SystemVerilog Notes

Gautam Batra

January 2, 2021

1 About SystemVerilog/Features

- Loosely typed
- Variable types:
  - Register
  - Integer
  - Time
  - Real
- Net type determines how it's resolved if it has multiple drivers
- Several transistor and gate level primitives
- Construct available for user defined primitives
- Programmable Logic Array (PLA) modelling mechanism
- Propagation delay for nets and primitives
- Drive strength for continuous assignments and primitives
- Path delays, pulse filtering and timing checks for modules
- Comprehensive, i.e., no user extensions
- Description of HW as set of modules
- Can instantiate one module in another (creates a copy)
- 2 ways of specifying module:
  1. Behavioral representation: program-like (starting point)
  2. Structural representation: internal logic structure
- Modules interconnected by nets (like wires)
- What we can do with SystemVerilog code
  - Simulate system and verify the operation
    - Use *test benches* to provide input and verify output
  - Use synthesis tool to map to HW
    - Converts it into netlist of low-level primitives
    - HW could be ASIC or FPGA
    - Don't need testbenches, can directly test using real signals

    * ASIC: high performance, high packing density, used in large numbers, expensivePGA: fast turnaround time, flexible, performance trade-off

- **initial** : block executed only once
- **$monitor** : like printer (prints whenever any of the variables in brackets change)
- %b : bit
- **$dumpfile**: dump testbench changes to a vcd (value change dump) file
- **$dumpvars (0, testbenchModule)**: All variables to be dumped.
- Test Bench
  - the variables in the initial block that appear on LHS (ie, whose values we are changing) need to be declared as reg
  - Other variables to be declared as wire
- Icarus verilog
  - Iverilog -o <output_file> source.v testbench.v |p <output_file> //runs the simulation and prints results

2 signals

- SystemVerilog supports 4 value levels
  1. 0
  2. 1
  3. x : unknown (all registers initialized to x)
  4. z : high impedance (all unconnected nets)
- Signal strengths (in descending order):

Strength	**Type**
supply	Driving
strong	Driving
pull	Driving
large	Storage
weak	Driving
medium	Storage
small	Storage
highz	High Impedance

- If 2 signals get driven on a wire, stronger signal prevails
- useful for MOS level circuits eg: dynamic MOS

3 Data Types

1. Net
   - Must be continuously driven
   - Can't store a value
   - Represents connection b/w HW elements
   - During synthesis always maps to a wire
   - 1 bit value by default
   - Default value is 'z' (high impedance state)
     - Not driven, neither 1 nor 0
   - Typically used to model connections b/w continuous assignments and instantiations
   - Different types:
     - wire, wor, wand, tri, supply0, supply1, etc.
     - wire and tri are equivalent
       - Wire can only be driven using assign statements
     - wor and wand insert OR and AND respectively at the connection
     - supply0 and supply1 model power supply connections
2. Register
   - Retains/holds last value assigned
   - Typically used to represent storage elements, sometimes can translate to combinational circuits also
   - X = A + B // no 'assign'
   - During synthesis, maps either to 'wire' or a 'storage cell' depending on the context of the assignment
   - Not to be confused with a hardware register, ie flip-flop
   - Register types in SystemVerilog

reg	most common
integer	used for counting
real	floating point numbers
time	simulation time (not used in synthesis)

(a) reg
- default value = x
- can be assigned in synchronism with a clock or otherwise
- Treated as unsigned number
- Used to model actual sequential HW components like counters, shift registers, etc.
- Can only be driven in procedural blocks like always and initial

(b) integer
- signed
- default size is 32 bits (can't specify size)
- Synthesis tool determines the size using data flow analysis

(c) real
- floating point numbers
- if assigned to an integer, rounded off

(d) time
- store simulation time
- *$time* gives current simulation time

4 Module

- $\sim$ Function
- Basic unit of HW
- Can't contain definition of other modules
  - Other modules can be instantiated inside a module
  - A copy of the instantiated module is embedded in the main (instantiating) module for each instantiation
- Allows creation of hierarchy