wdf_wren(app_wdf_wren),
        .app_rdy(app_rdy),
        .app_wdf_rdy(app_wdf_rdy),
        .app_rd_data(app_rd_data),
        .app_rd_data_valid(app_rd_data_valid)
    );

    integer i;
    integer pass_count = 0;
    integer fail_count = 0;
    reg [127:0] pattern;

    initial begin
        $display("Starting full DDR3 register interface stress test...");
        #20 rst = 0;
        write_trigger = 0;
        read_trigger = 0;

        for (i = 0; i < 256; i = i + 4) begin
            write_addr_reg = i[27:0];
            pattern = {96'hCAFEBABEDEADBEEF11223344 ^ i, i ^ (i << 2)};
write_data_reg = pattern;
            #10 write_trigger = 1;
            #10 write_trigger = 0;
            wait (status_reg != 2'b00);
            #10;
```

```
        if (status_reg == 2'b10) begin
            $display("[FAIL] Burst write failed at addr 0x%08h", write_addr_reg);
            fail_count = fail_count + 1;
        end

        read_addr_reg = i[27:0];
        #10 read_trigger = 1;
        #10 read_trigger = 0;
        wait (status_reg != 2'b00);
        #10;

        if (status_reg == 2'b10) begin
            $display("[FAIL] Burst read failed at addr 0x%08h", read_addr_reg);
            fail_count = fail_count + 1;
        end else begin
            $display("[PASS] Verified burst read @ 0x%08h", read_addr_reg);
            pass_count = pass_count + 1;
        end
    end

    $display("\n================= DDR3 Register Interface Stress Test Complete
=================");
    $display("Total Burst Blocks Tested: %0d", pass_count + fail_count);
    $display("  Passes: %0d", pass_count);
    $display("  Fails : %0d", fail_count);

$display("============================================================
=================");

    #100 $finish;
  end

  always #5 clk = ~clk;
endmodule
```

Appendix N - Version4.log

```
Command: /home/users14/dec71046/526L/project/Version4/./simv -l Version4.log
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP1_Full64; Runtime version U-2023.03-SP1_Full64;  Apr 21
17:04 2025
VCD+ Writer U-2023.03-SP1_Full64 Copyright (c) 1991-2023 by Synopsys Inc.
Starting full DDR3 register interface stress test...
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000000
[REGCTRL] WRITE 0x00000000 = 0xcafebabedeadbeef1122334400000000
[REGCTRL] WRITE 0x00000001 = 0xcafebabedeadbeef1122334400000001
[REGCTRL] WRITE 0x00000002 = 0xcafebabedeadbeef1122334400000002
[REGCTRL] WRITE 0x00000003 = 0xcafebabedeadbeef1122334400000003
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000000
[REGCTRL] READ request for addr 0x00000000
[REGCTRL] READ request for addr 0x00000001
[REGCTRL] READ request for addr 0x00000002
[REGCTRL] READ request for addr 0x00000003
[REGCTRL] READ OK   Addr: 0x00000000 Data: 0xcafebabedeadbeef1122334400000000
[REGCTRL] READ OK   Addr: 0x00000001 Data: 0xcafebabedeadbeef1122334400000001
[REGCTRL] READ OK   Addr: 0x00000002 Data: 0xcafebabedeadbeef1122334400000002
[REGCTRL] READ OK   Addr: 0x00000003 Data: 0xcafebabedeadbeef1122334400000003
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000000
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000004
[REGCTRL] WRITE 0x00000004 = 0xcafebabedeadbeef1122334000000014
[REGCTRL] WRITE 0x00000005 = 0xcafebabedeadbeef1122334000000015
[REGCTRL] WRITE 0x00000006 = 0xcafebabedeadbeef1122334000000016
[REGCTRL] WRITE 0x00000007 = 0xcafebabedeadbeef1122334000000017
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000004
[REGCTRL] READ request for addr 0x00000004
[REGCTRL] READ request for addr 0x00000005
[REGCTRL] READ request for addr 0x00000006
[REGCTRL] READ request for addr 0x00000007
```

[REGCTRL] READ OK Addr: 0x00000004 Data: 0xcafebabedeadbeef1122334000000014
[REGCTRL] READ OK Addr: 0x00000005 Data: 0xcafebabedeadbeef1122334000000015
[REGCTRL] READ OK Addr: 0x00000006 Data: 0xcafebabedeadbeef1122334000000016
[REGCTRL] READ OK Addr: 0x00000007 Data: 0xcafebabedeadbeef1122334000000017
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000004
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000008
[REGCTRL] WRITE 0x00000008 = 0xcafebabedeadbeef1122334c00000028
[REGCTRL] WRITE 0x00000009 = 0xcafebabedeadbeef1122334c00000029
[REGCTRL] WRITE 0x0000000a = 0xcafebabedeadbeef1122334c0000002a
[REGCTRL] WRITE 0x0000000b = 0xcafebabedeadbeef1122334c0000002b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000008
[REGCTRL] READ request for addr 0x00000008
[REGCTRL] READ request for addr 0x00000009
[REGCTRL] READ request for addr 0x0000000a
[REGCTRL] READ request for addr 0x0000000b
[REGCTRL] READ OK Addr: 0x00000008 Data: 0xcafebabedeadbeef1122334c00000028
[REGCTRL] READ OK Addr: 0x00000009 Data: 0xcafebabedeadbeef1122334c00000029
[REGCTRL] READ OK Addr: 0x0000000a Data: 0xcafebabedeadbeef1122334c0000002a
[REGCTRL] READ OK Addr: 0x0000000b Data: 0xcafebabedeadbeef1122334c0000002b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000008
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000000c
[REGCTRL] WRITE 0x0000000c = 0xcafebabedeadbeef112233480000003c
[REGCTRL] WRITE 0x0000000d = 0xcafebabedeadbeef112233480000003d
[REGCTRL] WRITE 0x0000000e = 0xcafebabedeadbeef112233480000003e
[REGCTRL] WRITE 0x0000000f = 0xcafebabedeadbeef112233480000003f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000000c
[REGCTRL] READ request for addr 0x0000000c
[REGCTRL] READ request for addr 0x0000000d
[REGCTRL] READ request for addr 0x0000000e
[REGCTRL] READ request for addr 0x0000000f
[REGCTRL] READ OK Addr: 0x0000000c Data: 0xcafebabedeadbeef112233480000003c
[REGCTRL] READ OK Addr: 0x0000000d Data: 0xcafebabedeadbeef112233480000003d

[REGCTRL] READ OK Addr: 0x0000000e Data: 0xcafebabedeadbeef112233480000003e
[REGCTRL] READ OK Addr: 0x0000000f Data: 0xcafebabedeadbeef112233480000003f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000000c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000010
[REGCTRL] WRITE 0x00000010 = 0xcafebabedeadbeef1122335400000050
[REGCTRL] WRITE 0x00000011 = 0xcafebabedeadbeef1122335400000051
[REGCTRL] WRITE 0x00000012 = 0xcafebabedeadbeef1122335400000052
[REGCTRL] WRITE 0x00000013 = 0xcafebabedeadbeef1122335400000053
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000010
[REGCTRL] READ request for addr 0x00000010
[REGCTRL] READ request for addr 0x00000011
[REGCTRL] READ request for addr 0x00000012
[REGCTRL] READ request for addr 0x00000013
[REGCTRL] READ OK Addr: 0x00000010 Data: 0xcafebabedeadbeef1122335400000050
[REGCTRL] READ OK Addr: 0x00000011 Data: 0xcafebabedeadbeef1122335400000051
[REGCTRL] READ OK Addr: 0x00000012 Data: 0xcafebabedeadbeef1122335400000052
[REGCTRL] READ OK Addr: 0x00000013 Data: 0xcafebabedeadbeef1122335400000053
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000010
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000014
[REGCTRL] WRITE 0x00000014 = 0xcafebabedeadbeef1122335000000044
[REGCTRL] WRITE 0x00000015 = 0xcafebabedeadbeef1122335000000045
[REGCTRL] WRITE 0x00000016 = 0xcafebabedeadbeef1122335000000046
[REGCTRL] WRITE 0x00000017 = 0xcafebabedeadbeef1122335000000047
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000014
[REGCTRL] READ request for addr 0x00000014
[REGCTRL] READ request for addr 0x00000015
[REGCTRL] READ request for addr 0x00000016
[REGCTRL] READ request for addr 0x00000017
[REGCTRL] READ OK Addr: 0x00000014 Data: 0xcafebabedeadbeef1122335000000044
[REGCTRL] READ OK Addr: 0x00000015 Data: 0xcafebabedeadbeef1122335000000045
[REGCTRL] READ OK Addr: 0x00000016 Data: 0xcafebabedeadbeef1122335000000046
[REGCTRL] READ OK Addr: 0x00000017 Data: 0xcafebabedeadbeef1122335000000047

[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000014
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000018
[REGCTRL] WRITE 0x00000018 = 0xcafebabedeadbeef1122335c00000078
[REGCTRL] WRITE 0x00000019 = 0xcafebabedeadbeef1122335c00000079
[REGCTRL] WRITE 0x0000001a = 0xcafebabedeadbeef1122335c0000007a
[REGCTRL] WRITE 0x0000001b = 0xcafebabedeadbeef1122335c0000007b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000018
[REGCTRL] READ request for addr 0x00000018
[REGCTRL] READ request for addr 0x00000019
[REGCTRL] READ request for addr 0x0000001a
[REGCTRL] READ request for addr 0x0000001b
[REGCTRL] READ OK Addr: 0x00000018 Data: 0xcafebabedeadbeef1122335c00000078
[REGCTRL] READ OK Addr: 0x00000019 Data: 0xcafebabedeadbeef1122335c00000079
[REGCTRL] READ OK Addr: 0x0000001a Data: 0xcafebabedeadbeef1122335c0000007a
[REGCTRL] READ OK Addr: 0x0000001b Data: 0xcafebabedeadbeef1122335c0000007b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000018
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000001c
[REGCTRL] WRITE 0x0000001c = 0xcafebabedeadbeef112233580000006c
[REGCTRL] WRITE 0x0000001d = 0xcafebabedeadbeef112233580000006d
[REGCTRL] WRITE 0x0000001e = 0xcafebabedeadbeef112233580000006e
[REGCTRL] WRITE 0x0000001f = 0xcafebabedeadbeef112233580000006f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000001c
[REGCTRL] READ request for addr 0x0000001c
[REGCTRL] READ request for addr 0x0000001d
[REGCTRL] READ request for addr 0x0000001e
[REGCTRL] READ request for addr 0x0000001f
[REGCTRL] READ OK Addr: 0x0000001c Data: 0xcafebabedeadbeef112233580000006c
[REGCTRL] READ OK Addr: 0x0000001d Data: 0xcafebabedeadbeef112233580000006d
[REGCTRL] READ OK Addr: 0x0000001e Data: 0xcafebabedeadbeef112233580000006e
[REGCTRL] READ OK Addr: 0x0000001f Data: 0xcafebabedeadbeef112233580000006f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000001c

[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000020
[REGCTRL] WRITE 0x00000020 = 0xcafebabedeadbeef11223364000000a0
[REGCTRL] WRITE 0x00000021 = 0xcafebabedeadbeef11223364000000a1
[REGCTRL] WRITE 0x00000022 = 0xcafebabedeadbeef11223364000000a2
[REGCTRL] WRITE 0x00000023 = 0xcafebabedeadbeef11223364000000a3
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000020
[REGCTRL] READ request for addr 0x00000020
[REGCTRL] READ request for addr 0x00000021
[REGCTRL] READ request for addr 0x00000022
[REGCTRL] READ request for addr 0x00000023
[REGCTRL] READ OK Addr: 0x00000020 Data: 0xcafebabedeadbeef11223364000000a0
[REGCTRL] READ OK Addr: 0x00000021 Data: 0xcafebabedeadbeef11223364000000a1
[REGCTRL] READ OK Addr: 0x00000022 Data: 0xcafebabedeadbeef11223364000000a2
[REGCTRL] READ OK Addr: 0x00000023 Data: 0xcafebabedeadbeef11223364000000a3
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000020
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000024
[REGCTRL] WRITE 0x00000024 = 0xcafebabedeadbeef11223360000000b4
[REGCTRL] WRITE 0x00000025 = 0xcafebabedeadbeef11223360000000b5
[REGCTRL] WRITE 0x00000026 = 0xcafebabedeadbeef11223360000000b6
[REGCTRL] WRITE 0x00000027 = 0xcafebabedeadbeef11223360000000b7
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000024
[REGCTRL] READ request for addr 0x00000024
[REGCTRL] READ request for addr 0x00000025
[REGCTRL] READ request for addr 0x00000026
[REGCTRL] READ request for addr 0x00000027
[REGCTRL] READ OK Addr: 0x00000024 Data: 0xcafebabedeadbeef11223360000000b4
[REGCTRL] READ OK Addr: 0x00000025 Data: 0xcafebabedeadbeef11223360000000b5
[REGCTRL] READ OK Addr: 0x00000026 Data: 0xcafebabedeadbeef11223360000000b6
[REGCTRL] READ OK Addr: 0x00000027 Data: 0xcafebabedeadbeef11223360000000b7
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000024
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000028

[REGCTRL] WRITE 0x00000028 = 0xcafebabedeadbeef1122336c00000088
[REGCTRL] WRITE 0x00000029 = 0xcafebabedeadbeef1122336c00000089
[REGCTRL] WRITE 0x0000002a = 0xcafebabedeadbeef1122336c0000008a
[REGCTRL] WRITE 0x0000002b = 0xcafebabedeadbeef1122336c0000008b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000028
[REGCTRL] READ request for addr 0x00000028
[REGCTRL] READ request for addr 0x00000029
[REGCTRL] READ request for addr 0x0000002a
[REGCTRL] READ request for addr 0x0000002b
[REGCTRL] READ OK Addr: 0x00000028 Data: 0xcafebabedeadbeef1122336c00000088
[REGCTRL] READ OK Addr: 0x00000029 Data: 0xcafebabedeadbeef1122336c00000089
[REGCTRL] READ OK Addr: 0x0000002a Data: 0xcafebabedeadbeef1122336c0000008a
[REGCTRL] READ OK Addr: 0x0000002b Data: 0xcafebabedeadbeef1122336c0000008b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000028
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000002c
[REGCTRL] WRITE 0x0000002c = 0xcafebabedeadbeef112233680000009c
[REGCTRL] WRITE 0x0000002d = 0xcafebabedeadbeef112233680000009d
[REGCTRL] WRITE 0x0000002e = 0xcafebabedeadbeef112233680000009e
[REGCTRL] WRITE 0x0000002f = 0xcafebabedeadbeef112233680000009f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000002c
[REGCTRL] READ request for addr 0x0000002c
[REGCTRL] READ request for addr 0x0000002d
[REGCTRL] READ request for addr 0x0000002e
[REGCTRL] READ request for addr 0x0000002f
[REGCTRL] READ OK Addr: 0x0000002c Data: 0xcafebabedeadbeef112233680000009c
[REGCTRL] READ OK Addr: 0x0000002d Data: 0xcafebabedeadbeef112233680000009d
[REGCTRL] READ OK Addr: 0x0000002e Data: 0xcafebabedeadbeef112233680000009e
[REGCTRL] READ OK Addr: 0x0000002f Data: 0xcafebabedeadbeef112233680000009f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000002c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000030
[REGCTRL] WRITE 0x00000030 = 0xcafebabedeadbeef11223374000000f0
[REGCTRL] WRITE 0x00000031 = 0xcafebabedeadbeef11223374000000f1

[REGCTRL] WRITE 0x00000032 = 0xcafebabedeadbeef11223374000000f2
[REGCTRL] WRITE 0x00000033 = 0xcafebabedeadbeef11223374000000f3
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000030
[REGCTRL] READ request for addr 0x00000030
[REGCTRL] READ request for addr 0x00000031
[REGCTRL] READ request for addr 0x00000032
[REGCTRL] READ request for addr 0x00000033
[REGCTRL] READ OK Addr: 0x00000030 Data: 0xcafebabedeadbeef11223374000000f0
[REGCTRL] READ OK Addr: 0x00000031 Data: 0xcafebabedeadbeef11223374000000f1
[REGCTRL] READ OK Addr: 0x00000032 Data: 0xcafebabedeadbeef11223374000000f2
[REGCTRL] READ OK Addr: 0x00000033 Data: 0xcafebabedeadbeef11223374000000f3
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000030
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000034
[REGCTRL] WRITE 0x00000034 = 0xcafebabedeadbeef11223370000000e4
[REGCTRL] WRITE 0x00000035 = 0xcafebabedeadbeef11223370000000e5
[REGCTRL] WRITE 0x00000036 = 0xcafebabedeadbeef11223370000000e6
[REGCTRL] WRITE 0x00000037 = 0xcafebabedeadbeef11223370000000e7
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000034
[REGCTRL] READ request for addr 0x00000034
[REGCTRL] READ request for addr 0x00000035
[REGCTRL] READ request for addr 0x00000036
[REGCTRL] READ request for addr 0x00000037
[REGCTRL] READ OK Addr: 0x00000034 Data: 0xcafebabedeadbeef11223370000000e4
[REGCTRL] READ OK Addr: 0x00000035 Data: 0xcafebabedeadbeef11223370000000e5
[REGCTRL] READ OK Addr: 0x00000036 Data: 0xcafebabedeadbeef11223370000000e6
[REGCTRL] READ OK Addr: 0x00000037 Data: 0xcafebabedeadbeef11223370000000e7
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000034
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000038
[REGCTRL] WRITE 0x00000038 = 0xcafebabedeadbeef1122337c000000d8
[REGCTRL] WRITE 0x00000039 = 0xcafebabedeadbeef1122337c000000d9
[REGCTRL] WRITE 0x0000003a = 0xcafebabedeadbeef1122337c000000da
[REGCTRL] WRITE 0x0000003b = 0xcafebabedeadbeef1122337c000000db

[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000038
[REGCTRL] READ request for addr 0x00000038
[REGCTRL] READ request for addr 0x00000039
[REGCTRL] READ request for addr 0x0000003a
[REGCTRL] READ request for addr 0x0000003b
[REGCTRL] READ OK Addr: 0x00000038 Data: 0xcafebabedeadbeef1122337c000000d8
[REGCTRL] READ OK Addr: 0x00000039 Data: 0xcafebabedeadbeef1122337c000000d9
[REGCTRL] READ OK Addr: 0x0000003a Data: 0xcafebabedeadbeef1122337c000000da
[REGCTRL] READ OK Addr: 0x0000003b Data: 0xcafebabedeadbeef1122337c000000db
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000038
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000003c
[REGCTRL] WRITE 0x0000003c = 0xcafebabedeadbeef11223378000000cc
[REGCTRL] WRITE 0x0000003d = 0xcafebabedeadbeef11223378000000cd
[REGCTRL] WRITE 0x0000003e = 0xcafebabedeadbeef11223378000000ce
[REGCTRL] WRITE 0x0000003f = 0xcafebabedeadbeef11223378000000cf
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000003c
[REGCTRL] READ request for addr 0x0000003c
[REGCTRL] READ request for addr 0x0000003d
[REGCTRL] READ request for addr 0x0000003e
[REGCTRL] READ request for addr 0x0000003f
[REGCTRL] READ OK Addr: 0x0000003c Data: 0xcafebabedeadbeef11223378000000cc
[REGCTRL] READ OK Addr: 0x0000003d Data: 0xcafebabedeadbeef11223378000000cd
[REGCTRL] READ OK Addr: 0x0000003e Data: 0xcafebabedeadbeef11223378000000ce
[REGCTRL] READ OK Addr: 0x0000003f Data: 0xcafebabedeadbeef11223378000000cf
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000003c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000040
[REGCTRL] WRITE 0x00000040 = 0xcafebabedeadbeef1122330400000140
[REGCTRL] WRITE 0x00000041 = 0xcafebabedeadbeef1122330400000141
[REGCTRL] WRITE 0x00000042 = 0xcafebabedeadbeef1122330400000142
[REGCTRL] WRITE 0x00000043 = 0xcafebabedeadbeef1122330400000143
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.

[REGCTRL] Read trigger received. Starting burst read at addr 0x00000040
[REGCTRL] READ request for addr 0x00000040
[REGCTRL] READ request for addr 0x00000041
[REGCTRL] READ request for addr 0x00000042
[REGCTRL] READ request for addr 0x00000043
[REGCTRL] READ OK Addr: 0x00000040 Data: 0xcafebabedeadbeef1122330400000140
[REGCTRL] READ OK Addr: 0x00000041 Data: 0xcafebabedeadbeef1122330400000141
[REGCTRL] READ OK Addr: 0x00000042 Data: 0xcafebabedeadbeef1122330400000142
[REGCTRL] READ OK Addr: 0x00000043 Data: 0xcafebabedeadbeef1122330400000143
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000040
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000044
[REGCTRL] WRITE 0x00000044 = 0xcafebabedeadbeef1122330000000154
[REGCTRL] WRITE 0x00000045 = 0xcafebabedeadbeef1122330000000155
[REGCTRL] WRITE 0x00000046 = 0xcafebabedeadbeef1122330000000156
[REGCTRL] WRITE 0x00000047 = 0xcafebabedeadbeef1122330000000157
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000044
[REGCTRL] READ request for addr 0x00000044
[REGCTRL] READ request for addr 0x00000045
[REGCTRL] READ request for addr 0x00000046
[REGCTRL] READ request for addr 0x00000047
[REGCTRL] READ OK Addr: 0x00000044 Data: 0xcafebabedeadbeef1122330000000154
[REGCTRL] READ OK Addr: 0x00000045 Data: 0xcafebabedeadbeef1122330000000155
[REGCTRL] READ OK Addr: 0x00000046 Data: 0xcafebabedeadbeef1122330000000156
[REGCTRL] READ OK Addr: 0x00000047 Data: 0xcafebabedeadbeef1122330000000157
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000044
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000048
[REGCTRL] WRITE 0x00000048 = 0xcafebabedeadbeef1122330c00000168
[REGCTRL] WRITE 0x00000049 = 0xcafebabedeadbeef1122330c00000169
[REGCTRL] WRITE 0x0000004a = 0xcafebabedeadbeef1122330c0000016a
[REGCTRL] WRITE 0x0000004b = 0xcafebabedeadbeef1122330c0000016b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000048
[REGCTRL] READ request for addr 0x00000048

[REGCTRL] READ request for addr 0x00000049
[REGCTRL] READ request for addr 0x0000004a
[REGCTRL] READ request for addr 0x0000004b
[REGCTRL] READ OK Addr: 0x00000048 Data: 0xcafebabedeadbeef1122330c00000168
[REGCTRL] READ OK Addr: 0x00000049 Data: 0xcafebabedeadbeef1122330c00000169
[REGCTRL] READ OK Addr: 0x0000004a Data: 0xcafebabedeadbeef1122330c0000016a
[REGCTRL] READ OK Addr: 0x0000004b Data: 0xcafebabedeadbeef1122330c0000016b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000048
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000004c
[REGCTRL] WRITE 0x0000004c = 0xcafebabedeadbeef112233080000017c
[REGCTRL] WRITE 0x0000004d = 0xcafebabedeadbeef112233080000017d
[REGCTRL] WRITE 0x0000004e = 0xcafebabedeadbeef112233080000017e
[REGCTRL] WRITE 0x0000004f = 0xcafebabedeadbeef112233080000017f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000004c
[REGCTRL] READ request for addr 0x0000004c
[REGCTRL] READ request for addr 0x0000004d
[REGCTRL] READ request for addr 0x0000004e
[REGCTRL] READ request for addr 0x0000004f
[REGCTRL] READ OK Addr: 0x0000004c Data: 0xcafebabedeadbeef112233080000017c
[REGCTRL] READ OK Addr: 0x0000004d Data: 0xcafebabedeadbeef112233080000017d
[REGCTRL] READ OK Addr: 0x0000004e Data: 0xcafebabedeadbeef112233080000017e
[REGCTRL] READ OK Addr: 0x0000004f Data: 0xcafebabedeadbeef112233080000017f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000004c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000050
[REGCTRL] WRITE 0x00000050 = 0xcafebabedeadbeef1122331400000110
[REGCTRL] WRITE 0x00000051 = 0xcafebabedeadbeef1122331400000111
[REGCTRL] WRITE 0x00000052 = 0xcafebabedeadbeef1122331400000112
[REGCTRL] WRITE 0x00000053 = 0xcafebabedeadbeef1122331400000113
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000050
[REGCTRL] READ request for addr 0x00000050
[REGCTRL] READ request for addr 0x00000051
[REGCTRL] READ request for addr 0x00000052

[REGCTRL] READ request for addr 0x00000053
[REGCTRL] READ OK Addr: 0x00000050 Data: 0xcafebabedeadbeef1122331400000110
[REGCTRL] READ OK Addr: 0x00000051 Data: 0xcafebabedeadbeef1122331400000111
[REGCTRL] READ OK Addr: 0x00000052 Data: 0xcafebabedeadbeef1122331400000112
[REGCTRL] READ OK Addr: 0x00000053 Data: 0xcafebabedeadbeef1122331400000113
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000050
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000054
[REGCTRL] WRITE 0x00000054 = 0xcafebabedeadbeef1122331000000104
[REGCTRL] WRITE 0x00000055 = 0xcafebabedeadbeef1122331000000105
[REGCTRL] WRITE 0x00000056 = 0xcafebabedeadbeef1122331000000106
[REGCTRL] WRITE 0x00000057 = 0xcafebabedeadbeef1122331000000107
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000054
[REGCTRL] READ request for addr 0x00000054
[REGCTRL] READ request for addr 0x00000055
[REGCTRL] READ request for addr 0x00000056
[REGCTRL] READ request for addr 0x00000057
[REGCTRL] READ OK Addr: 0x00000054 Data: 0xcafebabedeadbeef1122331000000104
[REGCTRL] READ OK Addr: 0x00000055 Data: 0xcafebabedeadbeef1122331000000105
[REGCTRL] READ OK Addr: 0x00000056 Data: 0xcafebabedeadbeef1122331000000106
[REGCTRL] READ OK Addr: 0x00000057 Data: 0xcafebabedeadbeef1122331000000107
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000054
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000058
[REGCTRL] WRITE 0x00000058 = 0xcafebabedeadbeef1122331c00000138
[REGCTRL] WRITE 0x00000059 = 0xcafebabedeadbeef1122331c00000139
[REGCTRL] WRITE 0x0000005a = 0xcafebabedeadbeef1122331c0000013a
[REGCTRL] WRITE 0x0000005b = 0xcafebabedeadbeef1122331c0000013b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000058
[REGCTRL] READ request for addr 0x00000058
[REGCTRL] READ request for addr 0x00000059
[REGCTRL] READ request for addr 0x0000005a
[REGCTRL] READ request for addr 0x0000005b
[REGCTRL] READ OK Addr: 0x00000058 Data: 0xcafebabedeadbeef1122331c00000138

[REGCTRL] READ OK Addr: 0x00000059 Data: 0xcafebabedeadbeef1122331c00000139
[REGCTRL] READ OK Addr: 0x0000005a Data: 0xcafebabedeadbeef1122331c0000013a
[REGCTRL] READ OK Addr: 0x0000005b Data: 0xcafebabedeadbeef1122331c0000013b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000058
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000005c
[REGCTRL] WRITE 0x0000005c = 0xcafebabedeadbeef112233180000012c
[REGCTRL] WRITE 0x0000005d = 0xcafebabedeadbeef112233180000012d
[REGCTRL] WRITE 0x0000005e = 0xcafebabedeadbeef112233180000012e
[REGCTRL] WRITE 0x0000005f = 0xcafebabedeadbeef112233180000012f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000005c
[REGCTRL] READ request for addr 0x0000005c
[REGCTRL] READ request for addr 0x0000005d
[REGCTRL] READ request for addr 0x0000005e
[REGCTRL] READ request for addr 0x0000005f
[REGCTRL] READ OK Addr: 0x0000005c Data: 0xcafebabedeadbeef112233180000012c
[REGCTRL] READ OK Addr: 0x0000005d Data: 0xcafebabedeadbeef112233180000012d
[REGCTRL] READ OK Addr: 0x0000005e Data: 0xcafebabedeadbeef112233180000012e
[REGCTRL] READ OK Addr: 0x0000005f Data: 0xcafebabedeadbeef112233180000012f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000005c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000060
[REGCTRL] WRITE 0x00000060 = 0xcafebabedeadbeef11223324000001e0
[REGCTRL] WRITE 0x00000061 = 0xcafebabedeadbeef11223324000001e1
[REGCTRL] WRITE 0x00000062 = 0xcafebabedeadbeef11223324000001e2
[REGCTRL] WRITE 0x00000063 = 0xcafebabedeadbeef11223324000001e3
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000060
[REGCTRL] READ request for addr 0x00000060
[REGCTRL] READ request for addr 0x00000061
[REGCTRL] READ request for addr 0x00000062
[REGCTRL] READ request for addr 0x00000063
[REGCTRL] READ OK Addr: 0x00000060 Data: 0xcafebabedeadbeef11223324000001e0
[REGCTRL] READ OK Addr: 0x00000061 Data: 0xcafebabedeadbeef11223324000001e1
[REGCTRL] READ OK Addr: 0x00000062 Data: 0xcafebabedeadbeef11223324000001e2

[REGCTRL] READ OK Addr: 0x00000063 Data: 0xcafebabedeadbeef11223324000001e3
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000060
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000064
[REGCTRL] WRITE 0x00000064 = 0xcafebabedeadbeef11223320000001f4
[REGCTRL] WRITE 0x00000065 = 0xcafebabedeadbeef11223320000001f5
[REGCTRL] WRITE 0x00000066 = 0xcafebabedeadbeef11223320000001f6
[REGCTRL] WRITE 0x00000067 = 0xcafebabedeadbeef11223320000001f7
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000064
[REGCTRL] READ request for addr 0x00000064
[REGCTRL] READ request for addr 0x00000065
[REGCTRL] READ request for addr 0x00000066
[REGCTRL] READ request for addr 0x00000067
[REGCTRL] READ OK Addr: 0x00000064 Data: 0xcafebabedeadbeef11223320000001f4
[REGCTRL] READ OK Addr: 0x00000065 Data: 0xcafebabedeadbeef11223320000001f5
[REGCTRL] READ OK Addr: 0x00000066 Data: 0xcafebabedeadbeef11223320000001f6
[REGCTRL] READ OK Addr: 0x00000067 Data: 0xcafebabedeadbeef11223320000001f7
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000064
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000068
[REGCTRL] WRITE 0x00000068 = 0xcafebabedeadbeef1122332c000001c8
[REGCTRL] WRITE 0x00000069 = 0xcafebabedeadbeef1122332c000001c9
[REGCTRL] WRITE 0x0000006a = 0xcafebabedeadbeef1122332c000001ca
[REGCTRL] WRITE 0x0000006b = 0xcafebabedeadbeef1122332c000001cb
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000068
[REGCTRL] READ request for addr 0x00000068
[REGCTRL] READ request for addr 0x00000069
[REGCTRL] READ request for addr 0x0000006a
[REGCTRL] READ request for addr 0x0000006b
[REGCTRL] READ OK Addr: 0x00000068 Data: 0xcafebabedeadbeef1122332c000001c8
[REGCTRL] READ OK Addr: 0x00000069 Data: 0xcafebabedeadbeef1122332c000001c9
[REGCTRL] READ OK Addr: 0x0000006a Data: 0xcafebabedeadbeef1122332c000001ca
[REGCTRL] READ OK Addr: 0x0000006b Data: 0xcafebabedeadbeef1122332c000001cb
[REGCTRL] Burst read complete and verified.

[PASS] Verified burst read @ 0x00000068
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000006c
[REGCTRL] WRITE 0x0000006c = 0xcafebabedeadbeef11223328000001dc
[REGCTRL] WRITE 0x0000006d = 0xcafebabedeadbeef11223328000001dd
[REGCTRL] WRITE 0x0000006e = 0xcafebabedeadbeef11223328000001de
[REGCTRL] WRITE 0x0000006f = 0xcafebabedeadbeef11223328000001df
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000006c
[REGCTRL] READ request for addr 0x0000006c
[REGCTRL] READ request for addr 0x0000006d
[REGCTRL] READ request for addr 0x0000006e
[REGCTRL] READ request for addr 0x0000006f
[REGCTRL] READ OK Addr: 0x0000006c Data: 0xcafebabedeadbeef11223328000001dc
[REGCTRL] READ OK Addr: 0x0000006d Data: 0xcafebabedeadbeef11223328000001dd
[REGCTRL] READ OK Addr: 0x0000006e Data: 0xcafebabedeadbeef11223328000001de
[REGCTRL] READ OK Addr: 0x0000006f Data: 0xcafebabedeadbeef11223328000001df
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000006c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000070
[REGCTRL] WRITE 0x00000070 = 0xcafebabedeadbeef11223334000001b0
[REGCTRL] WRITE 0x00000071 = 0xcafebabedeadbeef11223334000001b1
[REGCTRL] WRITE 0x00000072 = 0xcafebabedeadbeef11223334000001b2
[REGCTRL] WRITE 0x00000073 = 0xcafebabedeadbeef11223334000001b3
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000070
[REGCTRL] READ request for addr 0x00000070
[REGCTRL] READ request for addr 0x00000071
[REGCTRL] READ request for addr 0x00000072
[REGCTRL] READ request for addr 0x00000073
[REGCTRL] READ OK Addr: 0x00000070 Data: 0xcafebabedeadbeef11223334000001b0
[REGCTRL] READ OK Addr: 0x00000071 Data: 0xcafebabedeadbeef11223334000001b1
[REGCTRL] READ OK Addr: 0x00000072 Data: 0xcafebabedeadbeef11223334000001b2
[REGCTRL] READ OK Addr: 0x00000073 Data: 0xcafebabedeadbeef11223334000001b3
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000070
[REGCTRL] Returning to IDLE.

[REGCTRL] Write trigger received. Starting burst write at addr 0x00000074
[REGCTRL] WRITE 0x00000074 = 0xcafebabedeadbeef11223330000001a4
[REGCTRL] WRITE 0x00000075 = 0xcafebabedeadbeef11223330000001a5
[REGCTRL] WRITE 0x00000076 = 0xcafebabedeadbeef11223330000001a6
[REGCTRL] WRITE 0x00000077 = 0xcafebabedeadbeef11223330000001a7
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000074
[REGCTRL] READ request for addr 0x00000074
[REGCTRL] READ request for addr 0x00000075
[REGCTRL] READ request for addr 0x00000076
[REGCTRL] READ request for addr 0x00000077
[REGCTRL] READ OK Addr: 0x00000074 Data: 0xcafebabedeadbeef11223330000001a4
[REGCTRL] READ OK Addr: 0x00000075 Data: 0xcafebabedeadbeef11223330000001a5
[REGCTRL] READ OK Addr: 0x00000076 Data: 0xcafebabedeadbeef11223330000001a6
[REGCTRL] READ OK Addr: 0x00000077 Data: 0xcafebabedeadbeef11223330000001a7
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000074
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000078
[REGCTRL] WRITE 0x00000078 = 0xcafebabedeadbeef1122333c00000198
[REGCTRL] WRITE 0x00000079 = 0xcafebabedeadbeef1122333c00000199
[REGCTRL] WRITE 0x0000007a = 0xcafebabedeadbeef1122333c0000019a
[REGCTRL] WRITE 0x0000007b = 0xcafebabedeadbeef1122333c0000019b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000078
[REGCTRL] READ request for addr 0x00000078
[REGCTRL] READ request for addr 0x00000079
[REGCTRL] READ request for addr 0x0000007a
[REGCTRL] READ request for addr 0x0000007b
[REGCTRL] READ OK Addr: 0x00000078 Data: 0xcafebabedeadbeef1122333c00000198
[REGCTRL] READ OK Addr: 0x00000079 Data: 0xcafebabedeadbeef1122333c00000199
[REGCTRL] READ OK Addr: 0x0000007a Data: 0xcafebabedeadbeef1122333c0000019a
[REGCTRL] READ OK Addr: 0x0000007b Data: 0xcafebabedeadbeef1122333c0000019b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000078
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000007c
[REGCTRL] WRITE 0x0000007c = 0xcafebabedeadbeef112233380000018c

[REGCTRL] WRITE 0x0000007d = 0xcafebabedeadbeef112233380000018d
[REGCTRL] WRITE 0x0000007e = 0xcafebabedeadbeef112233380000018e
[REGCTRL] WRITE 0x0000007f = 0xcafebabedeadbeef112233380000018f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000007c
[REGCTRL] READ request for addr 0x0000007c
[REGCTRL] READ request for addr 0x0000007d
[REGCTRL] READ request for addr 0x0000007e
[REGCTRL] READ request for addr 0x0000007f
[REGCTRL] READ OK Addr: 0x0000007c Data: 0xcafebabedeadbeef112233380000018c
[REGCTRL] READ OK Addr: 0x0000007d Data: 0xcafebabedeadbeef112233380000018d
[REGCTRL] READ OK Addr: 0x0000007e Data: 0xcafebabedeadbeef112233380000018e
[REGCTRL] READ OK Addr: 0x0000007f Data: 0xcafebabedeadbeef112233380000018f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000007c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000080
[REGCTRL] WRITE 0x00000080 = 0xcafebabedeadbeef112233c400000280
[REGCTRL] WRITE 0x00000081 = 0xcafebabedeadbeef112233c400000281
[REGCTRL] WRITE 0x00000082 = 0xcafebabedeadbeef112233c400000282
[REGCTRL] WRITE 0x00000083 = 0xcafebabedeadbeef112233c400000283
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000080
[REGCTRL] READ request for addr 0x00000080
[REGCTRL] READ request for addr 0x00000081
[REGCTRL] READ request for addr 0x00000082
[REGCTRL] READ request for addr 0x00000083
[REGCTRL] READ OK Addr: 0x00000080 Data: 0xcafebabedeadbeef112233c400000280
[REGCTRL] READ OK Addr: 0x00000081 Data: 0xcafebabedeadbeef112233c400000281
[REGCTRL] READ OK Addr: 0x00000082 Data: 0xcafebabedeadbeef112233c400000282
[REGCTRL] READ OK Addr: 0x00000083 Data: 0xcafebabedeadbeef112233c400000283
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000080
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000084
[REGCTRL] WRITE 0x00000084 = 0xcafebabedeadbeef112233c000000294
[REGCTRL] WRITE 0x00000085 = 0xcafebabedeadbeef112233c000000295
[REGCTRL] WRITE 0x00000086 = 0xcafebabedeadbeef112233c000000296

[REGCTRL] WRITE 0x00000087 = 0xcafebabedeadbeef112233c000000297
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000084
[REGCTRL] READ request for addr 0x00000084
[REGCTRL] READ request for addr 0x00000085
[REGCTRL] READ request for addr 0x00000086
[REGCTRL] READ request for addr 0x00000087
[REGCTRL] READ OK Addr: 0x00000084 Data: 0xcafebabedeadbeef112233c000000294
[REGCTRL] READ OK Addr: 0x00000085 Data: 0xcafebabedeadbeef112233c000000295
[REGCTRL] READ OK Addr: 0x00000086 Data: 0xcafebabedeadbeef112233c000000296
[REGCTRL] READ OK Addr: 0x00000087 Data: 0xcafebabedeadbeef112233c000000297
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000084
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000088
[REGCTRL] WRITE 0x00000088 = 0xcafebabedeadbeef112233cc000002a8
[REGCTRL] WRITE 0x00000089 = 0xcafebabedeadbeef112233cc000002a9
[REGCTRL] WRITE 0x0000008a = 0xcafebabedeadbeef112233cc000002aa
[REGCTRL] WRITE 0x0000008b = 0xcafebabedeadbeef112233cc000002ab
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000088
[REGCTRL] READ request for addr 0x00000088
[REGCTRL] READ request for addr 0x00000089
[REGCTRL] READ request for addr 0x0000008a
[REGCTRL] READ request for addr 0x0000008b
[REGCTRL] READ OK Addr: 0x00000088 Data: 0xcafebabedeadbeef112233cc000002a8
[REGCTRL] READ OK Addr: 0x00000089 Data: 0xcafebabedeadbeef112233cc000002a9
[REGCTRL] READ OK Addr: 0x0000008a Data: 0xcafebabedeadbeef112233cc000002aa
[REGCTRL] READ OK Addr: 0x0000008b Data: 0xcafebabedeadbeef112233cc000002ab
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000088
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000008c
[REGCTRL] WRITE 0x0000008c = 0xcafebabedeadbeef112233c8000002bc
[REGCTRL] WRITE 0x0000008d = 0xcafebabedeadbeef112233c8000002bd
[REGCTRL] WRITE 0x0000008e = 0xcafebabedeadbeef112233c8000002be
[REGCTRL] WRITE 0x0000008f = 0xcafebabedeadbeef112233c8000002bf
[REGCTRL] Write burst complete.

[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000008c
[REGCTRL] READ request for addr 0x0000008c
[REGCTRL] READ request for addr 0x0000008d
[REGCTRL] READ request for addr 0x0000008e
[REGCTRL] READ request for addr 0x0000008f
[REGCTRL] READ OK Addr: 0x0000008c Data: 0xcafebabedeadbeef112233c8000002bc
[REGCTRL] READ OK Addr: 0x0000008d Data: 0xcafebabedeadbeef112233c8000002bd
[REGCTRL] READ OK Addr: 0x0000008e Data: 0xcafebabedeadbeef112233c8000002be
[REGCTRL] READ OK Addr: 0x0000008f Data: 0xcafebabedeadbeef112233c8000002bf
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000008c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000090
[REGCTRL] WRITE 0x00000090 = 0xcafebabedeadbeef112233d4000002d0
[REGCTRL] WRITE 0x00000091 = 0xcafebabedeadbeef112233d4000002d1
[REGCTRL] WRITE 0x00000092 = 0xcafebabedeadbeef112233d4000002d2
[REGCTRL] WRITE 0x00000093 = 0xcafebabedeadbeef112233d4000002d3
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000090
[REGCTRL] READ request for addr 0x00000090
[REGCTRL] READ request for addr 0x00000091
[REGCTRL] READ request for addr 0x00000092
[REGCTRL] READ request for addr 0x00000093
[REGCTRL] READ OK Addr: 0x00000090 Data: 0xcafebabedeadbeef112233d4000002d0
[REGCTRL] READ OK Addr: 0x00000091 Data: 0xcafebabedeadbeef112233d4000002d1
[REGCTRL] READ OK Addr: 0x00000092 Data: 0xcafebabedeadbeef112233d4000002d2
[REGCTRL] READ OK Addr: 0x00000093 Data: 0xcafebabedeadbeef112233d4000002d3
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000090
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000094
[REGCTRL] WRITE 0x00000094 = 0xcafebabedeadbeef112233d0000002c4
[REGCTRL] WRITE 0x00000095 = 0xcafebabedeadbeef112233d0000002c5
[REGCTRL] WRITE 0x00000096 = 0xcafebabedeadbeef112233d0000002c6
[REGCTRL] WRITE 0x00000097 = 0xcafebabedeadbeef112233d0000002c7
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000094

[REGCTRL] READ request for addr 0x00000094
[REGCTRL] READ request for addr 0x00000095
[REGCTRL] READ request for addr 0x00000096
[REGCTRL] READ request for addr 0x00000097
[REGCTRL] READ OK Addr: 0x00000094 Data: 0xcafebabedeadbeef112233d0000002c4
[REGCTRL] READ OK Addr: 0x00000095 Data: 0xcafebabedeadbeef112233d0000002c5
[REGCTRL] READ OK Addr: 0x00000096 Data: 0xcafebabedeadbeef112233d0000002c6
[REGCTRL] READ OK Addr: 0x00000097 Data: 0xcafebabedeadbeef112233d0000002c7
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000094
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x00000098
[REGCTRL] WRITE 0x00000098 = 0xcafebabedeadbeef112233dc000002f8
[REGCTRL] WRITE 0x00000099 = 0xcafebabedeadbeef112233dc000002f9
[REGCTRL] WRITE 0x0000009a = 0xcafebabedeadbeef112233dc000002fa
[REGCTRL] WRITE 0x0000009b = 0xcafebabedeadbeef112233dc000002fb
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x00000098
[REGCTRL] READ request for addr 0x00000098
[REGCTRL] READ request for addr 0x00000099
[REGCTRL] READ request for addr 0x0000009a
[REGCTRL] READ request for addr 0x0000009b
[REGCTRL] READ OK Addr: 0x00000098 Data: 0xcafebabedeadbeef112233dc000002f8
[REGCTRL] READ OK Addr: 0x00000099 Data: 0xcafebabedeadbeef112233dc000002f9
[REGCTRL] READ OK Addr: 0x0000009a Data: 0xcafebabedeadbeef112233dc000002fa
[REGCTRL] READ OK Addr: 0x0000009b Data: 0xcafebabedeadbeef112233dc000002fb
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x00000098
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x0000009c
[REGCTRL] WRITE 0x0000009c = 0xcafebabedeadbeef112233d8000002ec
[REGCTRL] WRITE 0x0000009d = 0xcafebabedeadbeef112233d8000002ed
[REGCTRL] WRITE 0x0000009e = 0xcafebabedeadbeef112233d8000002ee
[REGCTRL] WRITE 0x0000009f = 0xcafebabedeadbeef112233d8000002ef
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x0000009c
[REGCTRL] READ request for addr 0x0000009c
[REGCTRL] READ request for addr 0x0000009d

[REGCTRL] READ request for addr 0x0000009e
[REGCTRL] READ request for addr 0x0000009f
[REGCTRL] READ OK Addr: 0x0000009c Data: 0xcafebabedeadbeef112233d8000002ec
[REGCTRL] READ OK Addr: 0x0000009d Data: 0xcafebabedeadbeef112233d8000002ed
[REGCTRL] READ OK Addr: 0x0000009e Data: 0xcafebabedeadbeef112233d8000002ee
[REGCTRL] READ OK Addr: 0x0000009f Data: 0xcafebabedeadbeef112233d8000002ef
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x0000009c
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000a0
[REGCTRL] WRITE 0x000000a0 = 0xcafebabedeadbeef112233e400000220
[REGCTRL] WRITE 0x000000a1 = 0xcafebabedeadbeef112233e400000221
[REGCTRL] WRITE 0x000000a2 = 0xcafebabedeadbeef112233e400000222
[REGCTRL] WRITE 0x000000a3 = 0xcafebabedeadbeef112233e400000223
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000a0
[REGCTRL] READ request for addr 0x000000a0
[REGCTRL] READ request for addr 0x000000a1
[REGCTRL] READ request for addr 0x000000a2
[REGCTRL] READ request for addr 0x000000a3
[REGCTRL] READ OK Addr: 0x000000a0 Data: 0xcafebabedeadbeef112233e400000220
[REGCTRL] READ OK Addr: 0x000000a1 Data: 0xcafebabedeadbeef112233e400000221
[REGCTRL] READ OK Addr: 0x000000a2 Data: 0xcafebabedeadbeef112233e400000222
[REGCTRL] READ OK Addr: 0x000000a3 Data: 0xcafebabedeadbeef112233e400000223
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000a0
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000a4
[REGCTRL] WRITE 0x000000a4 = 0xcafebabedeadbeef112233e000000234
[REGCTRL] WRITE 0x000000a5 = 0xcafebabedeadbeef112233e000000235
[REGCTRL] WRITE 0x000000a6 = 0xcafebabedeadbeef112233e000000236
[REGCTRL] WRITE 0x000000a7 = 0xcafebabedeadbeef112233e000000237
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000a4
[REGCTRL] READ request for addr 0x000000a4
[REGCTRL] READ request for addr 0x000000a5
[REGCTRL] READ request for addr 0x000000a6
[REGCTRL] READ request for addr 0x000000a7

[REGCTRL] READ OK Addr: 0x000000a4 Data: 0xcafebabedeadbeef112233e000000234
[REGCTRL] READ OK Addr: 0x000000a5 Data: 0xcafebabedeadbeef112233e000000235
[REGCTRL] READ OK Addr: 0x000000a6 Data: 0xcafebabedeadbeef112233e000000236
[REGCTRL] READ OK Addr: 0x000000a7 Data: 0xcafebabedeadbeef112233e000000237
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000a4
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000a8
[REGCTRL] WRITE 0x000000a8 = 0xcafebabedeadbeef112233ec00000208
[REGCTRL] WRITE 0x000000a9 = 0xcafebabedeadbeef112233ec00000209
[REGCTRL] WRITE 0x000000aa = 0xcafebabedeadbeef112233ec0000020a
[REGCTRL] WRITE 0x000000ab = 0xcafebabedeadbeef112233ec0000020b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000a8
[REGCTRL] READ request for addr 0x000000a8
[REGCTRL] READ request for addr 0x000000a9
[REGCTRL] READ request for addr 0x000000aa
[REGCTRL] READ request for addr 0x000000ab
[REGCTRL] READ OK Addr: 0x000000a8 Data: 0xcafebabedeadbeef112233ec00000208
[REGCTRL] READ OK Addr: 0x000000a9 Data: 0xcafebabedeadbeef112233ec00000209
[REGCTRL] READ OK Addr: 0x000000aa Data: 0xcafebabedeadbeef112233ec0000020a
[REGCTRL] READ OK Addr: 0x000000ab Data: 0xcafebabedeadbeef112233ec0000020b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000a8
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000ac
[REGCTRL] WRITE 0x000000ac = 0xcafebabedeadbeef112233e80000021c
[REGCTRL] WRITE 0x000000ad = 0xcafebabedeadbeef112233e80000021d
[REGCTRL] WRITE 0x000000ae = 0xcafebabedeadbeef112233e80000021e
[REGCTRL] WRITE 0x000000af = 0xcafebabedeadbeef112233e80000021f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000ac
[REGCTRL] READ request for addr 0x000000ac
[REGCTRL] READ request for addr 0x000000ad
[REGCTRL] READ request for addr 0x000000ae
[REGCTRL] READ request for addr 0x000000af
[REGCTRL] READ OK Addr: 0x000000ac Data: 0xcafebabedeadbeef112233e80000021c
[REGCTRL] READ OK Addr: 0x000000ad Data: 0xcafebabedeadbeef112233e80000021d

[REGCTRL] READ OK Addr: 0x000000ae Data: 0xcafebabedeadbeef112233e80000021e
[REGCTRL] READ OK Addr: 0x000000af Data: 0xcafebabedeadbeef112233e80000021f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000ac
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000b0
[REGCTRL] WRITE 0x000000b0 = 0xcafebabedeadbeef112233f400000270
[REGCTRL] WRITE 0x000000b1 = 0xcafebabedeadbeef112233f400000271
[REGCTRL] WRITE 0x000000b2 = 0xcafebabedeadbeef112233f400000272
[REGCTRL] WRITE 0x000000b3 = 0xcafebabedeadbeef112233f400000273
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000b0
[REGCTRL] READ request for addr 0x000000b0
[REGCTRL] READ request for addr 0x000000b1
[REGCTRL] READ request for addr 0x000000b2
[REGCTRL] READ request for addr 0x000000b3
[REGCTRL] READ OK Addr: 0x000000b0 Data: 0xcafebabedeadbeef112233f400000270
[REGCTRL] READ OK Addr: 0x000000b1 Data: 0xcafebabedeadbeef112233f400000271
[REGCTRL] READ OK Addr: 0x000000b2 Data: 0xcafebabedeadbeef112233f400000272
[REGCTRL] READ OK Addr: 0x000000b3 Data: 0xcafebabedeadbeef112233f400000273
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000b0
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000b4
[REGCTRL] WRITE 0x000000b4 = 0xcafebabedeadbeef112233f000000264
[REGCTRL] WRITE 0x000000b5 = 0xcafebabedeadbeef112233f000000265
[REGCTRL] WRITE 0x000000b6 = 0xcafebabedeadbeef112233f000000266
[REGCTRL] WRITE 0x000000b7 = 0xcafebabedeadbeef112233f000000267
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000b4
[REGCTRL] READ request for addr 0x000000b4
[REGCTRL] READ request for addr 0x000000b5
[REGCTRL] READ request for addr 0x000000b6
[REGCTRL] READ request for addr 0x000000b7
[REGCTRL] READ OK Addr: 0x000000b4 Data: 0xcafebabedeadbeef112233f000000264
[REGCTRL] READ OK Addr: 0x000000b5 Data: 0xcafebabedeadbeef112233f000000265
[REGCTRL] READ OK Addr: 0x000000b6 Data: 0xcafebabedeadbeef112233f000000266
[REGCTRL] READ OK Addr: 0x000000b7 Data: 0xcafebabedeadbeef112233f000000267

[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000b4
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000b8
[REGCTRL] WRITE 0x000000b8 = 0xcafebabedeadbeef112233fc00000258
[REGCTRL] WRITE 0x000000b9 = 0xcafebabedeadbeef112233fc00000259
[REGCTRL] WRITE 0x000000ba = 0xcafebabedeadbeef112233fc0000025a
[REGCTRL] WRITE 0x000000bb = 0xcafebabedeadbeef112233fc0000025b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000b8
[REGCTRL] READ request for addr 0x000000b8
[REGCTRL] READ request for addr 0x000000b9
[REGCTRL] READ request for addr 0x000000ba
[REGCTRL] READ request for addr 0x000000bb
[REGCTRL] READ OK Addr: 0x000000b8 Data: 0xcafebabedeadbeef112233fc00000258
[REGCTRL] READ OK Addr: 0x000000b9 Data: 0xcafebabedeadbeef112233fc00000259
[REGCTRL] READ OK Addr: 0x000000ba Data: 0xcafebabedeadbeef112233fc0000025a
[REGCTRL] READ OK Addr: 0x000000bb Data: 0xcafebabedeadbeef112233fc0000025b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000b8
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000bc
[REGCTRL] WRITE 0x000000bc = 0xcafebabedeadbeef112233f80000024c
[REGCTRL] WRITE 0x000000bd = 0xcafebabedeadbeef112233f80000024d
[REGCTRL] WRITE 0x000000be = 0xcafebabedeadbeef112233f80000024e
[REGCTRL] WRITE 0x000000bf = 0xcafebabedeadbeef112233f80000024f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000bc
[REGCTRL] READ request for addr 0x000000bc
[REGCTRL] READ request for addr 0x000000bd
[REGCTRL] READ request for addr 0x000000be
[REGCTRL] READ request for addr 0x000000bf
[REGCTRL] READ OK Addr: 0x000000bc Data: 0xcafebabedeadbeef112233f80000024c
[REGCTRL] READ OK Addr: 0x000000bd Data: 0xcafebabedeadbeef112233f80000024d
[REGCTRL] READ OK Addr: 0x000000be Data: 0xcafebabedeadbeef112233f80000024e
[REGCTRL] READ OK Addr: 0x000000bf Data: 0xcafebabedeadbeef112233f80000024f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000bc

[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000c0
[REGCTRL] WRITE 0x000000c0 = 0xcafebabedeadbeef11223384000003c0
[REGCTRL] WRITE 0x000000c1 = 0xcafebabedeadbeef11223384000003c1
[REGCTRL] WRITE 0x000000c2 = 0xcafebabedeadbeef11223384000003c2
[REGCTRL] WRITE 0x000000c3 = 0xcafebabedeadbeef11223384000003c3
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000c0
[REGCTRL] READ request for addr 0x000000c0
[REGCTRL] READ request for addr 0x000000c1
[REGCTRL] READ request for addr 0x000000c2
[REGCTRL] READ request for addr 0x000000c3
[REGCTRL] READ OK Addr: 0x000000c0 Data: 0xcafebabedeadbeef11223384000003c0
[REGCTRL] READ OK Addr: 0x000000c1 Data: 0xcafebabedeadbeef11223384000003c1
[REGCTRL] READ OK Addr: 0x000000c2 Data: 0xcafebabedeadbeef11223384000003c2
[REGCTRL] READ OK Addr: 0x000000c3 Data: 0xcafebabedeadbeef11223384000003c3
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000c0
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000c4
[REGCTRL] WRITE 0x000000c4 = 0xcafebabedeadbeef11223380000003d4
[REGCTRL] WRITE 0x000000c5 = 0xcafebabedeadbeef11223380000003d5
[REGCTRL] WRITE 0x000000c6 = 0xcafebabedeadbeef11223380000003d6
[REGCTRL] WRITE 0x000000c7 = 0xcafebabedeadbeef11223380000003d7
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000c4
[REGCTRL] READ request for addr 0x000000c4
[REGCTRL] READ request for addr 0x000000c5
[REGCTRL] READ request for addr 0x000000c6
[REGCTRL] READ request for addr 0x000000c7
[REGCTRL] READ OK Addr: 0x000000c4 Data: 0xcafebabedeadbeef11223380000003d4
[REGCTRL] READ OK Addr: 0x000000c5 Data: 0xcafebabedeadbeef11223380000003d5
[REGCTRL] READ OK Addr: 0x000000c6 Data: 0xcafebabedeadbeef11223380000003d6
[REGCTRL] READ OK Addr: 0x000000c7 Data: 0xcafebabedeadbeef11223380000003d7
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000c4
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000c8

[REGCTRL] WRITE 0x000000c8 = 0xcafebabedeadbeef1122338c000003e8
[REGCTRL] WRITE 0x000000c9 = 0xcafebabedeadbeef1122338c000003e9
[REGCTRL] WRITE 0x000000ca = 0xcafebabedeadbeef1122338c000003ea
[REGCTRL] WRITE 0x000000cb = 0xcafebabedeadbeef1122338c000003eb
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000c8
[REGCTRL] READ request for addr 0x000000c8
[REGCTRL] READ request for addr 0x000000c9
[REGCTRL] READ request for addr 0x000000ca
[REGCTRL] READ request for addr 0x000000cb
[REGCTRL] READ OK Addr: 0x000000c8 Data: 0xcafebabedeadbeef1122338c000003e8
[REGCTRL] READ OK Addr: 0x000000c9 Data: 0xcafebabedeadbeef1122338c000003e9
[REGCTRL] READ OK Addr: 0x000000ca Data: 0xcafebabedeadbeef1122338c000003ea
[REGCTRL] READ OK Addr: 0x000000cb Data: 0xcafebabedeadbeef1122338c000003eb
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000c8
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000cc
[REGCTRL] WRITE 0x000000cc = 0xcafebabedeadbeef11223388000003fc
[REGCTRL] WRITE 0x000000cd = 0xcafebabedeadbeef11223388000003fd
[REGCTRL] WRITE 0x000000ce = 0xcafebabedeadbeef11223388000003fe
[REGCTRL] WRITE 0x000000cf = 0xcafebabedeadbeef11223388000003ff
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000cc
[REGCTRL] READ request for addr 0x000000cc
[REGCTRL] READ request for addr 0x000000cd
[REGCTRL] READ request for addr 0x000000ce
[REGCTRL] READ request for addr 0x000000cf
[REGCTRL] READ OK Addr: 0x000000cc Data: 0xcafebabedeadbeef11223388000003fc
[REGCTRL] READ OK Addr: 0x000000cd Data: 0xcafebabedeadbeef11223388000003fd
[REGCTRL] READ OK Addr: 0x000000ce Data: 0xcafebabedeadbeef11223388000003fe
[REGCTRL] READ OK Addr: 0x000000cf Data: 0xcafebabedeadbeef11223388000003ff
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000cc
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000d0
[REGCTRL] WRITE 0x000000d0 = 0xcafebabedeadbeef1122339400000390
[REGCTRL] WRITE 0x000000d1 = 0xcafebabedeadbeef1122339400000391

[REGCTRL] WRITE 0x000000d2 = 0xcafebabedeadbeef1122339400000392
[REGCTRL] WRITE 0x000000d3 = 0xcafebabedeadbeef1122339400000393
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000d0
[REGCTRL] READ request for addr 0x000000d0
[REGCTRL] READ request for addr 0x000000d1
[REGCTRL] READ request for addr 0x000000d2
[REGCTRL] READ request for addr 0x000000d3
[REGCTRL] READ OK Addr: 0x000000d0 Data: 0xcafebabedeadbeef1122339400000390
[REGCTRL] READ OK Addr: 0x000000d1 Data: 0xcafebabedeadbeef1122339400000391
[REGCTRL] READ OK Addr: 0x000000d2 Data: 0xcafebabedeadbeef1122339400000392
[REGCTRL] READ OK Addr: 0x000000d3 Data: 0xcafebabedeadbeef1122339400000393
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000d0
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000d4
[REGCTRL] WRITE 0x000000d4 = 0xcafebabedeadbeef1122339000000384
[REGCTRL] WRITE 0x000000d5 = 0xcafebabedeadbeef1122339000000385
[REGCTRL] WRITE 0x000000d6 = 0xcafebabedeadbeef1122339000000386
[REGCTRL] WRITE 0x000000d7 = 0xcafebabedeadbeef1122339000000387
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000d4
[REGCTRL] READ request for addr 0x000000d4
[REGCTRL] READ request for addr 0x000000d5
[REGCTRL] READ request for addr 0x000000d6
[REGCTRL] READ request for addr 0x000000d7
[REGCTRL] READ OK Addr: 0x000000d4 Data: 0xcafebabedeadbeef1122339000000384
[REGCTRL] READ OK Addr: 0x000000d5 Data: 0xcafebabedeadbeef1122339000000385
[REGCTRL] READ OK Addr: 0x000000d6 Data: 0xcafebabedeadbeef1122339000000386
[REGCTRL] READ OK Addr: 0x000000d7 Data: 0xcafebabedeadbeef1122339000000387
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000d4
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000d8
[REGCTRL] WRITE 0x000000d8 = 0xcafebabedeadbeef1122339c000003b8
[REGCTRL] WRITE 0x000000d9 = 0xcafebabedeadbeef1122339c000003b9
[REGCTRL] WRITE 0x000000da = 0xcafebabedeadbeef1122339c000003ba
[REGCTRL] WRITE 0x000000db = 0xcafebabedeadbeef1122339c000003bb

[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000d8
[REGCTRL] READ request for addr 0x000000d8
[REGCTRL] READ request for addr 0x000000d9
[REGCTRL] READ request for addr 0x000000da
[REGCTRL] READ request for addr 0x000000db
[REGCTRL] READ OK Addr: 0x000000d8 Data: 0xcafebabedeadbeef1122339c000003b8
[REGCTRL] READ OK Addr: 0x000000d9 Data: 0xcafebabedeadbeef1122339c000003b9
[REGCTRL] READ OK Addr: 0x000000da Data: 0xcafebabedeadbeef1122339c000003ba
[REGCTRL] READ OK Addr: 0x000000db Data: 0xcafebabedeadbeef1122339c000003bb
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000d8
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000dc
[REGCTRL] WRITE 0x000000dc = 0xcafebabedeadbeef11223398000003ac
[REGCTRL] WRITE 0x000000dd = 0xcafebabedeadbeef11223398000003ad
[REGCTRL] WRITE 0x000000de = 0xcafebabedeadbeef11223398000003ae
[REGCTRL] WRITE 0x000000df = 0xcafebabedeadbeef11223398000003af
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000dc
[REGCTRL] READ request for addr 0x000000dc
[REGCTRL] READ request for addr 0x000000dd
[REGCTRL] READ request for addr 0x000000de
[REGCTRL] READ request for addr 0x000000df
[REGCTRL] READ OK Addr: 0x000000dc Data: 0xcafebabedeadbeef11223398000003ac
[REGCTRL] READ OK Addr: 0x000000dd Data: 0xcafebabedeadbeef11223398000003ad
[REGCTRL] READ OK Addr: 0x000000de Data: 0xcafebabedeadbeef11223398000003ae
[REGCTRL] READ OK Addr: 0x000000df Data: 0xcafebabedeadbeef11223398000003af
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000dc
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000e0
[REGCTRL] WRITE 0x000000e0 = 0xcafebabedeadbeef112233a400000360
[REGCTRL] WRITE 0x000000e1 = 0xcafebabedeadbeef112233a400000361
[REGCTRL] WRITE 0x000000e2 = 0xcafebabedeadbeef112233a400000362
[REGCTRL] WRITE 0x000000e3 = 0xcafebabedeadbeef112233a400000363
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.

[REGCTRL] Read trigger received. Starting burst read at addr 0x000000e0
[REGCTRL] READ request for addr 0x000000e0
[REGCTRL] READ request for addr 0x000000e1
[REGCTRL] READ request for addr 0x000000e2
[REGCTRL] READ request for addr 0x000000e3
[REGCTRL] READ OK Addr: 0x000000e0 Data: 0xcafebabedeadbeef112233a400000360
[REGCTRL] READ OK Addr: 0x000000e1 Data: 0xcafebabedeadbeef112233a400000361
[REGCTRL] READ OK Addr: 0x000000e2 Data: 0xcafebabedeadbeef112233a400000362
[REGCTRL] READ OK Addr: 0x000000e3 Data: 0xcafebabedeadbeef112233a400000363
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000e0
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000e4
[REGCTRL] WRITE 0x000000e4 = 0xcafebabedeadbeef112233a000000374
[REGCTRL] WRITE 0x000000e5 = 0xcafebabedeadbeef112233a000000375
[REGCTRL] WRITE 0x000000e6 = 0xcafebabedeadbeef112233a000000376
[REGCTRL] WRITE 0x000000e7 = 0xcafebabedeadbeef112233a000000377
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000e4
[REGCTRL] READ request for addr 0x000000e4
[REGCTRL] READ request for addr 0x000000e5
[REGCTRL] READ request for addr 0x000000e6
[REGCTRL] READ request for addr 0x000000e7
[REGCTRL] READ OK Addr: 0x000000e4 Data: 0xcafebabedeadbeef112233a000000374
[REGCTRL] READ OK Addr: 0x000000e5 Data: 0xcafebabedeadbeef112233a000000375
[REGCTRL] READ OK Addr: 0x000000e6 Data: 0xcafebabedeadbeef112233a000000376
[REGCTRL] READ OK Addr: 0x000000e7 Data: 0xcafebabedeadbeef112233a000000377
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000e4
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000e8
[REGCTRL] WRITE 0x000000e8 = 0xcafebabedeadbeef112233ac00000348
[REGCTRL] WRITE 0x000000e9 = 0xcafebabedeadbeef112233ac00000349
[REGCTRL] WRITE 0x000000ea = 0xcafebabedeadbeef112233ac0000034a
[REGCTRL] WRITE 0x000000eb = 0xcafebabedeadbeef112233ac0000034b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000e8
[REGCTRL] READ request for addr 0x000000e8

[REGCTRL] READ request for addr 0x000000e9
[REGCTRL] READ request for addr 0x000000ea
[REGCTRL] READ request for addr 0x000000eb
[REGCTRL] READ OK Addr: 0x000000e8 Data: 0xcafebabedeadbeef112233ac00000348
[REGCTRL] READ OK Addr: 0x000000e9 Data: 0xcafebabedeadbeef112233ac00000349
[REGCTRL] READ OK Addr: 0x000000ea Data: 0xcafebabedeadbeef112233ac0000034a
[REGCTRL] READ OK Addr: 0x000000eb Data: 0xcafebabedeadbeef112233ac0000034b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000e8
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000ec
[REGCTRL] WRITE 0x000000ec = 0xcafebabedeadbeef112233a80000035c
[REGCTRL] WRITE 0x000000ed = 0xcafebabedeadbeef112233a80000035d
[REGCTRL] WRITE 0x000000ee = 0xcafebabedeadbeef112233a80000035e
[REGCTRL] WRITE 0x000000ef = 0xcafebabedeadbeef112233a80000035f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000ec
[REGCTRL] READ request for addr 0x000000ec
[REGCTRL] READ request for addr 0x000000ed
[REGCTRL] READ request for addr 0x000000ee
[REGCTRL] READ request for addr 0x000000ef
[REGCTRL] READ OK Addr: 0x000000ec Data: 0xcafebabedeadbeef112233a80000035c
[REGCTRL] READ OK Addr: 0x000000ed Data: 0xcafebabedeadbeef112233a80000035d
[REGCTRL] READ OK Addr: 0x000000ee Data: 0xcafebabedeadbeef112233a80000035e
[REGCTRL] READ OK Addr: 0x000000ef Data: 0xcafebabedeadbeef112233a80000035f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000ec
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000f0
[REGCTRL] WRITE 0x000000f0 = 0xcafebabedeadbeef112233b400000330
[REGCTRL] WRITE 0x000000f1 = 0xcafebabedeadbeef112233b400000331
[REGCTRL] WRITE 0x000000f2 = 0xcafebabedeadbeef112233b400000332
[REGCTRL] WRITE 0x000000f3 = 0xcafebabedeadbeef112233b400000333
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000f0
[REGCTRL] READ request for addr 0x000000f0
[REGCTRL] READ request for addr 0x000000f1
[REGCTRL] READ request for addr 0x000000f2

[REGCTRL] READ request for addr 0x000000f3
[REGCTRL] READ OK Addr: 0x000000f0 Data: 0xcafebabedeadbeef112233b400000330
[REGCTRL] READ OK Addr: 0x000000f1 Data: 0xcafebabedeadbeef112233b400000331
[REGCTRL] READ OK Addr: 0x000000f2 Data: 0xcafebabedeadbeef112233b400000332
[REGCTRL] READ OK Addr: 0x000000f3 Data: 0xcafebabedeadbeef112233b400000333
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000f0
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000f4
[REGCTRL] WRITE 0x000000f4 = 0xcafebabedeadbeef112233b000000324
[REGCTRL] WRITE 0x000000f5 = 0xcafebabedeadbeef112233b000000325
[REGCTRL] WRITE 0x000000f6 = 0xcafebabedeadbeef112233b000000326
[REGCTRL] WRITE 0x000000f7 = 0xcafebabedeadbeef112233b000000327
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000f4
[REGCTRL] READ request for addr 0x000000f4
[REGCTRL] READ request for addr 0x000000f5
[REGCTRL] READ request for addr 0x000000f6
[REGCTRL] READ request for addr 0x000000f7
[REGCTRL] READ OK Addr: 0x000000f4 Data: 0xcafebabedeadbeef112233b000000324
[REGCTRL] READ OK Addr: 0x000000f5 Data: 0xcafebabedeadbeef112233b000000325
[REGCTRL] READ OK Addr: 0x000000f6 Data: 0xcafebabedeadbeef112233b000000326
[REGCTRL] READ OK Addr: 0x000000f7 Data: 0xcafebabedeadbeef112233b000000327
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000f4
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000f8
[REGCTRL] WRITE 0x000000f8 = 0xcafebabedeadbeef112233bc00000318
[REGCTRL] WRITE 0x000000f9 = 0xcafebabedeadbeef112233bc00000319
[REGCTRL] WRITE 0x000000fa = 0xcafebabedeadbeef112233bc0000031a
[REGCTRL] WRITE 0x000000fb = 0xcafebabedeadbeef112233bc0000031b
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000f8
[REGCTRL] READ request for addr 0x000000f8
[REGCTRL] READ request for addr 0x000000f9
[REGCTRL] READ request for addr 0x000000fa
[REGCTRL] READ request for addr 0x000000fb
[REGCTRL] READ OK Addr: 0x000000f8 Data: 0xcafebabedeadbeef112233bc00000318

```
[REGCTRL] READ OK   Addr: 0x000000f9 Data: 0xcafebabedeadbeef112233bc00000319
[REGCTRL] READ OK   Addr: 0x000000fa Data: 0xcafebabedeadbeef112233bc0000031a
[REGCTRL] READ OK   Addr: 0x000000fb Data: 0xcafebabedeadbeef112233bc0000031b
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000f8
[REGCTRL] Returning to IDLE.
[REGCTRL] Write trigger received. Starting burst write at addr 0x000000fc
[REGCTRL] WRITE 0x000000fc = 0xcafebabedeadbeef112233b80000030c
[REGCTRL] WRITE 0x000000fd = 0xcafebabedeadbeef112233b80000030d
[REGCTRL] WRITE 0x000000fe = 0xcafebabedeadbeef112233b80000030e
[REGCTRL] WRITE 0x000000ff = 0xcafebabedeadbeef112233b80000030f
[REGCTRL] Write burst complete.
[REGCTRL] Returning to IDLE.
[REGCTRL] Read trigger received. Starting burst read at addr 0x000000fc
[REGCTRL] READ request for addr 0x000000fc
[REGCTRL] READ request for addr 0x000000fd
[REGCTRL] READ request for addr 0x000000fe
[REGCTRL] READ request for addr 0x000000ff
[REGCTRL] READ OK   Addr: 0x000000fc Data: 0xcafebabedeadbeef112233b80000030c
[REGCTRL] READ OK   Addr: 0x000000fd Data: 0xcafebabedeadbeef112233b80000030d
[REGCTRL] READ OK   Addr: 0x000000fe Data: 0xcafebabedeadbeef112233b80000030e
[REGCTRL] READ OK   Addr: 0x000000ff Data: 0xcafebabedeadbeef112233b80000030f
[REGCTRL] Burst read complete and verified.
[PASS] Verified burst read @ 0x000000fc

================ DDR3 Register Interface Stress Test Complete ================
Total Burst Blocks Tested: 64
  Passes: 64
  Fails : 0
================================================================================
==========
[REGCTRL] Returning to IDLE.
$finish called from file "ddr3_top.v", line 106.
$finish at simulation time             25085000
           V C S   S i m u l a t i o n   R e p o r t
Time: 25085000 ps
CPU Time:      0.180 seconds;       Data structure size:   0.0Mb
Mon Apr 21 17:04:22 2025
```