SYSTEM TASKS COMPILER DIRECTIVES

PATH DELAYS

- The delay between source pin (input or inout) and destination pin (ouput or inout) of module is called module path delay.
- Verilog has the provision to specify and check delays associated with total paths – from any input to any output of a module.
- Such paths and delays are at the chip or system level.
- They are referred to as “module path delays”.
- Constructs available make room for specifying their paths and assigning delay values to them – separately or together

Specify Blocks

- Module paths are specified and values assigned to their delays through specify blocks.
- They are used to specify rise time, fall time, path delays pulse widths, and the like.
- The block starts with the keyword “specify” and ends with the keyword “endspecify”. Specify blocks can appear anywhere within a module
- A “specify” block can have the form shown in below

```
specify
specparam rise_time = 5, fall_time = 6;
(a =>b) = (rise_time, fall_time);
(c => d) = (6, 7);
endspecify
```

Module Paths

- Module Path delays are assigned in Verilog within the keywords specify and endspecify.
- The statements within these keywords constitute a specify block.
- Module paths can be specified in different ways inside a specify block

  1. Parallel connection
  2. Full connection

1. Parallel connection

- Every path delay statement has a source field and a destination field.
- A parallel connection is specified by the symbol => and is used as

Usage: (=>) = ;

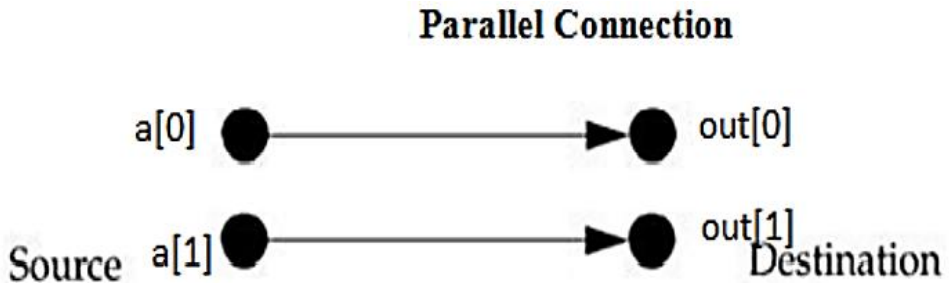
- In a parallel connection, each bit in source field connects to its corresponding bit in the destination field.
- If the source and the destination fields are vectors, they must have the same number of bits; otherwise, there is a mismatch. Thus, a parallel connection specifies delays from each bit in source to each bit in destination

Example: Parallel Connection

(a => out) = 9;

- Bit-to-bit connection. Both a and out are single-bit vector connection. a is source field, out is destination field.

```
// for three bit-to-bit connection statements.
(a[0] => out[0]) = 9;
(a[1] => out[1]) = 9;
(a[2] => out[2]) = 9;
```



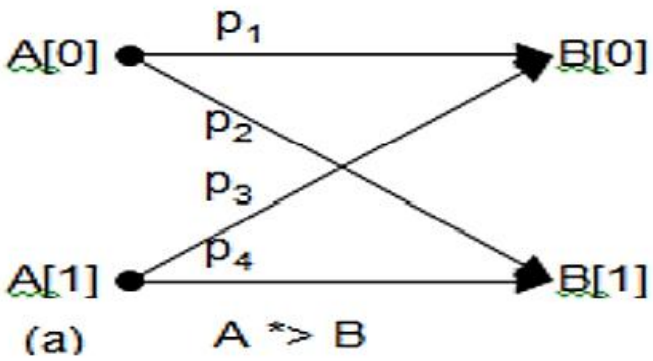

```
//illegal connection. a[4:0] is a 5-bit vector, out[3:0] is 4-bit.
 //Mismatch between bit width of source and destination fields
(a => out) = 9;                    //bit width does not match.
```

2. Full connection

- A full connection is specified by the symbol *> and is used as

  Usage: (*>) = ;

- In a full connection, each bit in the source field connects to every bit in the destination field.
- If the source and the destination are vectors, then they need not have the same number of bits.
- A full connection describes the delay between each bit of the source and every bit in the destination.
- Example: Figure below illustrates a case of all possible paths from a 2-bit vector A to another 2-bit vector B; the specification implies 4 path



//Full Connection

```
module M (out, a, b, c, d);
output out;
input a, b, c, d;
wire e, f;
specify
(a,b *> out) = 9;
(c,d *> out) = 11;
endspecify
and a1(e, a, b);
and a2(f, c, d);
and a3(out, e, f);
endmodule
```

// Parallel connection

```
module M (out, a, b, c, d);
output out;
input a, b, c, d;
wire e, f;
 //Specify block with path delay statements
specify
(a => out) = 9;
(b => out) = 9;
(c => out) = 11;
(d => out) = 11;
endspecify

//gate instantiations
and a1(e, a, b);
and a2(f, c, d);
and a3(out, e, f);
endmodule
```

MODULE PARAMETERS

- Module parameters are associated with size of bus, register, memory, ALU, and so on.
- They can be specified within the concerned module but their value can be altered during instantiation.
- The alterations can be brought about through assignments made with defparam. Such defparam assignments can appear anywhere in a module.

Example: The parameter ***msb*** specifies the ALU size — consistently in the input and the output vectors of the ALU. The size assignment has been made separately through the assignment statement parameter msb = 3;

The ALU module with its size declared as a parameter.

```
module alu (d, co, a, b, f,cci);
 parameter msb=3;
output [msb:0] d; output co;
wire[msb:0]d;
input cci;
input [msb : 0 ] a, b;
input [1 : 0] f;
specify
(a,b=>d)=(1,2);
(a,b,cci*>co)=1;
endspecify
assign {co,d}= (f==2'b00)?(a+b+cci):((f==2'b01)?(a-b):((f==2'b10)?{1'bz,a^b}:{1'bz,~a}));
endmodule
```

SYSTEM TASKS AND FUNCTIONS

- Verilog has a number of System Tasks and Functions defined in the LRM (language reference manual).
- They are for taking output from simulation, control simulation, debugging design modules, testing modules for specifications, etc.
- ***A "$" sign preceding a word or a word group signifies a system task or a system function.***

- Verilog HDL provides built-in system tasks and system functions, that is, tasks and functions that are predefined in the language. These are grouped as follows

i. Display tasks
ii. File I/O tasks
iii. Timescale tasks
iv. Simulation control tasks
v. Timing check tasks
vi. PLA modeling tasks
vii. Stochastic modeling tasks
viii. Conversion functions for reals
ix. Probabilistic distribution functions

THANK YOU