**Inputs:**

* clk: Clock signal, used to synchronize all actions in the design.
* rst: Reset signal, to reset the state machine and registers.
* i\_cmd\_valid: Input signal to indicate if a command is valid.
* i\_cmd: 16-bit input command (first cycle and second cycle).
* i\_cmd\_id: 16-bit input command ID.
* i\_addr: 40-bit input address (contains row and column address information).
* i\_cmd\_param: 32-bit input command parameter (additional control data for the command).
* i\_keep\_wait: Input signal indicating if the controller should remain in the WAIT state.
* i\_rready: Input signal indicating if there is space available for reading data.
* i\_rb\_n: NAND ready/busy signal. Indicates if the NAND flash is ready for operations.
* i\_dqs: Data strobe signal, used to synchronize data transfer.
* i\_dq: 32-bit data input from the NAND flash.

**Outputs:**

* o\_cmd\_ready: 1-bit output indicating if the controller is ready to accept a new command.
* o\_status: 2-bit status output, indicating IDLE, BUSY, WAIT, or READY state.
* o\_rvalid: 1-bit output indicating if read data is valid.
* o\_rdata: 32-bit output representing the read data from the NAND flash.
* o\_rlast: Output signal indicating the last piece of data in a read operation.
* o\_rid: 16-bit output representing the command ID for the read operation.
* o\_ruser: 16-bit output representing user-defined metadata for the read operation.
* io\_busy: 1-bit output indicating if the controller is busy.
* o\_ce\_n: Chip enable signal (active-low), used to select the NAND flash.
* o\_we\_n: Write enable signal, used to write data to the NAND flash.
* o\_cle: Command latch enable, used to send commands to the NAND flash.
* o\_ale: Address latch enable, used to send addresses to the NAND flash.
* o\_re: 4-bit output representing read enable signals.
* o\_dqs\_tri\_en: Tri-state enable signal for the data strobe, used to switch between input and output.
* o\_dq\_tri\_en: Tri-state enable signal for the data bus, used to switch between input and output.
* o\_dq: 32-bit output representing the data to be written to the NAND flash.

**localparam Definitions**

verilog

Copy code

localparam

IDLE = 10'b00\_0000\_0001,

CMD1 = 10'b00\_0000\_0010,

ADDR = 10'b00\_0000\_0100,

CMD2 = 10'b00\_0000\_1000,

BUSY = 10'b00\_0001\_0000,

LOCK = 10'b00\_0010\_0000,

WAIT = 10'b00\_0100\_0000,

RPRE = 10'b00\_1000\_0000,

DATA = 10'b01\_0000\_0000,

RPST = 10'b10\_0000\_0000;

**Explaination:**

* **Purpose**: localparam is used to define constants that are local to this module.
* These constants represent the states of a finite state machine (FSM).
  + IDLE: The controller is idle.
  + CMD1: The controller is in the first command phase.
  + ADDR: The controller is in the address phase, sending addresses to the NAND flash.
  + CMD2: The second command phase.
  + BUSY: The controller is waiting for the NAND flash to become ready.
  + LOCK: The controller is waiting for the i\_rb\_n signal to change.
  + WAIT: The controller is waiting for an external condition.
  + RPRE: The controller is preparing to read data.
  + DATA: The controller is reading data from the NAND flash.
  + RPST: The controller is finishing the read operation.

**Registers**

**Verilog Code**

reg [ 9:0] state;

reg has\_cmd2;

reg [ 2:0] addr\_num;

reg [11:0] busy\_time;

reg [14:0] data\_num;

**Explaination**

* **Registers** store values over time in hardware. They hold state between clock cycles.
  + state: A 10-bit register representing the current state of the FSM.
  + has\_cmd2: A 1-bit flag indicating if there is a second command.
  + addr\_num: A 3-bit register indicating how many addresses need to be sent.
  + busy\_time: A 12-bit register representing the number of cycles the controller should wait while the NAND flash is busy.
  + data\_num: A 15-bit register representing the number of data bytes to be read.
* **State Transitions and Behavior**
* The rest of the code describes the behavior of the controller based on the state it is in. This is done using always blocks, which define how registers and outputs change on each clock cycle.

**always @(posedge clk or posedge rst)**

Verilog Code:

always @(posedge clk or posedge rst) begin

if (rst) begin

o\_cmd\_ready <= 1'b1;

io\_busy <= 1'b0;

o\_status <= 2'b00;

state <= IDLE;

has\_cmd2 <= 1'b0;

addr\_num <= 3'b0;

busy\_time <= 12'b0;

data\_num <= 15'b0;

end else begin

case (state)

IDLE: begin

o\_cmd\_ready <= 1'b1;

io\_busy <= 1'b0;

o\_status <= 2'b00;

if (i\_cmd\_valid) begin

state <= CMD1;

o\_cmd\_ready <= 1'b0;

has\_cmd2 <= (i\_cmd\_param[0] == 1'b1);

addr\_num <= i\_cmd\_param[3:1];

data\_num <= i\_cmd\_param[18:4];

end

end

**Reset and Initialization:**

* This block is sensitive to the **positive edge** of the clock (posedge clk) and the reset (rst).
  + If rst is active (rst == 1), it resets all the registers to their initial values:
    - o\_cmd\_ready: Command ready signal is set to 1, indicating the controller is ready to accept a new command.
    - io\_busy: Busy flag is set to 0 (controller is not busy).
    - o\_status: Status register is set to 00 (IDLE state).
    - state: The state machine moves to IDLE.
    - has\_cmd2: Flag indicating if there is a second command is set to 0.
    - addr\_num: Number of addresses to send is set to 0.
    - busy\_time: The busy timer is reset to 0.
    - data\_num: The number of data bytes to read is reset.

**IDLE State:**

* When the reset is not active (rst == 0), the state machine begins in the IDLE state.
  + In this state, the controller is waiting for a command.
  + o\_cmd\_ready is set to 1, meaning the controller is ready to accept a new command.
  + io\_busy is 0 because the controller is not performing any operations.
  + If i\_cmd\_valid is 1, indicating a new command has been issued, the state machine transitions to the CMD1 state.
    - o\_cmd\_ready is set to 0, indicating the controller is busy handling the command.
    - has\_cmd2: Set to 1 if the least significant bit of i\_cmd\_param is 1, indicating that the command has a second part.
    - addr\_num: This is the number of address cycles (from bits [3:1] of i\_cmd\_param).
    - data\_num: This is the number of data cycles (from bits [18:4] of i\_cmd\_param).

**CMD1 State: Sending First Command**

**Verilog Code:**

CMD1: begin

o\_cle <= 1'b1;

o\_ale <= 1'b0;

o\_dq <= {16'b0, i\_cmd};

o\_dq\_tri\_en <= 1'b1;

if (i\_keep\_wait) begin

state <= WAIT;

end else if (!i\_rb\_n) begin

state <= CMD2;

o\_cle <= 1'b0;

end else begin

state <= ADDR;

end

end

* In the CMD1 state:
  + o\_cle (Command Latch Enable) is set to 1, enabling the command latch so the NAND flash can receive a command.
  + o\_ale (Address Latch Enable) is 0, meaning we are not sending addresses in this state.
  + o\_dq: The output data bus is loaded with the command (i\_cmd) and padded with zeros in the upper 16 bits ({16'b0, i\_cmd}).
  + o\_dq\_tri\_en is set to 1, enabling the output on the data bus.
* The next state is determined based on the conditions:
  + If i\_keep\_wait is high, the controller moves to the WAIT state.
  + If i\_rb\_n (Ready/Busy signal) is low, indicating the NAND flash is not ready, the controller moves to the CMD2 state.
  + Otherwise, it moves to the ADDR state, where it sends addresses to the NAND flash.

**ADDR State: Sending Address**

Verilog Code

ADDR: begin

o\_ale <= 1'b1;

o\_cle <= 1'b0;

o\_dq <= i\_addr[addr\_num \* 8 +: 8];

addr\_num <= addr\_num - 1;

if (addr\_num == 3'b0) begin

state <= CMD2;

end

end

* In the ADDR state:
  + o\_ale is set to 1, enabling the address latch to send the address to the NAND flash.
  + o\_cle is set to 0, disabling the command latch.
  + The address (i\_addr) is split into 8-bit chunks and sent over the data bus (o\_dq).
  + addr\_num is decremented with each cycle until all address cycles are complete.
* Once all addresses are sent (addr\_num == 0), the state machine transitions to the CMD2 state.

**CMD2 State: Sending Second Command**

**Verilog Code**

CMD2: begin

if (has\_cmd2) begin

o\_cle <= 1'b1;

o\_ale <= 1'b0;

o\_dq <= {16'b0, i\_cmd};

o\_dq\_tri\_en <= 1'b1;

end

if (!i\_rb\_n) begin

state <= BUSY;

o\_cle <= 1'b0;

end

end

* In the CMD2 state:
  + If has\_cmd2 is 1, the second command is sent.
  + o\_cle is set to 1, enabling the command latch to send the second command.
  + The command is placed on the data bus (o\_dq), and the data tri-state is enabled.
* If i\_rb\_n is low, indicating the NAND flash is not ready, the controller moves to the BUSY state to wait for the NAND flash to become ready.

**BUSY State: Waiting for NAND Flash**

**Verilog Code**

BUSY: begin

if (i\_rb\_n) begin

if (data\_num == 15'b0) begin

state <= IDLE;

o\_status <= 2'b11;

end else begin

state <= RPRE;

end

end

end

* In the BUSY state:
  + The controller waits for the NAND flash to become ready (i\_rb\_n == 1).
  + If data\_num is 0, meaning no data needs to be read, the controller returns to the IDLE state and sets o\_status to 11 (READY).
  + Otherwise, it transitions to the RPRE (Read Prepare) state to begin reading data.

**RPRE State: Preparing to Read**

**Verilog Code**

RPRE: begin

o\_dqs\_tri\_en <= 1'b0;

o\_dq\_tri\_en <= 1'b0;

state <= DATA;

end

* In the RPRE state:
  + The data strobe tri-state (o\_dqs\_tri\_en) and data bus tri-state (o\_dq\_tri\_en) are disabled, setting the signals to input mode to receive data from the NAND flash.
  + The controller then transitions to the DATA state to begin reading data.

**DATA State: Reading Data**

Verilog Code

DATA: begin

if (i\_rready) begin

o\_rvalid <= 1'b1;

o\_rdata <= i\_dq;

o\_rid <= i\_cmd\_id;

o\_ruser <= i\_cmd\_param[31:16];

data\_num <= data\_num - 1;

if (data\_num == 15'b0) begin

state <= RPST;

end

end

end

* In the DATA state:
  + If i\_rready is 1, indicating that there is space to receive data, the controller:
    - Sets o\_rvalid to 1, indicating that valid data is available.
    - Loads the data from the NAND flash (i\_dq) into o\_rdata.
    - Sets o\_rid to the command ID and o\_ruser to user-defined metadata from the command parameter.
    - Decrements data\_num, counting how many pieces of data have been read.
* If all the data has been read (data\_num == 0), the state machine transitions to the RPST (Read Post) state.

**RPST State: Finishing Read**

Verilog Code

RPST: begin

o\_rvalid <= 1'b0;

state <= IDLE;

o\_status <= 2'b11;

end

* In the RPST state:
  + The controller clears the o\_rvalid signal, indicating that no more valid data is available.
  + The state machine returns to the IDLE state, and the o\_status is set to 11 (READY).

**Block 1: io\_busy Signal Control**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

io\_busy <= 1'h0;

end else if((state == IDLE) || (state == LOCK) || (state == WAIT)) begin

io\_busy <= 1'h0;

end else begin

io\_busy <= 1'h1;

end

**Purpose:**

The io\_busy signal indicates whether the system is busy performing an operation. When io\_busy is high, the system is actively working, and when low, it's idle or waiting.

**Detailed Explanation:**

* **Reset condition (rst):** On reset, io\_busy is set to 0 (inactive).
* **Idle, Lock, or Wait states:** In the IDLE, LOCK, or WAIT states, the system is not actively working, so io\_busy is set to 0.
* **Other states:** For all other states, io\_busy is set to 1, indicating that the system is busy performing a task.

**Block 2: Write Enable (o\_we\_n)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_we\_n <= 1'h1;

end else if(((state == CMD1) || (state == ADDR) || (state == CMD2)) && is\_we\_edge) begin

o\_we\_n <= ~o\_we\_n;

end

**Purpose:**

The o\_we\_n signal is used to control the write enable signal for the memory interface. The write enable signal is typically active low, meaning when o\_we\_n is 0, writing to memory is enabled.

**Detailed Explanation:**

* **Reset condition:** On reset, o\_we\_n is set to 1, disabling the write operation.
* **CMD1, ADDR, and CMD2 states:** During the CMD1, ADDR, or CMD2 states, if is\_we\_edge is high (which may indicate a write condition or edge), the value of o\_we\_n is toggled (~o\_we\_n). This controls the timing of the write enable signal.

**Block 3: Command Latch Enable (o\_cle)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_cle <= 1'h0;

end else if((state == CMD1) || (state == CMD2)) begin

o\_cle <= 1'h1;

end else begin

o\_cle <= 1'h0;

end

**Purpose:**

The o\_cle (Command Latch Enable) signal is used to control whether the command latch should be active. It's typically set high during command phases.

**Detailed Explanation:**

* **Reset condition:** On reset, o\_cle is set to 0, meaning the command latch is disabled.
* **CMD1 and CMD2 states:** During the CMD1 and CMD2 states, o\_cle is set to 1, enabling the command latch to capture the command.
* **Other states:** In all other states, o\_cle is set to 0, disabling the command latch.

**Block 4: Address Latch Enable (o\_ale)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_ale <= 1'h0;

end else if(state == ADDR) begin

o\_ale <= 1'h1;

end else begin

o\_ale <= 1'h0;

end

**Purpose:**

The o\_ale (Address Latch Enable) signal controls the address latch. It should be active when the system is latching an address.

**Detailed Explanation:**

* **Reset condition:** On reset, o\_ale is set to 0, disabling the address latch.
* **ADDR state:** In the ADDR state, o\_ale is set to 1, allowing the address to be latched.
* **Other states:** In other states, the address latch is disabled (o\_ale = 0).

**Block 5: Read Enable (o\_re)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_re <= 4'hf;

end else if(((state == RPRE) && (rpre\_cnt > 8'h1)) || (state == RPST)) begin

o\_re <= 4'h0;

end else if((state == DATA) & (~i\_rready)) begin

o\_re <= {4{o\_re[3]}};

end else if(state == DATA) begin

o\_re <= 4'h5;

end else begin

o\_re <= 4'hf;

end

**Purpose:**

The o\_re signal controls the read enable lines. It is dynamically adjusted based on the state to enable or disable reading.

**Detailed Explanation:**

* **Reset condition:** On reset, o\_re is set to 4'hf, indicating that all read enables are inactive.
* **RPRE and RPST states:** In the RPRE (Read Pre) state with a counter condition or in the RPST (Read Post) state, o\_re is set to 0, enabling the read operation.
* **DATA state with i\_rready low:** If the system is in the DATA state but the i\_rready signal is low (indicating the system is not ready to receive data), the read enable is held constant based on the most significant bit of o\_re[3].
* **DATA state with i\_rready high:** In the DATA state, if i\_rready is high, o\_re is set to 4'h5, enabling the read lines in a specific pattern for data transfer.

**Block 6: Chip Enable (o\_ce\_n)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_ce\_n <= 1'h1;

end else if(((state == IDLE) & (~i\_cmd\_valid)) || ((state == RPST) && (rpst\_cnt >= `tRPST)) || (state == LOCK) || (state == WAIT)) begin

o\_ce\_n <= 1'h1;

end else begin

o\_ce\_n <= 1'h0;

end

**Purpose:**

The o\_ce\_n signal controls the chip enable, which activates or deactivates the memory chip.

**Detailed Explanation:**

* **Reset condition:** On reset, o\_ce\_n is set to 1, disabling the chip.
* **IDLE, RPST, LOCK, WAIT states:** In the IDLE state with i\_cmd\_valid low (no valid command), the RPST state with a counter condition, or the LOCK or WAIT states, the chip is disabled (o\_ce\_n = 1).
* **Other states:** In all other states, the chip is enabled by setting o\_ce\_n to 0.

**Block 7: Data Strobe (o\_dqs)**

Verilog Code

assign o\_dqs = 4'hf;

**Purpose:**

The o\_dqs signal is assigned a fixed value of 4'hf. It may represent an idle or inactive state for the data strobe.

**Block 8: Data Strobe Tri-State Enable (o\_dqs\_tri\_en)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_dqs\_tri\_en <= 1'h0;

end else if(((state == RPRE) && (rpre\_cnt >= `tDQSRH)) || (state == DATA) || (state == RPST)) begin

o\_dqs\_tri\_en <= 1'h1;

end else begin

o\_dqs\_tri\_en <= 1'h0;

end

**Purpose:**

The o\_dqs\_tri\_en signal controls the tri-state behavior of the data strobe lines (dqs), determining whether they should be active or floating (disconnected).

**Detailed Explanation:**

* **Reset condition:** On reset, o\_dqs\_tri\_en is set to 0, disabling the tri-state (data strobe is not floating).
* **RPRE, DATA, RPST states:** In the RPRE, DATA, or RPST states, the o\_dqs\_tri\_en signal is set to 1, enabling the tri-state (data strobe lines are floating).
* **Other states:** In other states, o\_dqs\_tri\_en is set to 0.

**Block 9: Data Output Register (o\_dq)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_dq <= 32'h0;

end else if(state == CMD1) begin

o\_dq <= {4{i\_cmd[7:0]}};

end else if(state == CMD2) begin

o\_dq <= {4{i\_cmd[15:8]}};

end else if(state == ADDR) begin

o\_dq <= {4{dq\_addr}};

end else begin

o\_dq <= 32'h0;

end

**Purpose:**  
o\_dq is used to output data or address information to the memory or external devices.

**Detailed Explanation:**

* **Reset condition:** When rst is high, o\_dq is set to 32'h0, initializing the data output to zero.
* **CMD1 state:** When the system is in the CMD1 state, o\_dq is set to {4{i\_cmd[7:0]}}. This replicates the lower 8 bits of i\_cmd into a 32-bit wide output.
* **CMD2 state:** When the system is in the CMD2 state, o\_dq is set to {4{i\_cmd[15:8]}}. This replicates the upper 8 bits of i\_cmd into a 32-bit wide output.
* **ADDR state:** When the system is in the ADDR state, o\_dq is set to {4{dq\_addr}}. This outputs the address, dq\_addr, which is an 8-bit address potentially used to interact with memory.
* **Other states:** For all other states, o\_dq is set to 32'h0, indicating no valid data.

**Block 10: Tri-State Enable for Data Lines (o\_dq\_tri\_en)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_dq\_tri\_en <= 1'h1;

end else if((state == CMD1) || (state == ADDR) || (state == CMD2)) begin

o\_dq\_tri\_en <= 1'h0; // output

end else begin

o\_dq\_tri\_en <= 1'h1;

end

**Purpose:**  
o\_dq\_tri\_en controls whether the data lines are in a high-impedance state (tri-state) or actively driving values.

**Detailed Explanation:**

* **Reset condition:** When rst is high, o\_dq\_tri\_en is set to 1'h1, enabling the tri-state (high-impedance) mode.
* **CMD1, ADDR, CMD2 states:** In these states, o\_dq\_tri\_en is set to 1'h0, which enables the output on the data lines (o\_dq is actively driving values).
* **Other states:** For all other states, o\_dq\_tri\_en is set to 1'h1, putting the data lines into tri-state (high-impedance mode).

**Block 11: Address Calculation (dq\_addr)**

Verilog Code

wire [7:0] dq\_addr;

assign dq\_addr = (i\_addr >> {addr\_cnt, 3'h0}) & 8'hff;

**Purpose:**  
dq\_addr is used to compute a modified address based on the input address and a counter.

**Detailed Explanation:**

* **Address Calculation:** dq\_addr is computed by shifting i\_addr right by addr\_cnt bits (where addr\_cnt is combined with 3'h0 to form the shift amount) and then masking it with 8'hff to ensure it is 8 bits wide.

**Block 12: Data Strobe Output Enable (dqs\_out\_en)**

Verilog Code

wire dqs\_out\_en;

assign dqs\_out\_en = ((state == DATA) || (state == RPST));

**Purpose:**  
dqs\_out\_en indicates whether the data strobe (DQS) should be enabled based on the current state.

**Detailed Explanation:**

* **Enable Condition:** dqs\_out\_en is set to 1 if the system is in the DATA or RPST states. This signal controls the enabling of the data strobe.

**Block 13: Data Input Counter (din\_cnt)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

din\_cnt <= 4'h0;

end else if(state == RPRE) begin

din\_cnt <= 4'h0;

end else if(dqs\_out\_en && (i\_dqs == 4'h5) && (din\_cnt < WARMUP\_DATA\_NUM)) begin

din\_cnt <= din\_cnt + 4'h4;

end

**Purpose:**  
din\_cnt counts the number of data input cycles, used to manage timing and readiness of the data input.

**Detailed Explanation:**

* **Reset condition:** When rst is high, din\_cnt is reset to 4'h0.
* **RPRE state:** When in the RPRE state, din\_cnt is reset to 4'h0.
* **Data Input Counting:** When dqs\_out\_en is true and i\_dqs is 4'h5, and din\_cnt is less than WARMUP\_DATA\_NUM, din\_cnt is incremented by 4'h4. This keeps track of the number of data input cycles.

**Block 14: Output Validity and Data (o\_rvalid, o\_rdata, o\_rid, o\_ruser)**

Verilog Code

always@(posedge clk or posedge rst)

if(rst) begin

o\_rvalid <= 1'h0;

o\_rdata <= 'h0;

o\_rid <= 16'h0;

o\_ruser <= 16'h0;

end else if(dqs\_out\_en && (i\_dqs == 4'h5) && (din\_cnt >= WARMUP\_DATA\_NUM)) begin

o\_rvalid <= 1'b1;

o\_rdata <= i\_dq;

o\_rid <= i\_cmd\_id;

o\_ruser <= i\_cmd;

end else begin

o\_rvalid <= 1'h0;

end

**Purpose:**  
This block controls the output validity and data transmission for the read operation.

**Detailed Explanation:**

* **Reset condition:** When rst is high, o\_rvalid, o\_rdata, o\_rid, and o\_ruser are reset to 0 or their default values.
* **Output Validity and Data:** When dqs\_out\_en is true, i\_dqs is 4'h5, and din\_cnt is greater than or equal to WARMUP\_DATA\_NUM, o\_rvalid is set to 1, indicating that the read data is valid. o\_rdata, o\_rid, and o\_ruser are set to i\_dq, i\_cmd\_id, and i\_cmd, respectively, providing the data, ID, and user information.

**Block 15: DQS Register (dqs\_r)**

Verilog Code

reg [3:0] dqs\_r;

always@(posedge clk or posedge rst)

if(rst) begin

dqs\_r <= 4'hf;

end else if(state == RPST) begin

dqs\_r <= i\_dqs;

end else begin

dqs\_r <= 4'hf;

end

**Purpose:**  
dqs\_r holds the previous value of the data strobe signal (i\_dqs) for comparison purposes.

**Detailed Explanation:**

* **Reset condition:** When rst is high, dqs\_r is set to 4'hf.
* **RPST state:** In the RPST state, dqs\_r is updated with i\_dqs.
* **Other states:** For all other states, dqs\_r is set to 4'hf.

**Block 16: Read Last (o\_rlast)**

Verilog Code

assign o\_rlast = (i\_dqs == 4'h0) && (dqs\_r == 4'h5);

**Purpose:**  
o\_rlast signals whether the current read operation is the last one based on the data strobe signal (i\_dqs) and its previous value (dqs\_r).

**Detailed Explanation:**

* **Read Last Condition:** o\_rlast is set to 1 if i\_dqs is 4'h0 (indicating the end of data strobe) and dqs\_r was 4'h5 (indicating the previous state of the strobe). This indicates the end of a read operation.