co-design automation

SUPERLOG

HDLCon 2001 Tutorial 6

A Practical Approach to System Verification and Hardware Design
The PUBLIC SUBSET Of The SUPERLOG Language

Peter Flake, Chief Technical Officer
David Rich, AE Director

---Lingling

Outline

- SUPERLOG Introduction
  - Why SUPERLOG
  - Language conponents
  - Streamlined methodology
- Language Evolution
  - Code fragments
  - Datatype System
  - Interfaces

Why SuperLog

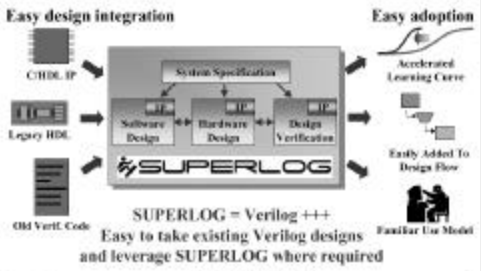
- System Specification language: SDL
- Software Programming Language: C
- Hardware Design Language: Verilog and VHDL
- Testbenth language: Vera

Why SuperLog

Thinking of a single language for four purposes:

- Describe hardware in C is difficult
- Verilog is poor as a general-purpose programming language.
  - No dynamic memory allocation
  - Poor built-ininput and text processing is extremely difficult
- VHDL also hampered by three problems

SUPERLOG - Evolution, Not Revolution!

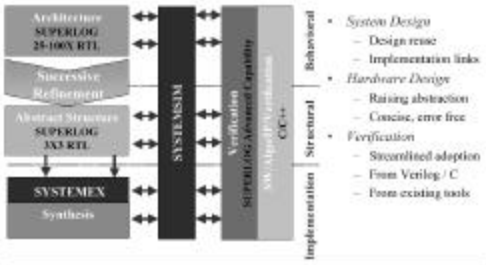


co-design automation

Unifying Language Components

- **Verilog 95 and 2K** -- superset
- **C and C++** --structures, pointers, types, etc.
- **SUPERLOG Extended Synthesizable Subset (ESS)**
- **System Abstraction** --include control communication abstraction, queues, processes
- **Verification Power**

Streamlined Methodology Components



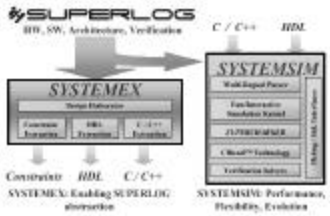
- *System Design*
  - Design reuse
  - Implementation links
- *Hardware Design*
  - Raising abstraction
  - Concise, error free
- *Verification*
  - Streamlined adoption
  - From Verilog / C
  - From existing tools

co-design automation

© 2001 Co-Design Automation, Inc.
Tutorial presented at HDLcon 2/28/2001 - SUPERLOG Public Subset

The Co-Design Automation Product Line

Streamlining Design, Accelerating Verification



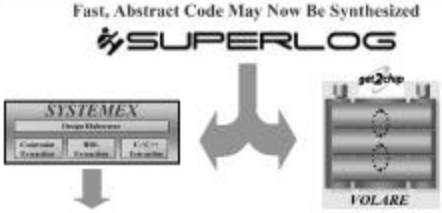
- Methodology performance
- Multilingual SoC flexibility
- Integration of key functions

co-design automation

© 2001 Co-Design Automation, Inc.
Tutorial presented at HDLcon 2/28/2001 - SUPERLOG Public Subset

SUPERLOG Abstraction to Implementation

Fast, Abstract Code May Now Be Synthesized



Synthesis Tools

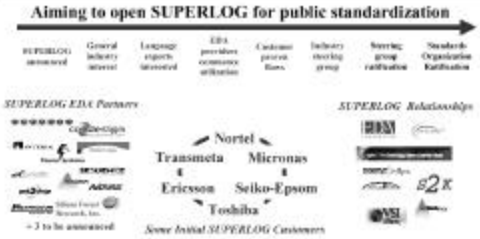
Get2Chip's Volare Architectural Synthesizer reads SUPERLOG directly

co-design automation

© 2001 Co-Design Automation, Inc.
Tutorial presented at HDLcon 2/28/2001 - SUPERLOG Public Subset

Co-Design Automation At Large

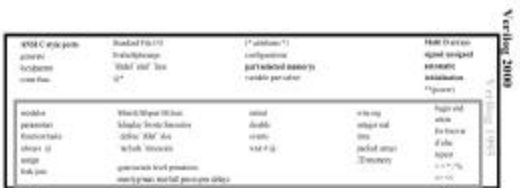
Aiming to open SUPERLOG for public standardization



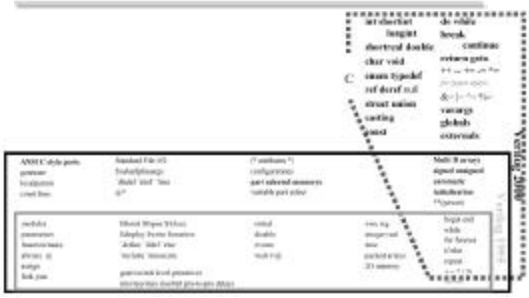
co-design automation

© 2001 Co-Design Automation, Inc.
Tutorial presented at HDLcon 2/28/2001 - SUPERLOG Public Subset

SUPERLOG –Building on Verilog



SUPERLOG –Building on Verilog



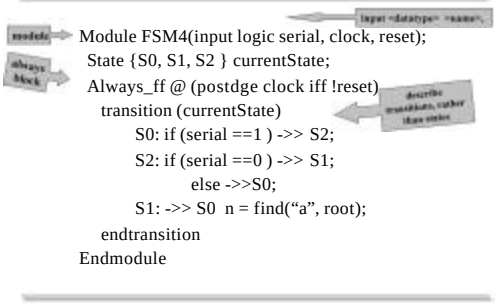
SUPERLOG –Building on Verilog

SUPERLOG code fragment

```
Typedef struct {string s; ref node
        left, right; } node;
//global data
Ref node n, root; //pointers to nodes
Int visited = 0;
Function ref node find( string str,
        ref node parent);
   if (parent == null) return null;
    visited ++;
   if (str == parent -> s) return
        parent;
   if (str == parent -> s) return
        find( str, parent -> left);
   if (str == parent -> s) return
        find( str, parent -> right);
endfunction
```

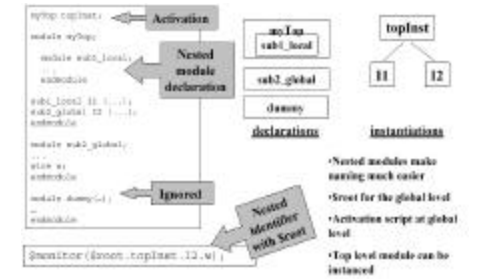
```
Module FSM4(input logic serial,
    clock, reset);
State {S0, S1, S2 } currentState;
Always_ff @ ( postdge clock
    iff !reset)
    transition (currentState )
        S0: if (serial ==1 ) ->>
    S2;
        S2: if (serial ==0 ) ->>
    S1;
                else ->>S0;
        S1: ->> S0  n = find("a",
    root);
    endtransition
Endmodule
```

Learning SUPERLOG

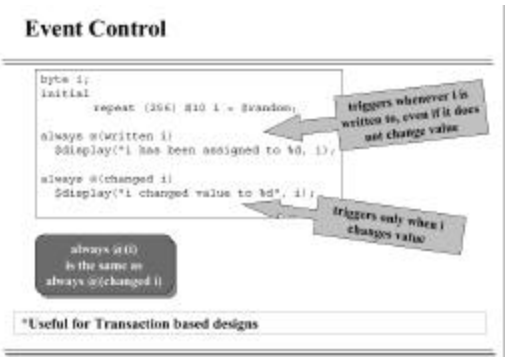



```
Module FSM4(input logic serial, clock, reset);
  State {S0, S1, S2 } currentState;
  Always_ff @ (postdge clock iff !reset)
    transition (currentState)
        S0: if (serial ==1 ) ->> S2;
        S2: if (serial ==0 ) ->> S1;
                else ->>S0;
        S1: ->> S0  n = find("a", root);
    endtransition
Endmodule
```

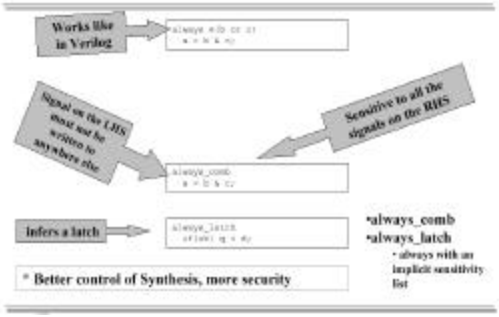
Module



Event Control



Always Block Enhancements



Datatypes

•Compatible to C

```
char c;          // 8 bit signed integer
int i;           // 32 bit signed integer
```

•Use typedef to get C compatibility

```
//          Superlog          C
typedef     shortint          short;
typedef     longint           longlong;
typedef     real              double;
typedef     shortreal         float;
```

•Basic SUPERLOG Datatype

```
bit b;     // single bit 0 or 1
logic w;   // 4-valued logic, x 0 1 or z as in Verilog
time t;    // 64 bit time value
byte b8;   // 8 bit signed integer
```

Extra SUPERLOG datatypes

Timescales

- Time constant
  - Legal time units

  Are s, ms, us, ns, ps, fs
- Time Rules

separate timeunit and timeprecision commands

```
`timescale 1ns/1fs
```

```
timeunit 1ns;
timeprecision 1fs;
top t;
module top;
```

You can use delays with an exists, like in Verilog

No space between digit and letter!

```
#10 a <= 1;
#5ns b <= 1b;
#10 $display("%d", b);
```

You can also specify units with your delays

•legal time units are s, ms, us, ns, ps, fs

•The default is #1 = 1s

What Is an Interface?

```
int i;
logic [7:0] a;

typedef struct {
  int i;
  logic [7:0] a;
} s_type;
```

At the simplest level an interface is to a wire what a struct is to a variable

```
wire a;
int [31:
module a (m, i[1:

interface intf;
  int i;
  wire [7:0] a;
endinterface
```

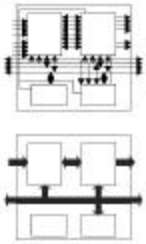
```
wire a;
int [31:
module a (m, i[1:
```

You can think of a wire as a built in interface

*Encapsulates communication like a struct encapsulates data

co-design automation
© 2001 Co-Design Automation, Inc.
Tutorial presented at HDLcon 2/26/2001 - SUPERLOG Public Subset

Systems Without the Interface Construct



schematics map to a Verilog netlist

```
module mem_controller
  input wire clk, global_reset,
  input wire [63:0] data_bus,
  input wire ctrl, ctrl, ctrl,
  input bit pred, pred, pred,
  input wire a1, a1, T_pdec,
  input wire f_dec, f_dip,
  f_ali, g_tyf, g_tyl, g_tyf,
  ....
```

As in Verilog: Hierarchy is only in the modules and not in the interfaces which are all exploded into separate wires

co-design automation
© 2001 Co-Design Automation, Inc.
Tutorial presented at HDLcon 2/26/2001 - SUPERLOG Public Subset

Modports

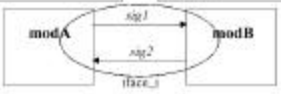
```
interface iface_1;
  bit sig1, sig2;
  int interval;
  modport modA(output sig1, input  sig2),
          modB(input  sig1, output sig2);
endinterface
```

Not visible outside of this interface

Specifies the accessibility and direction of interface signals

```
module modA(iface_1.modA iface);
...
  iface.sig1 <= 1;
  w <= iface.sig2;
...
endmodule
```

```
module modB(iface_1.modB iface);
...
  iface.sig2 <= 1;
...
endmodule
```



co-design automation
© 2001 Co-Design Automation, Inc.
Tutorial presented at HDLcon 2/26/2001 - SUPERLOG Public Subset