

SystemVerilog A Design & Synthesis Perspective

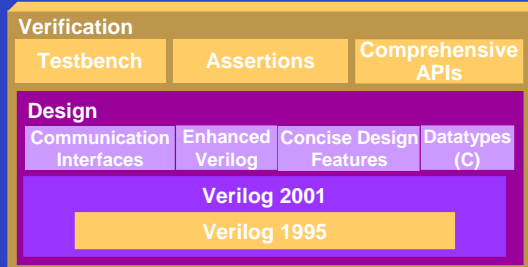
Karen Pieper
R&D Manager, HDL Compiler



Your Interoperability Partner

Insert Paper Title here

What is SystemVerilog?



- Speeds operations – 3-5X concise code
- Unifying design verification – simplifies flow, teamwork
- Evolution from Verilog – rapid incremental, adoption

 Your Interoperability Partner

Concise Coding Constructs for Design

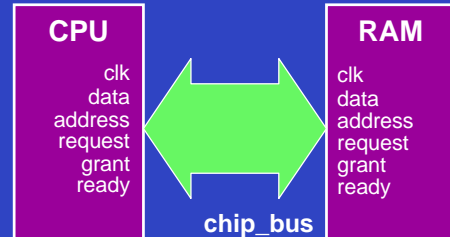
- Interfaces – Define once, use many times
 - Encapsulates communication between modules
 - Can be designed/verified separately
 - Supports multiple levels of abstraction – signal level, transaction
- Structures and user defined data types
 - Improves data modeling, simplifies port lists
- Assignment operators, array slices, etc.

Less code – Same synthesis results as Verilog

Interfaces Capture Chip Infrastructure

```
interface chip_bus (input wire clk);
  wire      request, grant, ready;
  wire [47:0] address;
  wire [63:0] data;
endinterface
```

Connection details are in the interface



```
module CPU (chip_bus io);
  ...
endmodule
```

Modules do not duplicate connection detail

```
module RAM (chip_bus pins);
  ...
endmodule
```

```
module top;
  wire clk;
  chip_bus a(clk); //instantiate the interface

  RAM mem(a); //connect interface to module instance
  CPU cpu(a); //connect interface to module instance
endmodule
```

Netlists do not duplicate connection detail

Operability Partner

More than Just a Bundle of Wires

- Declarations
 - Variables, parameters and other data that is shared
- Tasks and functions
 - Operations on the interface connections
- Always blocks
 - Actions at the highest instantiation level of the bus
- Modports
 - Specification of port directions
- Assertions
 - Protocol checking and other verification
- **Allows simple replacement of a bus**
 - Serial to parallel, parallel to serial
- **Allows a single owner of bus description**

 *Your Interoperability Partner*

Insert Paper Title here

Interface Synthesis

- Interfaces distribute hardware
 - The appropriate contents of an interface are inlined in modules using it
 - Names are scoped appropriately
 - Tasks and functions generate hardware at the callsite
 - Embedded always blocks exist in the instantiating module
 - Changing an interface requires resynthesizing all modules referring to that interface



➤ *Your Interoperability Partner*

Insert Paper Title here

Structs and User Defined Data Types – Improves Readability and Debug

Verilog

```

module fifo (clk, rstp, din_src,
din_dst, din_data,readp,writep,
dout_src,dout_dst, dout_data,
emptyp, fullp);
input          clk;
input          rstp;
input bit [7:0] din_src;
input bit [7:0] din_dst;
input bit [31:0] din_data;
input          readp;
input          writep;
output bit [7:0] dout_src;
output bit [7:0] dout_dst;
output bit [31:0] dout_data;
output logic   emptyp;
output logic   fullp;
.
.
.

```

SystemVerilog

```

typedef struct {
    bit [7:0] src;
    bit [7:0] dst;
    bit [31:0] data;
} packet_t;

```

Define Once

```

module fifo (
input          clk,
input          rstp;
input packet_t din,
input          readp;
input          writep;
output packet_t dout;
output logic   emptyp;
output logic   fullp
);

```

Use many times

Easy to read and debug

Insert Paper Title here

Type Synthesis

- Most type extensions are straightforward
- Structures
 - Similar to synthesizing VHDL
 - Unpacked arrays are preferred for synthesis
 - Aliasing for packed arrays
- Unions
 - Support only packed unions for synthesis today
 - Ensures a single interpretation independent of the implementation
 - Unpacked unions have no guarantee of alignment



 *Your Interoperability Partner*

Insert Paper Title here

Additional Concise Structures

- Port connections
- Return, break, continue, do ... while
- Assignment operators
- Multi-dimensional array slices

- **All have more verbose Verilog 2001 equivalents**
- **All are synthesized today**

 **Your Interoperability Partner**

Insert Paper Title here

Unifying Design and Verification

- Reducing the infamous “simulation synthesis mismatch”
 - Unique, priority
- Ensuring that what was intended is what was written
 - Always_comb, always_latch, always_ff
- Documenting and checking assumptions as the design is coded
 - Assertions

 *Your Interoperability Partner*

Insert Paper Title here

Unique and Priority

- Reduces simulation/synthesis mismatches
- Unique is `parallel_case` and `full_case`
 - Simulation error if there is more than one true condition
 - Simulation error if condition is not enumerated
- Priority is the same as `full_case`
 - Simulation error if condition is not enumerated
- Apply to both case statements and `if..else if` chains

 Your Interoperability Partner

Insert Paper Title here

Always_* Forms

- Always_comb -- represents combinational logic
 - Simulation activation better matches synthesis activation
 - Warning if latches or flip-flops are inferred
- Always_latch -- represents latch logic
 - Simulation activation better matches synthesis activation
 - Warning if a latch is not present
- Always_ff @(exp) -- represents flip-flop logic
 - Limits always block to one activation
 - Warning if a flip-flop is not present

 Your Interoperability Partner

Insert Paper Title here

Synthesis QoR Impact

- An example Verilog design rewritten using SV constructs
 - Typedefs
 - Structures
 - Interfaces
 - Modports
 - Always_* forms

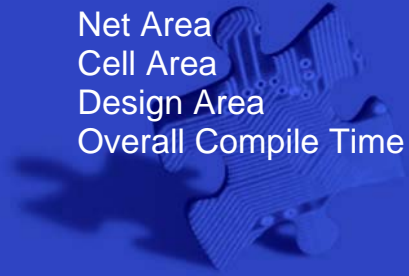


 *Your Interoperability Partner*

Insert Paper Title here

Synthesis QoR

	<u>Verilog</u>	<u>SystemVerilog</u>
Levels of Logic	10.00	10.00
Critical Path Length	6.38	6.38
Critical Path Slack	-6.38	-6.38
Total Negative Slack	-701.23	-701.91
No. of Violating Paths	204.00	204.00
Hierarchical Cell Count	242	242
Leaf Cell Count	6261	6266
Combinational Area	10590	10590
Noncombinational Area	14903	14907
Net Area	0	0
Cell Area	25493	25497
Design Area	25493	25497
Overall Compile Time	105.60	105.67



 *Your Interoperability Partner*

Insert Paper Title here

Summary

- Migration to SystemVerilog is easy
 - Verilog designs will largely work unchanged
- SystemVerilog raises the abstraction level improving productivity
 - 3x to 5x code size reduction
 - Better verification with synthesis
 - Faster verification through tighter integration with testbenches

  *Your Interoperability Partner*

Insert Paper Title here

EDA and IP Suppliers:

- For more information on SystemVerilog and synthesis, join the SystemVerilog Catalyst Program
- www.synopsys.com/partners/systemverilog



➤ *Your Interoperability Partner*

Insert Paper Title here