CS147 - Lab 04

Data Flow Modeling II

Kaushik Patra
(kaushik.patra@sjsu.edu)

Continuous assignment

- Most basic statement in data flow modeling.
  - It assigns a left hand side of an equation to the evaluated value from the right hand side. e.g. 'assign a = b & c';
  - It is always active. The value of 'a' will be evaluated as soon as value b or c changes.
  - This is equivalent to have a always block 'always @(b or c) begin a = b & c; end.
    - This is a behavioral modeling.

Continuous assignment

- LHS must be a scalar or vector net or concatenation of scalar or vector net. It can not be a scalar or vector register.

- RHS can be registers or nets or function calls. Register or nets can be scalar or vector.

3

Continuous assignment

```
// net to net assignment
assign out = i1 & i2;

// register vectors to net vector assignments
assign addr[15:0] = addr1[15:0] ^ addr2[15:0]; // XOR
operation

// Concatenation example
assign { cout, sum[3:0] } = a[3:0] + b[3:0] + c_in;
```

4

Implicit Continuous assignment

```
// Regular continuous assignments
wire out;
assign out = i1 & i2;

// Same achieved by implicit continuous assignments
wire out = i1 & i2;
```

5

Arithmetic operators

Type	Symbol	Notes
Binary	+	Addition. e.g. a = b + c;
Binary	-	Subtraction. e.g. a = b - c;
Binary	*	Multiplication. e.g. a = b *& c;
Binary	/	Integer Division. e.g. a = b / c;
Binary	%	Modulus. e.g. a = b % c;

Logical operators

Type	Symbol	Notes
Unary	!	Logical negation resulting one bit value. e.g. !(a)
Binary	&&	Logical AND resulting one bit value. e.g. (a && b)
Binary	||	Logical OR resulting one bit value. e.g. a = b *& c;

6

Relational operators

Type	Symbol	Notes
Binary	>	Greater than operation resulting one bit value. e.g. (a > b)
Binary	<	Less than operation resulting one bit value. e.g. (a < b)
Binary	>=	Greater than equal operation resulting one bit value. e.g. (a >= b)
Binary	<=	Less than equal operation resulting one bit value. e.g. (a <= b)

Equality operators

Type	Symbol	Notes
Binary	==	Equality operation where bit value x and z will result in 0.
Binary	!=	Inverse of ==
Binary	===	Case equality where it tries to match x and z.
Binary	!==	Inverse of ===

7

Bitwise operators

Type	Symbol	Notes
Unary	~	Bitwise negation
Binary	&	Bitwise and
Binary	|	Bitwise or
Binary	^	Bitwise xor
	^~ or ~^	Bitwise xnor

Reduction operators

Type	Symbol	Notes
Unary	&	AND between all bits of the variable (e.g. &x = 0 if x = 4'b0111).
Unary	~&	NAND between all bits of the variable (e.g. &x = 1 if x = 4'b0111).
Unary	|	OR between all bits of the variable (e.g. &x = 1 if x = 4'b0111).
Unary	~|	NOR between all bits of the variable (e.g. &x = 0 if x = 4'b0111).
Unary	^	XOR between all bits of the variable (e.g. &x = 1 if x = 4'b0111).
Unary	^~ or ~^	XNOR between all bits of the variable (e.g. &x = 0 if x = 4'b0111).

8

Miscellaneous operators

Type	Symbol	Notes
Binary	<<	Left shift; e.g. a << b;
Binary	>>	Right shift; e.g. a >> b;
Any number	{ }	Concatenation operation; e.g. wire [3:0] out = { cout, res[2:0] };
Any number	{ { } }	Replication operation to replicate a value over multiple bits. e.g. reg a = 1'b1; reg b = 1'b0; reg c = 1'b'1; wire [6:0] y = { 4{a},2{b},c}; The value of y will be 7'b1111001
Three Var	?:	Conditional assignments; e.g. assign a = (b < c)? b: c; This code will always assign a with the least value between b and c; Equivalent code is 'if (b<c) a = b; else a = c;

9

CS147 - Lab 04

Data Flow Modeling II

Kaushik Patra
(kaushik.patra@sjsu.edu)

10