Chapter 3: Operators and Expressions

Overview

SystemVerilog provides a rich set of operators for performing various operations on data. Understanding these operators and their precedence is crucial for writing efficient and correct SystemVerilog code. This chapter covers all major operator categories with practical examples.

Arithmetic Operators

Arithmetic operators perform mathematical operations on numeric values.

Basic Arithmetic Operators

Operator	Description	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
%	Modulus	a % b
**	Exponentiation	a ** b

Example 1: Basic Arithmetic Operations

Design under Test (DUT)

```
// arithmetic_example.sv
module arithmetic;
    logic [7:0] a = 8'd25;
    logic [7:0] b = 8'd5;
    logic [15:0] result;

    logic signed [7:0] x = -8'd10;
    logic signed [7:0] y = 8'd3;
    logic signed [15:0] signed_result;

    initial begin
        $display();

        // Unsigned arithmetic operations
        // cast to 16-bit to match result width
        $display("a = %d", a);
        $display("b = %d", b);

        result = 16'(a + b);     // result = 30
        $display("a + b = %d", result);
```

```
        result = 16'(a - b);     // result = 20
        $display("a - b = %d", result);

        result = 16'(a * b);     // result = 125
        $display("a * b = %d", result);

        result = 16'(a / b);     // result = 5
        $display("a / b = %d", result);

        result = 16'(a % b);     // result = 0 (25 % 5)
        $display("a %% b = %d", result);

        result = 16'(a ** 2);    // result = 625 (25^2)
        $display("a ** 2 = %d", result);

        // Signed arithmetic operations
        // cast to 16-bit signed to match signed_result width
        $display();
        $display("x = %d", x);
        $display("y = %d", y);

        signed_result = 16'(signed'(x + y));  // -7
        $display("x + y = %d", signed_result);

        signed_result = 16'(signed'(x / y));  // -3 (truncated toward zero)
        $display("x / y = %d", signed_result);

        $display();
    end
endmodule
```

Design Unit Test (DUT) Testbench

```
// arithmetic_testbench.sv
module arithmetic_testbench;
    // Instantiate the design under test
    arithmetic DESIGN_INSTANCE();

    // Test control
    initial begin
        // Enable VCD dumping
        $dumpfile("arithmetic_testbench.vcd");
        $dumpvars(0, arithmetic_testbench);

        // Wait for the arithmetic module to complete
        #20;

        // Finish simulation
        $finish;
    end
endmodule
```

```
from verilator_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_1__arithmetic/")
```

```
Docker Compose Output:
================================================================================
make: Entering directory '/work/obj_dir'
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated.o
/usr/local/share/verilator/include/verilated.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated_vcd_c.o
/usr/local/share/verilator/include/verilated_vcd_c.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_timing.o /usr/local/share/verilator/include/verilated_timing.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_threads.o /usr/local/share/verilator/include/verilated_threads.cpp
python3 /usr/local/share/verilator/bin/verilator_includer
-DVL_INCLUDE_OPT=include Varithmetic_testbench.cpp
Varithmetic_testbench___024root__DepSet_hadb3a670__0.cpp
Varithmetic_testbench___024root__DepSet_h930df209__0.cpp
Varithmetic_testbench__main.cpp Varithmetic_testbench__Trace__0.cpp
Varithmetic_testbench___024root__Slow.cpp
Varithmetic_testbench___024root__DepSet_h930df209__0__Slow.cpp
Varithmetic_testbench__Syms.cpp Varithmetic_testbench__Trace__0__Slow.cpp
Varithmetic_testbench__TraceDecls__0__Slow.cpp > Varithmetic_testbench__ALL.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
Varithmetic_testbench__ALL.o Varithmetic_testbench__ALL.cpp
```

```
echo "" > Varithmetic_testbench__ALL.verilator_deplist.tmp
g++    verilated.o verilated_vcd_c.o verilated_timing.o verilated_threads.o
Varithmetic_testbench__ALL.a    -pthread -lpthread -latomic   -o
Varithmetic_testbench
rm Varithmetic_testbench__ALL.verilator_deplist.tmp
make: Leaving directory '/work/obj_dir'
- V e r i l a t i o n   R e p o r t: Verilator 5.036 2025-04-27 rev v5.036
- Verilator: Built from 0.037 MB sources in 3 modules, into 0.036 MB in 10 C++
files needing 0.000 MB
- Verilator: Walltime 46.765 s (elab=0.001, cvt=0.037, bld=46.580); cpu 0.024 s
on 1 threads; alloced 20.180 MB

a =  25
b =   5
a + b =    30
a - b =    20
a * b =   125
a / b =     5
a % b =     0
a ** 2 =   625

x =  -10
y =    3
x + y =     -7
x / y =     -3

- arithmetic_testbench.sv:16: Verilog $finish
- S i m u l a t i o n   R e p o r t: Verilator 5.036 2025-04-27
- Verilator: $finish at 20ps; walltime 0.009 s; speed 10.871 ns/s
- Verilator: cpu 0.002 s on 1 threads; alloced 25 MB
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_1__arithmetic/obj_dir directory...
Chapter_3_examples/example_1__arithmetic/obj_dir removed successfully.

0
```

```
from gtkwave_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_1__arithmetic/")
```

```
Docker Compose Output:
================================================================================
 Container notebooks-verilator-1  Starting
 Container notebooks-verilator-1  Started

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI

[0] start time.
[20] end time.
WM Destroy
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_1__arithmetic/obj_dir directory...
Chapter_3_examples/example_1__arithmetic/obj_dir removed successfully.

0
```

Important Notes

- Division by zero results in 'x' (unknown)
- Integer division truncates toward zero
- Modulus result has the same sign as the first operand

Logical and Bitwise Operators

Logical Operators

Logical operators work on entire expressions and return 1-bit results.

Operator	Description	Example
&&	Logical AND	(a > 0) && (b < 10)
||	Logical OR	(a == 0) || (b == 0)
!	Logical NOT	!(a == b)

Bitwise Operators

Bitwise operators work on individual bits of operands.

Operator	Description	Example
&	Bitwise AND	a & b
|	Bitwise OR	a | b
^	Bitwise XOR	a ^ b
~	Bitwise NOT	~a
~&	Bitwise NAND	~&a or ~(a & b)
~|	Bitwise NOR	~|a or ~(a | b)
~^ or ^~	Bitwise XNOR	~^a or a ~^ b

Example

Design under Test (DUT)

```
// bitwise.sv
module bitwise;
    logic [3:0] a = 4'b1010;
    logic [3:0] b = 4'b1100;
    logic [3:0] result;
    logic logical_result;

    initial begin
        #10; // Wait for 10 time units before starting operations
        $display("Starting logical and bitwise operations...");
        $display("Initial values: a = %b, b = %b", a, b);
        $display();

        $display("a = %b", a);
        $display("b = %b", b);

        // Bitwise operations
        result = a & b;     // 4'b1000
        $display("a & b = %b (bitwise AND)", result);
```

```
        result = a | b;     // 4'b1110
        $display("a | b = %b (bitwise OR)", result);

        result = a ^ b;     // 4'b0110
        $display("a ^ b = %b (bitwise XOR)", result);

        result = ~a;        // 4'b0101
        $display("~a = %b (bitwise NOT)", result);

        result = 4'(~&a);   // 1'b1 (NAND of all bits) - cast to 4-bit
        $display("~&a = %b (reduction NAND)", result);

        $display();

        // Logical operations
        logical_result = (a > 0) && (b > 0);  // 1'b1
        $display("(a > 0) && (b > 0) = %b (logical AND)", logical_result);

        logical_result = (a == 0) || (b == 0); // 1'b0
        $display("(a == 0) || (b == 0) = %b (logical OR)", logical_result);

        logical_result = !(a == b);            // 1'b1
        $display("!(a == b) = %b (logical NOT)", logical_result);

        $display();
        $display("All operations completed successfully!");
    end
endmodule
```

Design Unit Test (DUT) Testbench

```
// bitwise_example_testbench.sv
module bitwise_testbench;  // Testbench module
  bitwise DESIGN_INSTANCE();  // Instantiate design under test

  initial begin
    // Dump waves
    $dumpfile("bitwise_testbench.vcd");       // Specify the VCD file
    $dumpvars(0, bitwise_testbench);          // Dump all variables in the test module
    #1;                                                    // Wait for a time unit
    $display();                                            // Display empty line
    $display("Hello from logical bitwise testbench!");       // Display message
    $display("Testing logical and bitwise operations...");
    $display();                                            // Display empty line

    // Wait for design to complete its operations
    #10;
    $display();

    // End simulation
    $finish;
  end

endmodule
```

```
from verilator_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_2__bitwise/")
```

```
Docker Compose Output:
================================================================================
make: Entering directory '/work/obj_dir'
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated.o
/usr/local/share/verilator/include/verilated.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated_vcd_c.o
/usr/local/share/verilator/include/verilated_vcd_c.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_timing.o /usr/local/share/verilator/include/verilated_timing.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_threads.o /usr/local/share/verilator/include/verilated_threads.cpp
python3 /usr/local/share/verilator/bin/verilator_includer
-DVL_INCLUDE_OPT=include Vbitwise_testbench.cpp
Vbitwise_testbench___024root__DepSet_h169e4b74__0.cpp
Vbitwise_testbench___024root__DepSet_h2a73bd68__0.cpp
Vbitwise_testbench__main.cpp Vbitwise_testbench__Trace__0.cpp
Vbitwise_testbench___024root__Slow.cpp
Vbitwise_testbench___024root__DepSet_h2a73bd68__0__Slow.cpp
Vbitwise_testbench__Syms.cpp Vbitwise_testbench__Trace__0__Slow.cpp
Vbitwise_testbench__TraceDecls__0__Slow.cpp > Vbitwise_testbench__ALL.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
Vbitwise_testbench__ALL.o Vbitwise_testbench__ALL.cpp
```

```
echo "" > Vbitwise_testbench__ALL.verilator_deplist.tmp
g++    verilated.o verilated_vcd_c.o verilated_timing.o verilated_threads.o
Vbitwise_testbench__ALL.a    -pthread -lpthread -latomic   -o Vbitwise_testbench
rm Vbitwise_testbench__ALL.verilator_deplist.tmp
make: Leaving directory '/work/obj_dir'
- V e r i l a t i o n   R e p o r t: Verilator 5.036 2025-04-27 rev v5.036
- Verilator: Built from 0.037 MB sources in 3 modules, into 0.036 MB in 10 C++
files needing 0.000 MB
- Verilator: Walltime 25.352 s (elab=0.001, cvt=0.048, bld=25.149); cpu 0.026 s
on 1 threads; alloced 20.172 MB

Hello from logical bitwise testbench!
Testing logical and bitwise operations...

Starting logical and bitwise operations...
Initial values: a = 1010, b = 1100

a = 1010
b = 1100
a & b = 1000 (bitwise AND)
a | b = 1110 (bitwise OR)
a ^ b = 0110 (bitwise XOR)
~a = 0101 (bitwise NOT)
~&a = 0001 (reduction NAND)

(a > 0) && (b > 0) = 1 (logical AND)
(a == 0) || (b == 0) = 0 (logical OR)
!(a == b) = 1 (logical NOT)

All operations completed successfully!

- bitwise_testbench.sv:20: Verilog $finish
- S i m u l a t i o n   R e p o r t: Verilator 5.036 2025-04-27
- Verilator: $finish at 11ps; walltime 0.007 s; speed 6.546 ns/s
- Verilator: cpu 0.002 s on 1 threads; alloced 25 MB
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_2__bitwise/obj_dir directory...
Chapter_3_examples/example_2__bitwise/obj_dir removed successfully.
```

```
0
```

```
from gtkwave_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_2__bitwise/")
```

```
Docker Compose Output:
================================================================================
 Container notebooks-verilator-1  Recreate
 Container notebooks-verilator-1  Recreated
 Container notebooks-verilator-1  Starting
 Container notebooks-verilator-1  Started

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI

[0] start time.
[11] end time.
```

```
WM Destroy
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_2__bitwise/obj_dir directory...
Chapter_3_examples/example_2__bitwise/obj_dir removed successfully.

0
```

Reduction Operators

Reduction operators perform operations across all bits of a single operand, returning a 1-bit result.

Operator	Description	Equivalent
&	Reduction AND	&a = a[0] & a[1] & ... & a[n]
|	Reduction OR	|a = a[0] | a[1] | ... | a[n]
^	Reduction XOR	^a = a[0] ^ a[1] ^ ... ^ a[n]
~&	Reduction NAND	~(&a)
~|	Reduction NOR	~(|a)
~^ or ^~	Reduction XNOR	~(^a)

Example 3: Reduction Operators

Design under Test (DUT)

```
// reduction.sv
module reduction;
    logic [7:0] data = 8'b11010010;
    logic result;

    initial begin
        #10; // Wait for 10 time units before starting operations
        $display();
        $display("Starting reduction operations...");
        $display("Input data: %b (decimal %d)", data, data);
        $display("Bit count analysis: %d ones, %d zeros", $countones(data), 8-$countones(data
        $display();

        // Reduction operations
        result = &data;   // 1'b0 (not all bits are 1)
        $display("&data = %b (AND reduction - all bits 1?)", result);

        result = |data;   // 1'b1 (at least one bit is 1)
        $display("|data = %b (OR reduction - any bit 1?)", result);

        result = ^data;   // 1'b1 (odd number of 1s - parity)
        $display("^data = %b (XOR reduction - odd parity?)", result);

        result = ~&data;  // 1'b1 (NAND - not all bits are 1)
        $display("~&data = %b (NAND reduction - not all bits 1?)", result);

        result = ~|data;  // 1'b0 (NOR - not all bits are 0)
        $display("~|data = %b (NOR reduction - all bits 0?)", result);

        result = ~^data;  // 1'b0 (XNOR - even parity)
        $display("~^data = %b (XNOR reduction - even parity?)", result);
```

```
        $display();

        // Additional test with different data patterns
        $display("=== Testing with different patterns ===");

        // Test with all 1s
        data = 8'b11111111;
        $display("All 1s (%b): &=%b |=%b ^=%b", data, &data, |data, ^data);

        // Test with all 0s
        data = 8'b00000000;
        $display("All 0s (%b): &=%b |=%b ^=%b", data, &data, |data, ^data);

        // Test with single 1
        data = 8'b00000001;
        $display("Single 1 (%b): &=%b |=%b ^=%b", data, &data, |data, ^data);

        // Test with even number of 1s
        data = 8'b11000011;
        $display("Even 1s (%b): &=%b |=%b ^=%b", data, &data, |data, ^data);

        $display();
        $display("All reduction operations completed successfully!");
        $display();
    end
endmodule
```

Design Unit Test (DUT) Testbench

```
// reduction_testbench.sv
module reduction_testbench;  // Testbench module
  reduction DESIGN_INSTANCE();  // Instantiate design under test

  // Variable for verification
  logic [7:0] original_data = 8'b11010010;

  initial begin
    // Dump waves
    $dumpfile("reduction_testbench.vcd");       // Specify the VCD file
    $dumpvars(0, reduction_testbench);          // Dump all variables in the test module

    // Wait for design to complete its operations
    #20;  // Longer wait since we have multiple test patterns

    // End simulation
    $finish;
  end

endmodule
```

```
from verilator_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_3__reduction/")
```

```
Docker Compose Output:
================================================================================
make: Entering directory '/work/obj_dir'
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated.o
/usr/local/share/verilator/include/verilated.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated_vcd_c.o
/usr/local/share/verilator/include/verilated_vcd_c.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_timing.o /usr/local/share/verilator/include/verilated_timing.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_threads.o /usr/local/share/verilator/include/verilated_threads.cpp
python3 /usr/local/share/verilator/bin/verilator_includer
-DVL_INCLUDE_OPT=include Vreduction_testbench.cpp
Vreduction_testbench___024root__DepSet_h2fcf9133__0.cpp
Vreduction_testbench___024root__DepSet_h077ad171__0.cpp
Vreduction_testbench__main.cpp Vreduction_testbench__Trace__0.cpp
Vreduction_testbench___024root__Slow.cpp
Vreduction_testbench___024root__DepSet_h077ad171__0__Slow.cpp
Vreduction_testbench__Syms.cpp Vreduction_testbench__Trace__0__Slow.cpp
Vreduction_testbench__TraceDecls__0__Slow.cpp > Vreduction_testbench__ALL.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
Vreduction_testbench__ALL.o Vreduction_testbench__ALL.cpp
echo "" > Vreduction_testbench__ALL.verilator_deplist.tmp
g++    verilated.o verilated_vcd_c.o verilated_timing.o verilated_threads.o
Vreduction_testbench__ALL.a    -pthread -lpthread -latomic   -o
Vreduction_testbench
```

```
rm Vreduction_testbench__ALL.verilator_deplist.tmp
make: Leaving directory '/work/obj_dir'
- V e r i l a t i o n   R e p o r t: Verilator 5.036 2025-04-27 rev v5.036
- Verilator: Built from 0.038 MB sources in 3 modules, into 0.040 MB in 10 C++
files needing 0.000 MB
- Verilator: Walltime 30.064 s (elab=0.001, cvt=0.056, bld=29.787); cpu 0.042 s
on 1 threads; alloced 20.176 MB

Starting reduction operations...
Input data: 11010010 (decimal 210)
Bit count analysis:           4 ones,           4 zeros

&data = 0 (AND reduction - all bits 1?)
|data = 1 (OR reduction - any bit 1?)
^data = 0 (XOR reduction - odd parity?)
~&data = 1 (NAND reduction - not all bits 1?)
~|data = 0 (NOR reduction - all bits 0?)
~^data = 1 (XNOR reduction - even parity?)

=== Testing with different patterns ===
All 1s (11111111): &=1 |=1 ^=0
All 0s (00000000): &=0 |=0 ^=0
Single 1 (00000001): &=0 |=1 ^=1
Even 1s (11000011): &=0 |=1 ^=0

All reduction operations completed successfully!

- reduction_testbench.sv:17: Verilog $finish
- S i m u l a t i o n   R e p o r t: Verilator 5.036 2025-04-27
- Verilator: $finish at 20ps; walltime 0.005 s; speed 12.535 ns/s
- Verilator: cpu 0.002 s on 1 threads; alloced 25 MB
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_3__reduction/obj_dir directory...
Chapter_3_examples/example_3__reduction/obj_dir removed successfully.
```

```
0
```

```
from gtkwave_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_3__reduction/")
```

```
Docker Compose Output:
================================================================================
 Container notebooks-verilator-1  Recreate
 Container notebooks-verilator-1  Recreated
 Container notebooks-verilator-1  Starting
 Container notebooks-verilator-1  Started

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI

[0] start time.
[20] end time.
WM Destroy
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_3__reduction/obj_dir directory...
```

```
Chapter_3_examples/example_3__reduction/obj_dir removed successfully.
0
```

Shift Operators

Shift operators move bits left or right within a vector.

Logical Shift Operators

Operator	Description	Fill bits
<<	Logical left shift	Zeros from right
>>	Logical right shift	Zeros from left

Arithmetic Shift Operators

Operator	Description	Fill bits
<<<	Arithmetic left shift	Zeros from right
>>>	Arithmetic right shift	Sign bit from left

Example

Design under Test (DUT)

```
// shift.sv
module shift;
    logic [7:0] data = 8'b10110100;
    logic signed [7:0] signed_data = 8'sb10110100; // -76 in decimal
    logic [7:0] result;
    logic signed [7:0] signed_result;

    // Move variable declaration to module level
    logic signed [7:0] pos_signed = 8'sb01110100; // +116

    initial begin
        #10; // Wait for initial setup
        $display();
        $display("Starting shift operations...");
        $display("Original data: %b (decimal %d)", data, data);
        $display("Signed data: %b (decimal %d)", signed_data, signed_data);
        $display();

        // Logical shifts
        result = data << 2;     // 8'b10110100 -> 8'b11010000
        $display("Logical left shift by 2:  %b << 2 = %b (decimal %d)", data, result, result)

        result = data >> 2;     // 8'b10110100 -> 8'b00101101
        $display("Logical right shift by 2: %b >> 2 = %b (decimal %d)", data, result, result)

        $display();

        // Arithmetic shifts
```

```
        result = data <<< 2;    // Same as logical left shift
        $display("Arithmetic left shift by 2:  %b <<< 2 = %b (decimal %d)", data, result, res

        signed_result = signed_data >>> 2; // Sign extension: 8'b11101101
        $display("Arithmetic right shift by 2: %b >>> 2 = %b (decimal %d)", signed_data, sign

        $display();

        // Additional test patterns
        $display("=== Testing different shift amounts ===");

        // Test shift by 1
        $display("Shift by 1:");
        $display("  %b << 1 = %b", data, data << 1);
        $display("  %b >> 1 = %b", data, data >> 1);
        $display("  %b >>> 1 = %b (signed)", signed_data, signed_data >>> 1);

        // Test shift by 4
        $display("Shift by 4:");
        $display("  %b << 4 = %b", data, data << 4);
        $display("  %b >> 4 = %b", data, data >> 4);
        $display("  %b >>> 4 = %b (signed)", signed_data, signed_data >>> 4);

        $display();

        // Test with positive signed number
        $display("=== Testing with positive signed number ===");
        $display("Positive signed: %b (decimal %d)", pos_signed, pos_signed);
        $display("  >>> 2 = %b (decimal %d)", pos_signed >>> 2, pos_signed >>> 2);

        $display();
        $display("All shift operations completed successfully!");
    end
endmodule
```

Design Unit Test (DUT) Testbench

```
// shift_testbench.sv
module shift_testbench;  // Testbench module
  shift DESIGN_INSTANCE();  // Instantiate design under test

  // Variables for verification - moved to module level
  logic [7:0] test_data = 8'b10110100;
  logic signed [7:0] test_signed = 8'sb10110100;

  // Edge case variables - moved to module level
  logic [7:0] all_ones = 8'b11111111;
  logic [7:0] all_zeros = 8'b00000000;
  logic signed [7:0] neg_one = -1;

  initial begin
    // Dump waves
    $dumpfile("shift_testbench.vcd");       // Specify the VCD file
    $dumpvars(0, shift_testbench);          // Dump all variables in the test module
```

```
    #1;                                             // Wait for a time unit
    $display("Hello from shift operations testbench!");       // Display message
    $display("Testing all shift operators...");
    $display();                                     // Display empty line

    // Wait for design to complete its operations
    #20;  // Longer wait for multiple test patterns

    // Display final results from the design
    $display();
    $display("=== Test Results Summary ===");
    $display("Final unsigned result: %b (decimal %d)",
             DESIGN_INSTANCE.result, DESIGN_INSTANCE.result);
    $display("Final signed result: %b (decimal %d)",
             DESIGN_INSTANCE.signed_result, DESIGN_INSTANCE.signed_result);

    // Manual verification of shift operations
    $display();
    $display("=== Manual Verification ===");
    $display("Test data: %b (decimal %d)", test_data, test_data);

    $display("Logical shifts:");
    $display("  << 2: Expected %b, Manual calc: %b", test_data << 2, test_data << 2);
    $display("  >> 2: Expected %b, Manual calc: %b", test_data >> 2, test_data >> 2);

    $display("Arithmetic shifts:");
    $display("  <<< 2: Expected %b, Manual calc: %b", test_data <<< 2, test_data <<< 2);
    $display("  >>> 2: Expected %b, Manual calc: %b", test_signed >>> 2, test_signed >>> 2);

    // Bit-by-bit analysis
    $display();
    $display("=== Bit-by-bit Analysis ===");
    $display("Original: %b", test_data);
    $display("Positions: 76543210");
    $display("After << 2: %b (bits shifted left, zeros fill right)", test_data << 2);
    $display("After >> 2: %b (bits shifted right, zeros fill left)", test_data >> 2);

    $display();
    $display("Signed arithmetic right shift analysis:");
    $display("Original signed: %b (decimal %d)", test_signed, test_signed);
    $display("After >>> 2:     %b (decimal %d) - sign bit extended",
             test_signed >>> 2, test_signed >>> 2);
    $display("Sign bit (%b) extended to fill left positions", test_signed[7]);

    // Edge case testing
    $display();
    $display("=== Edge Case Testing ===");

    $display("All ones (%b):", all_ones);
    $display("  << 1 = %b", all_ones << 1);
    $display("  >> 1 = %b", all_ones >> 1);

    $display("All zeros (%b):", all_zeros);
    $display("  << 3 = %b", all_zeros << 3);
    $display("  >> 3 = %b", all_zeros >> 3);
```

```
    $display("Negative one (%b = %d):", neg_one, neg_one);
    $display("  >>> 1 = %b = %d", neg_one >>> 1, neg_one >>> 1);
    $display("  >>> 3 = %b = %d", neg_one >>> 3, neg_one >>> 3);

    $display("=== Test Completed Successfully ===");
    $display();

    // End simulation
    $finish;
  end

endmodule
```

```
from verilator_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_4__shift/")
```

```
Docker Compose Output:
================================================================================
make: Entering directory '/work/obj_dir'
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated.o
/usr/local/share/verilator/include/verilated.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated_vcd_c.o
/usr/local/share/verilator/include/verilated_vcd_c.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_timing.o /usr/local/share/verilator/include/verilated_timing.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_threads.o /usr/local/share/verilator/include/verilated_threads.cpp
python3 /usr/local/share/verilator/bin/verilator_includer
-DVL_INCLUDE_OPT=include Vshift_testbench.cpp
Vshift_testbench___024root__DepSet_he9fac9f1__0.cpp
```

```
Vshift_testbench___024root__DepSet_h9727e118__0.cpp Vshift_testbench__main.cpp
Vshift_testbench__Trace__0.cpp Vshift_testbench___024root__Slow.cpp
Vshift_testbench___024root__DepSet_h9727e118__0__Slow.cpp
Vshift_testbench__Syms.cpp Vshift_testbench__Trace__0__Slow.cpp
Vshift_testbench__TraceDecls__0__Slow.cpp > Vshift_testbench__ALL.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
Vshift_testbench__ALL.o Vshift_testbench__ALL.cpp
echo "" > Vshift_testbench__ALL.verilator_deplist.tmp
g++    verilated.o verilated_vcd_c.o verilated_timing.o verilated_threads.o
Vshift_testbench__ALL.a    -pthread -lpthread -latomic   -o Vshift_testbench
rm Vshift_testbench__ALL.verilator_deplist.tmp
make: Leaving directory '/work/obj_dir'
- V e r i l a t i o n   R e p o r t: Verilator 5.036 2025-04-27 rev v5.036
- Verilator: Built from 0.041 MB sources in 3 modules, into 0.038 MB in 10 C++
files needing 0.000 MB
- Verilator: Walltime 24.861 s (elab=0.001, cvt=0.032, bld=24.712); cpu 0.022 s
on 1 threads; alloced 20.184 MB
Hello from shift operations testbench!
Testing all shift operators...


Starting shift operations...
Original data: 10110100 (decimal 180)
Signed data: 10110100 (decimal  -76)

Logical left shift by 2:  10110100 << 2 = 11010000 (decimal 208)
Logical right shift by 2: 10110100 >> 2 = 00101101 (decimal  45)

Arithmetic left shift by 2:  10110100 <<< 2 = 11010000 (decimal 208)
Arithmetic right shift by 2: 10110100 >>> 2 = 11101101 (decimal  -19)

=== Testing different shift amounts ===
Shift by 1:
  10110100 << 1 = 01101000
  10110100 >> 1 = 01011010
  10110100 >>> 1 = 11011010 (signed)
Shift by 4:
  10110100 << 4 = 01000000
  10110100 >> 4 = 00001011
  10110100 >>> 4 = 11111011 (signed)

=== Testing with positive signed number ===
Positive signed: 01110100 (decimal  116)
  >>> 2 = 00011101 (decimal   29)

All shift operations completed successfully!

=== Test Results Summary ===
Final unsigned result: 11010000 (decimal 208)
Final signed result: 11101101 (decimal  -19)
```

```
=== Manual Verification ===
Test data: 10110100 (decimal 180)
Logical shifts:
  << 2: Expected 11010000, Manual calc: 11010000
  >> 2: Expected 00101101, Manual calc: 00101101
Arithmetic shifts:
  <<< 2: Expected 11010000, Manual calc: 11010000
  >>> 2: Expected 11101101, Manual calc: 11101101

=== Bit-by-bit Analysis ===
Original: 10110100
Positions: 76543210
After << 2: 11010000 (bits shifted left, zeros fill right)
After >> 2: 00101101 (bits shifted right, zeros fill left)

Signed arithmetic right shift analysis:
Original signed: 10110100 (decimal  -76)
After >>> 2:     11101101 (decimal  -19) - sign bit extended
Sign bit (1) extended to fill left positions

=== Edge Case Testing ===
All ones (11111111):
  << 1 = 11111110
  >> 1 = 01111111
All zeros (00000000):
  << 3 = 00000000
  >> 3 = 00000000
Negative one (11111111 =   -1):
  >>> 1 = 11111111 =   -1
  >>> 3 = 11111111 =   -1
=== Test Completed Successfully ===

- shift_testbench.sv:82: Verilog $finish
- S i m u l a t i o n   R e p o r t: Verilator 5.036 2025-04-27
- Verilator: $finish at 21ps; walltime 0.003 s; speed 13.617 ns/s
- Verilator: cpu 0.002 s on 1 threads; alloced 25 MB
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_4__shift/obj_dir directory...
Chapter_3_examples/example_4__shift/obj_dir removed successfully.
0
```

```
from gtkwave_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_4__shift/")
```

```
Docker Compose Output:
================================================================================
 Container notebooks-verilator-1  Recreate
 Container notebooks-verilator-1  Recreated
 Container notebooks-verilator-1  Starting
 Container notebooks-verilator-1  Started

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI
```

```
[0] start time.
[21] end time.
WM Destroy
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_4__shift/obj_dir directory...
Chapter_3_examples/example_4__shift/obj_dir removed successfully.

0
```

Comparison and Equality Operators

Equality Operators

Operator	Description	X/Z handling
==	Logical equality	X/Z → unknown result
!=	Logical inequality	X/Z → unknown result
===	Case equality	X/Z compared exactly
!==	Case inequality	X/Z compared exactly

Relational Operators

Operator	Description
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal

Example

Design under Test (DUT)

```
// comparison.sv
module comparison;
    logic [3:0] a = 4'b1010;  // 10 in decimal
    logic [3:0] b = 4'b1010;  // 10 in decimal
    logic [3:0] c = 4'b1x1z;  // Contains unknown (x) and high-impedance (z)
    logic [3:0] d = 4'b0110;  // 6 in decimal
    logic [3:0] e = 4'b1111;  // 15 in decimal
    logic result;

    // Additional test variables
    logic signed [3:0] pos_num = 4'sb0111;  // +7
    logic signed [3:0] neg_num = 4'sb1001;  // -7 (two's complement)
    logic [7:0] wide_val = 8'b00001010;     // Same value as 'a' but wider

    initial begin
        #10; // Wait for initial setup
        $display();
        $display("Starting comparison operations...");
        $display("Values: a=%b(%d), b=%b(%d), c=%b, d=%b(%d), e=%b(%d)",
                 a, a, b, b, c, d, d, e, e);
        $display("Signed values: pos_num=%b(%d), neg_num=%b(%d)",
```

```
                pos_num, pos_num, neg_num, neg_num);
        $display();

        // === EQUALITY COMPARISONS ===
        $display("=== Equality Comparisons ===");

        // Logical equality (==) and inequality (!=)
        result = (a == b);
        $display("a == b: %b == %b = %b (identical values)", a, b, result);

        result = (a != b);
        $display("a != b: %b != %b = %b (should be 0)", a, b, result);

        result = (a == d);
        $display("a == d: %b == %b = %b (different values)", a, d, result);

        result = (a != d);
        $display("a != d: %b != %b = %b (different values)", a, d, result);

        $display();

        // Comparisons with unknown values
        $display("=== Comparisons with Unknown Values ===");
        result = (a == c);
        $display("a == c: %b == %b = %b (unknown due to x/z in c)", a, c, result);

        result = (a != c);
        $display("a != c: %b != %b = %b (unknown due to x/z in c)", a, c, result);

        // Case equality (===) and case inequality (!==)
        result = (a === b);
        $display("a === b: %b === %b = %b (exact bit-by-bit match)", a, b, result);

        result = (a === c);
        $display("a === c: %b === %b = %b (exact comparison, no x/z match)", a, c, result);

        result = (c === c);
        $display("c === c: %b === %b = %b (identical including x/z)", c, c, result);

        result = (a !== c);
        $display("a !== c: %b !== %b = %b (case inequality)", a, c, result);

        $display();

        // === RELATIONAL COMPARISONS ===
        $display("=== Relational Comparisons ===");

        // Less than (<)
        result = (a < e);
        $display("a < e: %b(%d) < %b(%d) = %b", a, a, e, e, result);

        result = (d < a);
        $display("d < a: %b(%d) < %b(%d) = %b", d, d, a, a, result);

        result = (a < a);
```

```
        $display("a < a: %b(%d) < %b(%d) = %b (same values)", a, a, a, a, result);

        // Less than or equal (<=)
        result = (a <= b);
        $display("a <= b: %b(%d) <= %b(%d) = %b (equal values)", a, a, b, b, result);

        result = (d <= a);
        $display("d <= a: %b(%d) <= %b(%d) = %b", d, d, a, a, result);

        // Greater than (>)
        result = (e > a);
        $display("e > a: %b(%d) > %b(%d) = %b", e, e, a, a, result);

        result = (a > d);
        $display("a > d: %b(%d) > %b(%d) = %b", a, a, d, d, result);

        // Greater than or equal (>=)
        result = (a >= b);
        $display("a >= b: %b(%d) >= %b(%d) = %b (equal values)", a, a, b, b, result);

        result = (a >= d);
        $display("a >= d: %b(%d) >= %b(%d) = %b", a, a, d, d, result);

        $display();

        // === SIGNED COMPARISONS ===
        $display("=== Signed Number Comparisons ===");
        result = (pos_num > neg_num);
        $display("pos_num > neg_num: %b(%d) > %b(%d) = %b",
                 pos_num, pos_num, neg_num, neg_num, result);

        result = (neg_num < pos_num);
        $display("neg_num < pos_num: %b(%d) < %b(%d) = %b",
                 neg_num, neg_num, pos_num, pos_num, result);

        result = (neg_num < 4'sd0);
        $display("neg_num < 0: %b(%d) < 0 = %b", neg_num, neg_num, result);

        $display();

        // === WIDTH MISMATCHED COMPARISONS ===
        $display("=== Width Mismatched Comparisons ===");
        /* verilator lint_off WIDTHEXPAND */
        result = (a == wide_val);
        /* verilator lint_on WIDTHEXPAND */
        $display("a == wide_val: %b(%d) == %b(%d) = %b (different widths, same value)",
                 a, a, wide_val, wide_val, result);

        result = (a === wide_val[3:0]);
        $display("a === wide_val[3:0]: %b === %b = %b (same width after slicing)",
                 a, wide_val[3:0], result);

        $display();

        // === EDGE CASES ===
```

```
        $display("=== Edge Cases ===");

        // All zeros and all ones
        result = (4'b0000 == 4'd0);
        $display("4'b0000 == 4'd0: %b", result);

        result = (4'b1111 == 4'd15);
        $display("4'b1111 == 4'd15: %b", result);

        // Comparison with unknown results
        result = (c > d);
        $display("c > d: %b > %b(%d) = %b (unknown due to x/z)", c, d, d, result);

        result = (c < d);
        $display("c < d: %b < %b(%d) = %b (unknown due to x/z)", c, d, d, result);

        $display();

        // === CHAINED COMPARISONS ===
        $display("=== Chained Comparisons ===");
        result = (d < a) && (a < e);
        $display("(d < a) && (a < e): (%b < %b) && (%b < %b) = %b && %b = %b",
                 d, a, a, e, (d < a), (a < e), result);

        result = (a >= d) || (a <= b);
        $display("(a >= d) || (a <= b): (%b >= %b) || (%b <= %b) = %b || %b = %b",
                 a, d, a, b, (a >= d), (a <= b), result);

        $display();
        $display("All comparison operations completed successfully!");
    end
endmodule
```

Design Unit Test (DUT) Testbench

```
// comparison_testbench.sv
module comparison_testbench;  // Testbench module
  comparison DESIGN_INSTANCE();  // Instantiate design under test

  // Test variables - declared at module level
  logic [3:0] test_a = 4'b1010;
  logic [3:0] test_b = 4'b1010;
  logic [3:0] test_c = 4'b1x1z;
  logic [3:0] test_d = 4'b0110;
  logic [3:0] test_small = 4'b0001;
  logic [3:0] test_large = 4'b1111;
  logic signed [3:0] test_pos = 4'sb0111;
  logic signed [3:0] test_neg = 4'sb1001;

  // Results storage
  logic eq_result, neq_result, lt_result, gt_result;
  logic case_eq_result, case_neq_result;

  // Loop variable for testing
```

```
    integer i;
    logic temp_result;

    initial begin
      // Dump waves
      $dumpfile("comparison_testbench.vcd");
      $dumpvars(0, comparison_testbench);
      #1;

      $display("Hello from comparison operations testbench!");
      $display("Testing all comparison operators...");
      $display();

      // Wait for design to complete
      #30;

      $display();
      $display("=== Testbench Verification ===");
      $display("Final result from design: %b", DESIGN_INSTANCE.result);

      // Manual verification of comparison operations
      $display();
      $display("=== Manual Verification Tests ===");

      // Test equality operators
      eq_result = (test_a == test_b);
      neq_result = (test_a != test_d);
      case_eq_result = (test_a === test_b);
      case_neq_result = (test_a !== test_c);

      $display("Equality Tests:");
      $display("  %b == %b = %b (Expected: 1)", test_a, test_b, eq_result);
      $display("  %b != %b = %b (Expected: 1)", test_a, test_d, neq_result);
      $display("  %b === %b = %b (Expected: 1)", test_a, test_b, case_eq_result);
      $display("  %b !== %b = %b (Expected: 1)", test_a, test_c, case_neq_result);

      // Test relational operators
      lt_result = (test_small < test_a);
      gt_result = (test_large > test_a);

      $display();
      $display("Relational Tests:");
      $display("  %b(%d) < %b(%d) = %b (Expected: 1)",
               test_small, test_small, test_a, test_a, lt_result);
      $display("  %b(%d) > %b(%d) = %b (Expected: 1)",
               test_large, test_large, test_a, test_a, gt_result);

      // Test signed comparisons
      $display();
      $display("Signed Comparison Tests:");
      $display("  %b(%d) > %b(%d) = %b (Expected: 1)",
               test_pos, test_pos, test_neg, test_neg, (test_pos > test_neg));
      $display("  %b(%d) < %b(%d) = %b (Expected: 1)",
               test_neg, test_neg, test_pos, test_pos, (test_neg < test_pos));
```

```
    // Test unknown value comparisons
    $display();
    $display("Unknown Value Tests:");
    $display("  %b == %b = %b (Expected: x)", test_a, test_c, (test_a == test_c));
    $display("  %b === %b = %b (Expected: 0)", test_a, test_c, (test_a === test_c));
    $display("  %b === %b = %b (Expected: 1)", test_c, test_c, (test_c === test_c));

    // Comprehensive comparison matrix
    $display();
    $display("=== Comprehensive Comparison Matrix ===");
    $display("Testing all combinations of a=%b, d=%b, small=%b, large=%b",
             test_a, test_d, test_small, test_large);
    $display();

    // Create comparison table
    $display("   Operator |  a vs d  | small vs a | large vs a | a vs large");
    $display("   ---------|----------|------------|------------|------------");
    $display("       ==   |    %b     |     %b      |     %b      |     %b     ",
             (test_a == test_d), (test_small == test_a),
             (test_large == test_a), (test_a == test_large));
    $display("       !=   |    %b     |     %b      |     %b      |     %b     ",
             (test_a != test_d), (test_small != test_a),
             (test_large != test_a), (test_a != test_large));
    $display("       <    |    %b     |     %b      |     %b      |     %b     ",
             (test_a < test_d), (test_small < test_a),
             (test_large < test_a), (test_a < test_large));
    $display("       <=   |    %b     |     %b      |     %b      |     %b     ",
             (test_a <= test_d), (test_small <= test_a),
             (test_large <= test_a), (test_a <= test_large));
    $display("       >    |    %b     |     %b      |     %b      |     %b     ",
             (test_a > test_d), (test_small > test_a),
             (test_large > test_a), (test_a > test_large));
    $display("       >=   |    %b     |     %b      |     %b      |     %b     ",
             (test_a >= test_d), (test_small >= test_a),
             (test_large >= test_a), (test_a >= test_large));

    // Edge case testing
    $display();
    $display("=== Edge Case Testing ===");

    // Test boundary values
    $display("Boundary Value Tests:");
    $display("  4'b0000 == 4'd0: %b", (4'b0000 == 4'd0));
    $display("  4'b1111 == 4'd15: %b", (4'b1111 == 4'd15));
    $display("  4'b1111 > 4'b1110: %b", (4'b1111 > 4'b1110));
    $display("  4'b0000 < 4'b0001: %b", (4'b0000 < 4'b0001));

    // Test self-comparison
    $display();
    $display("Self-Comparison Tests:");
    $display("  a == a: %b (Expected: 1)", (test_a == test_a));
    $display("  a < a: %b (Expected: 0)", (test_a < test_a));
    $display("  a <= a: %b (Expected: 1)", (test_a <= test_a));
    $display("  a >= a: %b (Expected: 1)", (test_a >= test_a));
    $display("  a > a: %b (Expected: 0)", (test_a > test_a));
```

```
    // Test complex boolean expressions
    $display();
    $display("=== Complex Boolean Expression Tests ===");
    $display("(small < a) && (a < large): %b (Expected: 1)",
             ((test_small < test_a) && (test_a < test_large)));
    $display("(a == d) || (a == test_b): %b (Expected: 1)",
             ((test_a == test_d) || (test_a == test_b)));
    $display("!(a != test_b): %b (Expected: 1)", !(test_a != test_b));

    // Performance check - multiple operations
    $display();
    $display("=== Operation Timing Test ===");
    $display("Performing 1000 comparison operations...");

    for (i = 0; i < 1000; i = i + 1) begin
      temp_result = (test_a == test_b) && (test_small < test_large);
    end
    $display("Completed 1000 operations. Final result: %b", temp_result);

    $display();
    $display("=== Summary ===");
    $display("All comparison operations tested successfully!");
    $display("Key takeaways:");
    $display("- Logical equality (==) returns X for unknown values");
    $display("- Case equality (===) performs exact bit comparison");
    $display("- Signed comparisons handle negative numbers correctly");
    $display("- Width mismatches are handled by zero-extension");

    $display();
    $display("=== Test Completed Successfully ===");
    $display();

    // End simulation
    $finish;
  end

endmodule
```

Design under Test (DUT)

Design Unit Test (DUT) Testbench

```
from verilator_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_5__comparison/")
```

```
Docker Compose Output:
================================================================================
make: Entering directory '/work/obj_dir'
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated.o
```

```
/usr/local/share/verilator/include/verilated.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated_vcd_c.o
/usr/local/share/verilator/include/verilated_vcd_c.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_timing.o /usr/local/share/verilator/include/verilated_timing.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_threads.o /usr/local/share/verilator/include/verilated_threads.cpp
python3 /usr/local/share/verilator/bin/verilator_includer
-DVL_INCLUDE_OPT=include Vcomparison_testbench.cpp
Vcomparison_testbench___024root__DepSet_h8c007bc1__0.cpp
Vcomparison_testbench___024root__DepSet_h712a830d__0.cpp
Vcomparison_testbench__main.cpp Vcomparison_testbench__Trace__0.cpp
Vcomparison_testbench___024root__Slow.cpp
Vcomparison_testbench___024root__DepSet_h712a830d__0__Slow.cpp
Vcomparison_testbench__Syms.cpp Vcomparison_testbench__Trace__0__Slow.cpp
Vcomparison_testbench__TraceDecls__0__Slow.cpp > Vcomparison_testbench__ALL.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
Vcomparison_testbench__ALL.o Vcomparison_testbench__ALL.cpp
echo "" > Vcomparison_testbench__ALL.verilator_deplist.tmp
g++    verilated.o verilated_vcd_c.o verilated_timing.o verilated_threads.o
Vcomparison_testbench__ALL.a    -pthread -lpthread -latomic   -o
Vcomparison_testbench
rm Vcomparison_testbench__ALL.verilator_deplist.tmp
make: Leaving directory '/work/obj_dir'
- V e r i l a t i o n   R e p o r t: Verilator 5.036 2025-04-27 rev v5.036
- Verilator: Built from 0.050 MB sources in 3 modules, into 0.051 MB in 10 C++
files needing 0.000 MB
- Verilator: Walltime 27.517 s (elab=0.002, cvt=0.048, bld=27.288); cpu 0.043 s
on 1 threads; alloced 20.176 MB
Hello from comparison operations testbench!
Testing all comparison operators...
```

```
Starting comparison operations...
Values: a=1010(10), b=1010(10), c=1010, d=0110( 6), e=1111(15)
Signed values: pos_num=0111( 7), neg_num=1001(-7)

=== Equality Comparisons ===
a == b: 1010 == 1010 = 1 (identical values)
a != b: 1010 != 1010 = 0 (should be 0)
a == d: 1010 == 0110 = 0 (different values)
a != d: 1010 != 0110 = 1 (different values)

=== Comparisons with Unknown Values ===
a == c: 1010 == 1010 = 1 (unknown due to x/z in c)
a != c: 1010 != 1010 = 0 (unknown due to x/z in c)
a === b: 1010 === 1010 = 1 (exact bit-by-bit match)
a === c: 1010 === 1010 = 1 (exact comparison, no x/z match)
c === c: 1010 === 1010 = 1 (identical including x/z)
a !== c: 1010 !== 1010 = 0 (case inequality)

=== Relational Comparisons ===
a < e: 1010(10) < 1111(15) = 1
d < a: 0110( 6) < 1010(10) = 1
a < a: 1010(10) < 1010(10) = 0 (same values)
a <= b: 1010(10) <= 1010(10) = 1 (equal values)
d <= a: 0110( 6) <= 1010(10) = 1
e > a: 1111(15) > 1010(10) = 1
a > d: 1010(10) > 0110( 6) = 1
a >= b: 1010(10) >= 1010(10) = 1 (equal values)
a >= d: 1010(10) >= 0110( 6) = 1

=== Signed Number Comparisons ===
pos_num > neg_num: 0111( 7) > 1001(-7) = 1
neg_num < pos_num: 1001(-7) < 0111( 7) = 1
neg_num < 0: 1001(-7) < 0 = 1

=== Width Mismatched Comparisons ===
a == wide_val: 1010(10) == 00001010( 10) = 1 (different widths, same value)
a === wide_val[3:0]: 1010 === 1010 = 1 (same width after slicing)

=== Edge Cases ===
4'b0000 == 4'd0: 1
4'b1111 == 4'd15: 1
c > d: 1010 > 0110( 6) = 1 (unknown due to x/z)
c < d: 1010 < 0110( 6) = 0 (unknown due to x/z)

=== Chained Comparisons ===
(d < a) && (a < e): (0110 < 1010) && (1010 < 1111) = 1 && 1 = 1
(a >= d) || (a <= b): (1010 >= 0110) || (1010 <= 1010) = 1 || 1 = 1

All comparison operations completed successfully!

=== Testbench Verification ===
Final result from design: 1

=== Manual Verification Tests ===
Equality Tests:
```

```
  1010 == 1010 = 1 (Expected: 1)
  1010 != 0110 = 1 (Expected: 1)
  1010 === 1010 = 1 (Expected: 1)
  1010 !== 1010 = 0 (Expected: 1)

Relational Tests:
  0001( 1) < 1010(10) = 1 (Expected: 1)
  1111(15) > 1010(10) = 1 (Expected: 1)

Signed Comparison Tests:
  0111( 7) > 1001(-7) = 1 (Expected: 1)
  1001(-7) < 0111( 7) = 1 (Expected: 1)

Unknown Value Tests:
  1010 == 1010 = 1 (Expected: x)
  1010 === 1010 = 1 (Expected: 0)
  1010 === 1010 = 1 (Expected: 1)

=== Comprehensive Comparison Matrix ===
Testing all combinations of a=1010, d=0110, small=0001, large=1111

   Operator |  a vs d  | small vs a | large vs a | a vs large
   ---------|----------|------------|------------|------------
       ==   |    0     |     0      |     0      |     0
       !=   |    1     |     1      |     1      |     1
       <    |    0     |     1      |     0      |     1
       <=   |    0     |     1      |     0      |     1
       >    |    1     |     0      |     1      |     0
       >=   |    1     |     0      |     1      |     0

=== Edge Case Testing ===
Boundary Value Tests:
  4'b0000 == 4'd0: 1
  4'b1111 == 4'd15: 1
  4'b1111 > 4'b1110: 1
  4'b0000 < 4'b0001: 1

Self-Comparison Tests:
  a == a: 1 (Expected: 1)
  a < a: 0 (Expected: 0)
  a <= a: 1 (Expected: 1)
  a >= a: 1 (Expected: 1)
  a > a: 0 (Expected: 0)

=== Complex Boolean Expression Tests ===
(small < a) && (a < large): 1 (Expected: 1)
(a == d) || (a == test_b): 1 (Expected: 1)
!(a != test_b): 1 (Expected: 1)

=== Operation Timing Test ===
Performing 1000 comparison operations...
Completed 1000 operations. Final result: 1

=== Summary ===
All comparison operations tested successfully!
Key takeaways:
```

```
- Logical equality (==) returns X for unknown values
- Case equality (===) performs exact bit comparison
- Signed comparisons handle negative numbers correctly
- Width mismatches are handled by zero-extension

=== Test Completed Successfully ===

- comparison_testbench.sv:164: Verilog $finish
- S i m u l a t i o n   R e p o r t: Verilator 5.036 2025-04-27
- Verilator: $finish at 31ps; walltime 0.006 s; speed 16.674 ns/s
- Verilator: cpu 0.002 s on 1 threads; alloced 25 MB
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_5__comparison/obj_dir directory...
Chapter_3_examples/example_5__comparison/obj_dir removed successfully.
```

```
0
```

```
from gtkwave_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_5__comparison/")
```

```
Docker Compose Output:
================================================================================
 Container notebooks-verilator-1  Recreate
 Container notebooks-verilator-1  Recreated
 Container notebooks-verilator-1  Starting
 Container notebooks-verilator-1  Started

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI

[0] start time.
[31] end time.
WM Destroy
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_5__comparison/obj_dir directory...
Chapter_3_examples/example_5__comparison/obj_dir removed successfully.
```

```
0
```

Conditional Operator

The conditional operator provides a compact way to select between two values based on a condition.

Syntax

```
condition ? true_expression : false_expression
```

Example

Design under Test (DUT)

```
// conditional.sv
module conditional;
    logic [7:0] a = 8'd10;
    logic [7:0] b = 8'd20;
    logic [7:0] max_val;
    logic [7:0] abs_diff;
    logic [1:0] sel = 2'b10;
    logic [7:0] mux_out;

    initial begin
        $display();                                          // Display empty line
        $display("Hello from conditional design!");    // Display message

        // Find maximum
        max_val = (a > b) ? a : b;  // max_val = 20

        // Absolute difference
        abs_diff = (a > b) ? (a - b) : (b - a);  // abs_diff = 10

        // Nested conditional
        mux_out = (sel == 2'b00) ? 8'd1 :
                  (sel == 2'b01) ? 8'd2 :
                  (sel == 2'b10) ? 8'd4 : 8'd8;  // mux_out = 4

        $display("max(%d, %d) = %d", a, b, max_val);
        $display("abs_diff = %d", abs_diff);
        $display("mux_out = %d", mux_out);
    end
endmodule
```

Design Unit Test (DUT) Testbench

```
// conditional_testbench.sv
module conditional_testbench;  // Testbench module
  conditional CONDITIONAL_INSTANCE();  // Instantiate design under test

  initial begin
    // Dump waves
    $dumpfile("conditional_testbench.vcd");     // Specify the VCD file
    $dumpvars(0, conditional_testbench);        // Dump all variables in the test module
    #1;                                          // Wait for a time unit
    $display("Hello from conditional testbench!"); // Display message
    $display();                                  // Display empty line
  end

endmodule
```

```
from verilator_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_6__conditional/")
```

```
Docker Compose Output:
================================================================================
make: Entering directory '/work/obj_dir'
```

```
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated.o
/usr/local/share/verilator/include/verilated.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated_vcd_c.o
/usr/local/share/verilator/include/verilated_vcd_c.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_timing.o /usr/local/share/verilator/include/verilated_timing.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_threads.o /usr/local/share/verilator/include/verilated_threads.cpp
python3 /usr/local/share/verilator/bin/verilator_includer
-DVL_INCLUDE_OPT=include Vconditional_testbench.cpp
Vconditional_testbench___024root__DepSet_hbbfbec53__0.cpp
Vconditional_testbench___024root__DepSet_hbb4057d5__0.cpp
Vconditional_testbench__main.cpp Vconditional_testbench__Trace__0.cpp
Vconditional_testbench___024root__Slow.cpp
Vconditional_testbench___024root__DepSet_hbb4057d5__0__Slow.cpp
Vconditional_testbench__Syms.cpp Vconditional_testbench__Trace__0__Slow.cpp
Vconditional_testbench__TraceDecls__0__Slow.cpp >
Vconditional_testbench__ALL.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
Vconditional_testbench__ALL.o Vconditional_testbench__ALL.cpp
echo "" > Vconditional_testbench__ALL.verilator_deplist.tmp
g++    verilated.o verilated_vcd_c.o verilated_timing.o verilated_threads.o
Vconditional_testbench__ALL.a    -pthread -lpthread -latomic   -o
Vconditional_testbench
rm Vconditional_testbench__ALL.verilator_deplist.tmp
make: Leaving directory '/work/obj_dir'
```

```
- V e r i l a t i o n   R e p o r t: Verilator 5.036 2025-04-27 rev v5.036
- Verilator: Built from 0.036 MB sources in 3 modules, into 0.037 MB in 10 C++
files needing 0.000 MB
- Verilator: Walltime 26.739 s (elab=0.000, cvt=0.030, bld=26.590); cpu 0.021 s
on 1 threads; alloced 20.176 MB

Hello from conditional design!
max( 10,  20) =  20
abs_diff =  10
mux_out =   4
Hello from conditional testbench!

- S i m u l a t i o n   R e p o r t: Verilator 5.036 2025-04-27
- Verilator: end at 1ps; walltime 0.003 s; speed 669.478 ps/s
- Verilator: cpu 0.001 s on 1 threads; alloced 25 MB
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_6__conditional/obj_dir directory...
Chapter_3_examples/example_6__conditional/obj_dir removed successfully.
```

0

```
from gtkwave_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_6__conditional/")
```

```
Docker Compose Output:
================================================================================
 Container notebooks-verilator-1  Recreate
 Container notebooks-verilator-1  Recreated
 Container notebooks-verilator-1  Starting
 Container notebooks-verilator-1  Started

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI

[0] start time.
[1] end time.
WM Destroy
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_6__conditional/obj_dir directory...
Chapter_3_examples/example_6__conditional/obj_dir removed successfully.
```

0

Operator Precedence

Understanding operator precedence is crucial for writing correct expressions. Operators are listed from highest to lowest precedence:

Precedence	Operators	Description
1 (Highest)	`() [] :: .`	Parentheses, brackets, scope, member selection
2	`+ - ! ~ & ~& | ~| ^ ~^ ^~`	Unary operators
3	`**`	Exponentiation
4	`* / %`	Multiplication, division, modulus

Precedence	Operators	Description
5	+ -	Addition, subtraction
6	<< >> <<< >>>	Shift operators
7	< <= > >=	Relational operators
8	== != === !==	Equality operators
9	&	Bitwise AND
10	^ ~^ ^~	Bitwise XOR, XNOR
11	|	Bitwise OR
12	&&	Logical AND
13	||	Logical OR
14 (Lowest)	?:	Conditional operator

Examples

```
module precedence_example;
    logic [7:0] a = 8'd2;
    logic [7:0] b = 8'd3;
    logic [7:0] c = 8'd4;
    logic [7:0] result;

    initial begin
        // Without parentheses - follows precedence
        result = a + b * c;     // 2 + (3 * 4) = 14

        // With parentheses - overrides precedence
        result = (a + b) * c;   // (2 + 3) * 4 = 20

        // Complex expression
        result = a < b && b < c ? a + b : b * c;
        // Evaluated as: ((a < b) && (b < c)) ? (a + b) : (b * c)
        // Result: 5 (since 2 < 3 && 3 < 4 is true)

        $display("a + b * c = %d", a + b * c);
        $display("(a + b) * c = %d", (a + b) * c);
        $display("Complex expression = %d", result);
    end
endmodule
```

Design under Test (DUT)

```
// precedence.sv
module precedence;
    logic [7:0] a = 8'd2;
    logic [7:0] b = 8'd3;
    logic [7:0] c = 8'd4;
    logic [7:0] result;

    initial begin
        $display();                                     // Display empty line
        $display("Hello from precedence design!");      // Display message

        // Without parentheses - follows precedence
        result = a + b * c;     // 2 + (3 * 4) = 14
```

```
        $display("a + b * c = %d", result);

        // With parentheses - overrides precedence
        result = (a + b) * c;   // (2 + 3) * 4 = 20
        $display("(a + b) * c = %d", result);

        // Complex expression
        result = a < b && b < c ? a + b : b * c;
        // Evaluated as: ((a < b) && (b < c)) ? (a + b) : (b * c)
        // Result: 5 (since 2 < 3 && 3 < 4 is true)
        $display("Complex expression = %d", result);

        // Additional precedence examples
        result = a | b & c;     // a | (b & c) = 2 | (3 & 4) = 2 | 0 = 2
        $display("a | b & c = %d", result);

        result = (a | b) & c;   // (2 | 3) & 4 = 3 & 4 = 0
        $display("(a | b) & c = %d", result);

        // Shift and arithmetic precedence
        result = a + b << 1;    // (a + b) << 1 = (2 + 3) << 1 = 5 << 1 = 10
        $display("a + b << 1 = %d", result);

        result = a + (b << 1);  // a + (b << 1) = 2 + (3 << 1) = 2 + 6 = 8
        $display("a + (b << 1) = %d", result);
    end
endmodule
```

Design Unit Test (DUT) Testbench

```
// precedence_testbench.sv
module precedence_testbench;  // Testbench module
  precedence PRECEDENCE_INSTANCE();  // Instantiate design under test

  initial begin
    // Dump waves
    $dumpfile("precedence_testbench.vcd");     // Specify the VCD file
    $dumpvars(0, precedence_testbench);        // Dump all variables in the test module
    #1;                                         // Wait for a time unit
    $display("Hello from precedence testbench!"); // Display message
    $display();                                 // Display empty line
  end

endmodule
```

```
from verilator_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_7__precedence/")
```

```
Docker Compose Output:
================================================================================
make: Entering directory '/work/obj_dir'
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
```

```
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated.o
/usr/local/share/verilator/include/verilated.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o verilated_vcd_c.o
/usr/local/share/verilator/include/verilated_vcd_c.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_timing.o /usr/local/share/verilator/include/verilated_timing.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
verilated_threads.o /usr/local/share/verilator/include/verilated_threads.cpp
python3 /usr/local/share/verilator/bin/verilator_includer
-DVL_INCLUDE_OPT=include Vprecedence_testbench.cpp
Vprecedence_testbench___024root__DepSet_hc5215b27__0.cpp
Vprecedence_testbench___024root__DepSet_h21ad6a2a__0.cpp
Vprecedence_testbench__main.cpp Vprecedence_testbench__Trace__0.cpp
Vprecedence_testbench___024root__Slow.cpp
Vprecedence_testbench___024root__DepSet_h21ad6a2a__0__Slow.cpp
Vprecedence_testbench__Syms.cpp Vprecedence_testbench__Trace__0__Slow.cpp
Vprecedence_testbench__TraceDecls__0__Slow.cpp > Vprecedence_testbench__ALL.cpp
ccache g++ -Os  -I.  -MMD -I/usr/local/share/verilator/include
-I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0
-DVM_TIMING=1 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -DVM_TRACE_SAIF=0
-faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-
compare -Wno-subobject-linkage -Wno-tautological-compare -Wno-uninitialized
-Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter
-Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutines -c -o
Vprecedence_testbench__ALL.o Vprecedence_testbench__ALL.cpp
echo "" > Vprecedence_testbench__ALL.verilator_deplist.tmp
g++    verilated.o verilated_vcd_c.o verilated_timing.o verilated_threads.o
Vprecedence_testbench__ALL.a    -pthread -lpthread -latomic   -o
Vprecedence_testbench
rm Vprecedence_testbench__ALL.verilator_deplist.tmp
make: Leaving directory '/work/obj_dir'
- V e r i l a t i o n   R e p o r t: Verilator 5.036 2025-04-27 rev v5.036
- Verilator: Built from 0.036 MB sources in 3 modules, into 0.035 MB in 10 C++
files needing 0.000 MB
```

```
- Verilator: Walltime 26.655 s (elab=0.000, cvt=0.056, bld=26.421); cpu 0.025 s
on 1 threads; alloced 20.180 MB

Hello from precedence design!
a + b * c =  14
(a + b) * c =  20
Complex expression =   5
a | b & c =   2
(a | b) & c =   0
a + b << 1 =  10
a + (b << 1) =   8
Hello from precedence testbench!

- S i m u l a t i o n   R e p o r t: Verilator 5.036 2025-04-27
- Verilator: end at 1ps; walltime 0.003 s; speed 745.490 ps/s
- Verilator: cpu 0.001 s on 1 threads; alloced 25 MB
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_7__precedence/obj_dir directory...
Chapter_3_examples/example_7__precedence/obj_dir removed successfully.
```

```
0
```

```
from gtkwave_runner import run_docker_compose

run_docker_compose("Chapter_3_examples/example_7__precedence/")
```

```
Docker Compose Output:
================================================================================
 Container notebooks-verilator-1  Recreate
 Container notebooks-verilator-1  Recreated
 Container notebooks-verilator-1  Starting
 Container notebooks-verilator-1  Started

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI

[0] start time.
[1] end time.
WM Destroy
================================================================================
Process finished with return code: 0
Removing Chapter_3_examples/example_7__precedence/obj_dir directory...
Chapter_3_examples/example_7__precedence/obj_dir removed successfully.
```

```
0
```

Best Practices

1. **Use parentheses for clarity**: Even when not required by precedence, parentheses make expressions more readable.
2. **Be careful with signed/unsigned mixing**: SystemVerilog has specific rules for mixed arithmetic.
3. **Use case equality for X/Z values**: Use === and !== when you need to compare X and Z values exactly.
4. **Consider bit widths**: Ensure your result variables are wide enough to hold the operation results.
5. **Use reduction operators efficiently**: They're powerful for checking conditions across all bits.

Summary

SystemVerilog operators provide powerful tools for data manipulation and decision making. Key points to remember:

- Arithmetic operators follow standard mathematical rules with special handling for division and modulus
- Logical operators work on expressions, bitwise operators work on individual bits
- Reduction operators collapse multi-bit values to single bits
- Shift operators provide both logical and arithmetic variants
- Comparison operators include both standard and case-sensitive versions
- The conditional operator enables compact selection logic
- Operator precedence follows intuitive mathematical conventions but should be clarified with parentheses when in doubt

Understanding these operators and their interactions is fundamental to writing effective SystemVerilog code for both design and verification.