endtask

12. Commonly used Compiler Directives

```
`define word_size 32
`include ../header.v
`timescale 100ns/1ns              // ref_time_unit / precision
`ifdef , `else , `endif
e.g.
module and_op (a,b,c);
  output a;
  input b,c;
  `ifdef behavioral
    wire a = b & c;
  `else
    and a1(a,b,c);
  `endif
endmodule
```

13. Observing Outputs

```
$display ("Value of variable is %d", var);
integer flag;
initial flag = $fopen("out_file");
always @(....);  // dump data in text file
    $fdisplay (flag, "%h", data[7:0]);
    ........
    $fclose ("out_file");
end
$monitor ($time, "a = %b and b = %b", clock, reset);
$fmonitor(flag, "value = %h", add[15:0]);
$monitoron;
$monitoroff;
```

14. Simulation Control

```
initial begin
    $dumpfile ("my.dump");      // dump in this file
    $dumpvars;                  // dump all signals
    $dumpvars (1,top);  // dump variables in module instance top.
    $dumpvars (2,top.m1);       // dump 2 levels below top.m1
    #1000 dumpoff ;             // stop dump
    #500 dumpon;                // start / restart dump
    $stop;                      // stop for interaction
    #1000 $finish;              // come out of simulation
end
```

15. Language Constructs not supported by most Synthesis tools

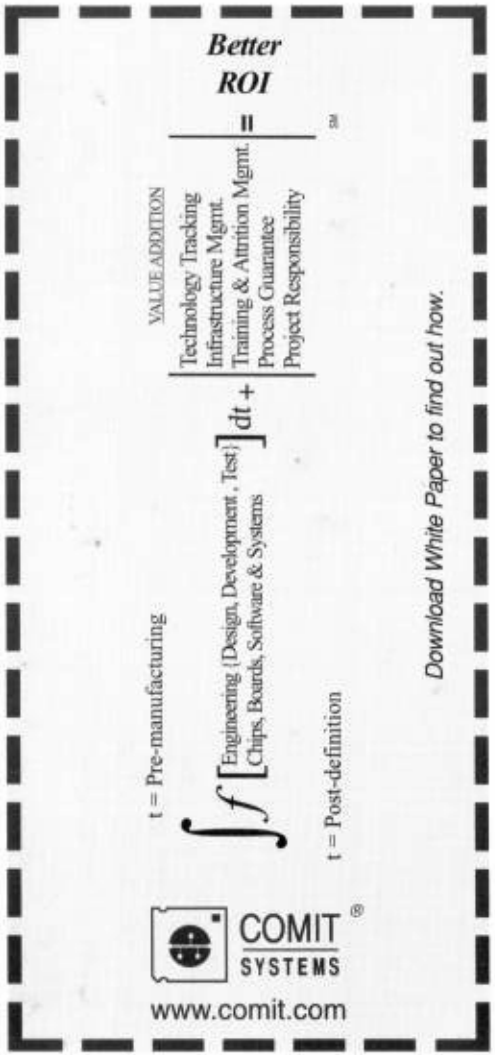
Declarations and Definitions
- **time** declaration
- **event** declaration
- **triand, trior, tri1, tri0,** and **trireg** net types
- Ranges and arrays for integers
- **primitive** definition

Statements
- **initial** statement
- delay control
- event control
- **wait** statement
- **repeat** statement
- **fork** statement
- **deassign** statement
- **force** statement
- **release** statement
- **defparam** statement

Operators
- Division and modulus operators for variables
- Case equality and inequality operators (=== and !==)

Gate-Level Constructs
- **pullup, pulldown,**
- **tranif0, tranif1, rtran, rtranif0, rtranif1**

Miscellaneous Constructs
- Compiler directives like **`ifdef, `endif** and **`else**
- Hierarchical names within a module



COMIT® SYSTEMS

Verilog® Quick Reference Card

1. Module

```
module module_name (list of ports);
  input / output / inout declarations
  net / reg declarations
  integer declarations
  parameter declarations

  gate/switch instances
  hierarchical instances
  parallel statements
endmodule
```

2. Parallel Statements

Following statements start executing simultaneously inside module

```
initial begin
  {sequential statements}
end
always begin
  {sequential statements}
end
assign wire_name = [expression];
```

3. Basic Data Types

a. Nets
e.g. **wire, wand, tri, wor**
- Continuously driven
- Gets new value when driver changes
- LHS of continuous assignment

```
tri [15:0] data;
  // unconditional
assign data[15:0] = data_in ;
  // conditional
assign data[15:0] = enable ? data_in : 16'bz;
```

b. Registers
e.g. **reg**
- Represents storage
- Always stores last assigned value
- LHS of an assignment in procedural block.

```
reg signal;
@ (posedge clock) signal = 1'b1; // positive edge
@ (reset) signal = 1'b0;         // event (both edges)
```

4. Sequential Statements

Given below are simple examples instead of BNF type of definitions.

```
• if (reset == 0) begin
    data = 8'b00;
  end
```