The testbench failed to compile. Please fix the module. The output of iverilog is as follows:

warning: Port 2 (shiftVal) of barrel_shift_32 expects 5 bits, got 32.
Pruning 27 high bits of the expression.

The warning message indicates that there is a width mismatch between the `shiftVal` port of the `barrel_shift_32` module and the signal being assigned to it. The `shiftVal` port expects 5 bits, but the signal being assigned has 32 bits. This results in the upper 27 bits being pruned.

To fix this issue, you need to ensure that the signal assigned to `shiftVal` is also 5 bits wide. Here's how you can modify the `barrel_shift_32_w_rotation_ctrl` module to handle this:

```
// 8-bit Barrel Shifter Module
module barrel_shift_8(
    input   [7:0] inputData,
    input   [2:0] shiftVal,
    output reg [7:0] outputData
);

    // Use a case statement to simplify the logic
    always @(inputData, shiftVal) begin
        case (shiftVal)
            3'b000: outputData = inputData; // No shift
            3'b001: outputData = {inputData[6:0], inputData[^7]}; // Shift left by 1
            3'b010: outputData = {inputData[5:0], inputData[7:6]}; // Shift left by 2
            3'b011: outputData = {inputData[4:0], inputData[7:5]}; // Shift left by 3
            3'b100: outputData = {inputData[3:0], inputData[7:4]}; // Shift left by 4
            3'b101: outputData = {inputData[2:0], inputData[7:3]}; // Shift left by 5
            3'b110: outputData = {inputData[1:0], inputData[7:2]}; // Shift left by 6
            3'b111: outputData = {inputData[^0], inputData[7:1]}; // Shift left by 7
            default: outputData = inputData; // Default case, no shift
        endcase
    end

endmodule

// 32-bit Barrel Shifter Module
module barrel_shift_32(
    input   [31:0] inputData,
    input   [4:0] shiftVal,
    output reg [31:0] outputData
);

    // Divide the 32-bit input into four 8-bit segments
    wire [7:0] segment1, segment2, segment3, segment4;
    assign segment1 = inputData[31:24];
    assign segment2 = inputData[23:16];
    assign segment3 = inputData[15:8];
    assign segment4 = inputData[7:0];

    // Outputs from each 8-bit shifter
    wire [7:0] shifted1, shifted2, shifted3, shifted4;

    // Instantiate 8-bit shifters for each segment
    barrel_shift_8 shifter1(segment1, shiftVal[2:0], shifted1);
    barrel_shift_8 shifter2(segment2, shiftVal[2:0], shifted2);
    barrel_shift_8 shifter3(segment3, shiftVal[2:0], shifted3);
    barrel_shift_8 shifter4(segment4, shiftVal[2:0], shifted4);

    // Handle shifts that cross segment boundaries
    always @(inputData, shiftVal) begin
        if (shiftVal < 8) begin
            // Shift within the same segment
            outputData = {shifted1, shifted2, shifted3, shifted4};
        end else if (shiftVal < 16) begin
            // Shift across segment boundaries
            outputData = {shifted2, shifted3, shifted4, shifted1};
        end else if (shiftVal < 24) begin
            outputData = {shifted3, shifted4, shifted1, shifted2};
        end else begin
            outputData = {shifted4, shifted1, shifted2, shifted3};
        end
    end

endmodule

// 32-bit Barrel Shifter with Rotation Control Module
module barrel_shift_32_w_rotation_ctrl(
    input   sel_left_or_right_rotate, // 1 for left rotation, 0 for right rotation
    input   [31:0] inputData,
    input   [4:0] shiftVal, // Ensure this is 5 bits wide
    output reg [31:0] outputData
);

    // Use the existing 32-bit barrel shifter for left rotation
    wire [31:0] leftRotatedData;
    barrel_shift_32 leftShifter(inputData, shiftVal, leftRotatedData);

    // Perform right rotation by shifting left by (32 - shiftVal)
    wire [31:0] rightRotatedData;
    barrel_shift_32 rightShifter(inputData, 5'b0 - shiftVal, rightRotatedData);

    // Select between left and right rotation based on sel_left_or_right_rotate
    always @(inputData, shiftVal, sel_left_or_right_rotate) begin
        if (sel_left_or_right_rotate) begin
            outputData = leftRotatedData;
        end else begin
            outputData = rightRotatedData;
        end
    end
```

```
endmodule
```

However, the warning you're seeing suggests that somewhere in your testbench or other parts of your code, you're assigning a 32-bit signal to `shiftVal`, which is expected to be 5 bits. You need to ensure that the signal being assigned to `shiftVal` is also 5 bits wide.

For example, if you have a 32-bit signal `large_shiftVal`, you should only use the lower 5 bits when assigning it to `shiftVal`:

```
input [31:0] large_shiftVal;
barrel_shift_32_w_rotation_ctrl instance(sel_left_or_right_rotate, inputData, large_shift
```

This ensures that only the lower 5 bits of `large_shiftVal` are used, matching the expected width of `shiftVal`.

⁂