SystemVerilog Cheatsheet

A concise reference for SystemVerilog syntax and constructs, covering data types, operators, procedural statements, and verification features.



Data Types & Declarations

Basic Data Types

log	Two-state type, can be 0 or 1. Preferred for synthesizable designs.
re g	Historically used for sequential logic outputs; now largely replaced by logic.
bi	Two-state, unsigned data type.
in	32-bit signed integer.
rea	64-bit floating-point number.
tim e	64-bit unsigned integer representing simulation time.

Arrays

Fixed-size array	(logic [7:0] data [0:15]; // 16 elements, each 8 bits wide.
Dynamic array	<pre>int dyn_array[]; dyn_array = new[array_size];</pre>
Associative array	<pre>bit [63:0] assoc_array [string]; // Index with string.</pre>

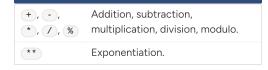
User-Defined Types

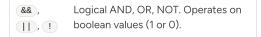
typed	<pre>typedef logic [3:0] nibble_t; nibble_t my_nibble;</pre>
struc	<pre>typedef struct { logic valid; logic [7:0] data; } packet_t; packet_t my_packet;</pre>
enum	<pre>typedef enum {IDLE, READ, WRITE} state_t; state_t current_state;</pre>

Operators & Expressions

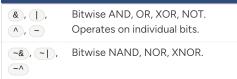
Arithmetic Operators

Logical Operators





Bitwise Operators



Reduction Operators

& ,	Reduction AND, OR, XOR. Operates on
1,	all bits of a vector to produce a single-
Λ	bit result.

Shift Operators

<< , >> ,	Logical left shift, logical right shift,
<<< ,	arithmetic left shift, arithmetic right
>>>	shift.

Comparison Operators

==, (1=),	Equality, inequality, case equality, case inequality. Case equality considers X and Z.
>, <,	Greater than, less than, greater than or equal to, less than or equal to.

Procedural Statements

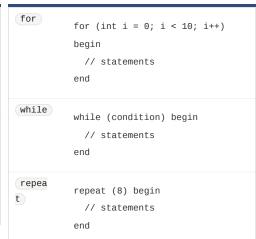
Sequential Blocks

alway s_com b	Combinational logic block. Re- evaluates whenever any of its inputs change.
alway	Sequential logic block. Used for describing flip-flops and registers.
alway s_latc	Latch inferrence. Avoid using latches in synchronous design.

Conditional Statements

if-else	<pre>if (condition) begin // statements end else begin // statements end</pre>
(case)	<pre>case (expression) value1: statement; value2: statement; default: statement; endcase</pre>

Loop Statements



Task and Function

tas k	Can consume simulation time. Can have input, output, and inout arguments.
func tion	Cannot consume simulation time. Returns a single value. Can only have input arguments.

Page 1 of 2 https://cheatsheetshero.com

Verification Features

Assertions

assert propert y	Checks if a property holds true. Can be used for functional coverage.
cover propert y	Collects coverage information based on property evaluation.

Constrained Random Verification

rand	Specifies that a variable should be randomized.
constr aint	Defines constraints that the random values must satisfy.

Coverage

Functional	Measure of how well the design's
Coverage	functionality has been exercised
	during verification. Check
	covergroup and coverpoint

Page 2 of 2 https://cheatsheetshero.com