Appendix E

Verilog Quick Reference Guide

Category	Definition	Example
Identifer Names	Can contain any letter, digit, underscore _, or $ Can not begin with a digit or be a keyword Case sensitive	q0 Prime_number lteflg
Signal Values	0 = logic value 0 1 = logic value 1 z or Z = high impedance x or X = unknown value	
Numbers	d = decimal b = binary h = hexadecimal o = octal	35 (default decimal) 4'b1001 8'a5 = 8'b10100101
Parameters	Associates an identifer name with a value that can be overridden with the **defparam** statement	#(**parameter** N = 8)
Local parameters	Associates an identifer name with a constant that cannot be directly overridden	**localparam** [1:0] s0 = 2'b00, s1 = 2'b01, s2 = 2'b10;
Nets and Variables Types	**wire** (used to connect one logic element to another) **reg** (variables assigned values in **always** block) **integer** (useful for loop control variables)	**wire** [3:0] d; **wire** led; **reg** [7:0] q; **integer** k;

Module	**module** module_name
	[#(parameter_port_list)]
	(port_dir_type_name,{ port_dir_type_name }
	);
	[**wire** declarations]
	[**reg** declarations]
	[**assign** assignments]
	[**always** blocks]
	
	endmodule
		module register
		#(**parameter** N = 8)
		(**input wire** load ,
		input wire clk ,
		input wire clr ,
		input wire [N-1:0] d ,
		output reg [N-1:0] q
		);
		
		always @(**posedge** clk **or posedge** clr)
		if(clr == 1)
		q <= 0;
		else if(load)
		q <= d;
		endmodule

Logic operators	~ (NOT) & (AND) | (OR) ~(&) (NAND) ~(|) (NOR) ^ (XOR) ~^ (XNOR	**assign** z = ~y; **assign** c = a & b; **assign** z = x | y; **assign** w = ~(u & v); **assign** r = ~(s | t); **assign** z = x ^ y; **assign** d = a ~^ b;
Reduction operators	& (AND) | (OR) ~& (NAND) ~| (NOR) ^ (XOR) ~^ (XNOR	**assign** c = &a; **assign** z = |y; **assign** w = ~&v; **assign** r = ~|t; **assign** z = ^y; **assign** d = ~^b;
Arithmetic operators	+ (addition) - (subtraction) * (multiplication) / (division) % (mod)	count <= count + 1; q <= q – 1;

Verilog Quick Reference Guide (cont.)

Relational operators	==, !=, >, <, >=, <=, ===, !==	`assign lteflg = (a <= b);`
		`assign eq = (a == b);`
		`if(clr == 1)`

Shift operators	<< (shift left) >> (shift right)	`c = a << 3;` `c = a >> 4;`

always block	**always @**(<sensitivity list>)
	always @(*)
		`always @(*)`
		`begin`
		`  s = a ^ b;`
		`  c = a & b;`
		`end`

if statement	**if**(expression1)
	begin
	  statement;
	end
	else if (expression2)
	begin
	  statement;
	end
	else
	begin
	  statement;
	end
		`if(s == 0)`
		`  y = a;`
		`else`
		`  y = b;`

case statement	**case**(expression)
	  alternative1: **begin**
	  statement;
	  end
	  alternative2: **begin**
	  statement;
	  end
	  [**default**: **begin**
	  statement;
	  end
	endcase
		`case(s)`
		`  0: y = a;`
		`  1: y = b;`
		`  2: y = c;`
		`  3: y = d;`
		`  default: y = a;`
		`endcase`

for loop	**for**(initial_index; terminal_index; increment) **begin** statement; **end**	`for(i=2; i<=4; i=i+1)` `  z = z & x[i];`
Assignment operator	= (blocking) <= (non-blocking)	`z = z & x[i];` `count <= count + 1;`

Module instantiation	Module_name instance_name(.port_name(expr)
	  {,.port_name([expr])});
		`hex7seg d7R(.d(y),`
		`  .a_to_g(a_to_g)`
		`);`

Parameter override	**defparam** instance_name.parameter_name = val;	`defparam  Reg.N = 16;`