DVCon 2013
Design & Verification Conference & Exhibition

February 25-28, 2013
DoubleTree, San Jose

accellera
SYSTEMS INITIATIVE

A Tale of Two Languages: SystemVerilog & SystemC

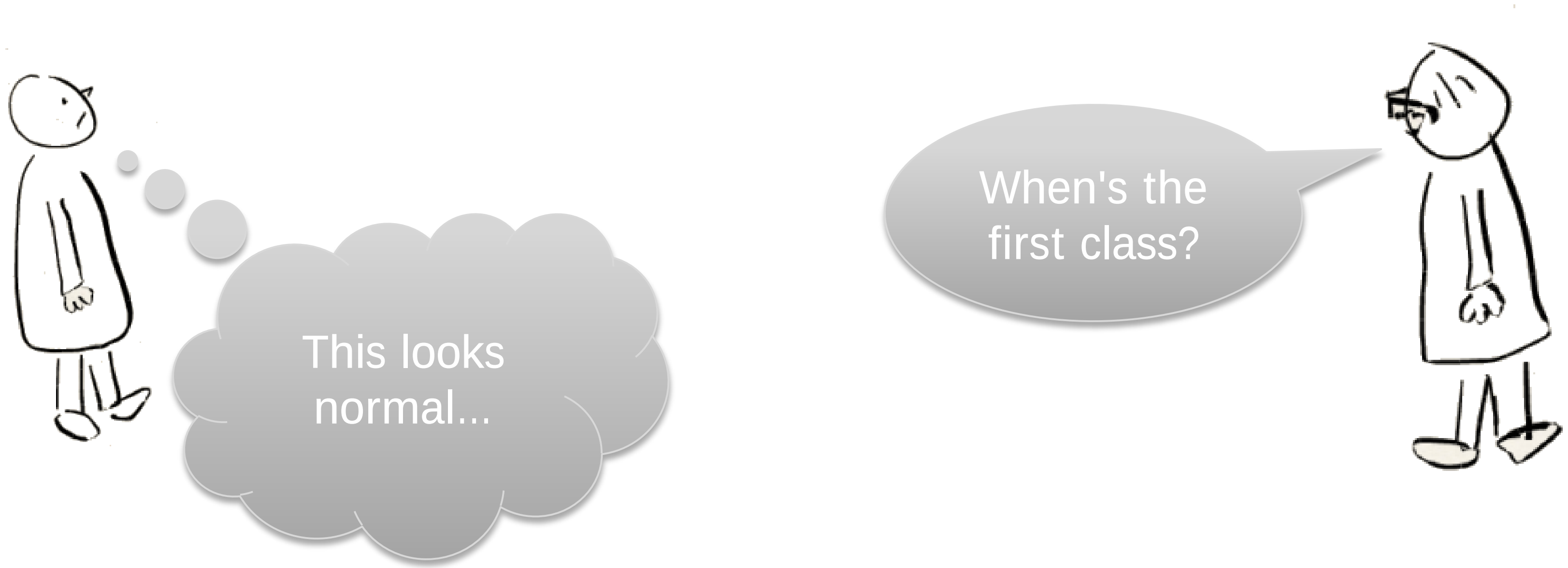
by
David C Black
Senior MTS
Doulos

DOULOS

Two languages...

New corporate policy

- *HR memos must be written in Borg.*
- *Programmers will henceforth use Romulan.*
- *Hardware designs shall be written in Klingon.*



Sponsored By:

System*?

- Unfortunate "System" prefix confuses many
  - <u>System</u>Verilog
    - A system for hardware design & verification
    - Significantly improved Verilog combining HDL with HVL
  - <u>System</u>C
    - A system for using C++ for abstract modeling
    - Used to model large electronic system-level designs (ESL)
- Intended for very different applications
- Best practice: use both cooperatively

What is SystemVerilog?

- Great RTL description language
  - Features to align gate-level simulation to RTL
  - C-style data and operators

```
module mux (
  input byte a, b, c,
  input [1:0] sel,
  output integer f);
  // combinational
  always_comb
    //parallel case
    unique if ( sel == 2'b10 )
      f += a;
    else if ( sel == 2'b01 )
      f = b;
    else
      f = c;
endmodule
```

What is SystemVerilog?

- Fantastic language for constrained random & coverage driven verification
- Solid OOP language for UVM & other reusable hardware verification methodologies

```
class instruction;
  rand bit [2:0] m_opcode;
  rand bit [1:0] m_mode;
  rand shortint unsigned m_data;
  constraint assert_mode {
    m_opcode[2]==0 ->
m_mode==2'b11;
  }
  covergroup cg @(posedge clk);
    coverpoint m_opcode;
    coverpoint m_mode;
    coverpoint m_data {
      bins tiny    [8] = {[0:7]  };
      bins moderate[8] = {[8:255]};
      bins huge    [8] = {[256:$]};
    }
  endgroup
endclass: instruction
```

What is "System" C?

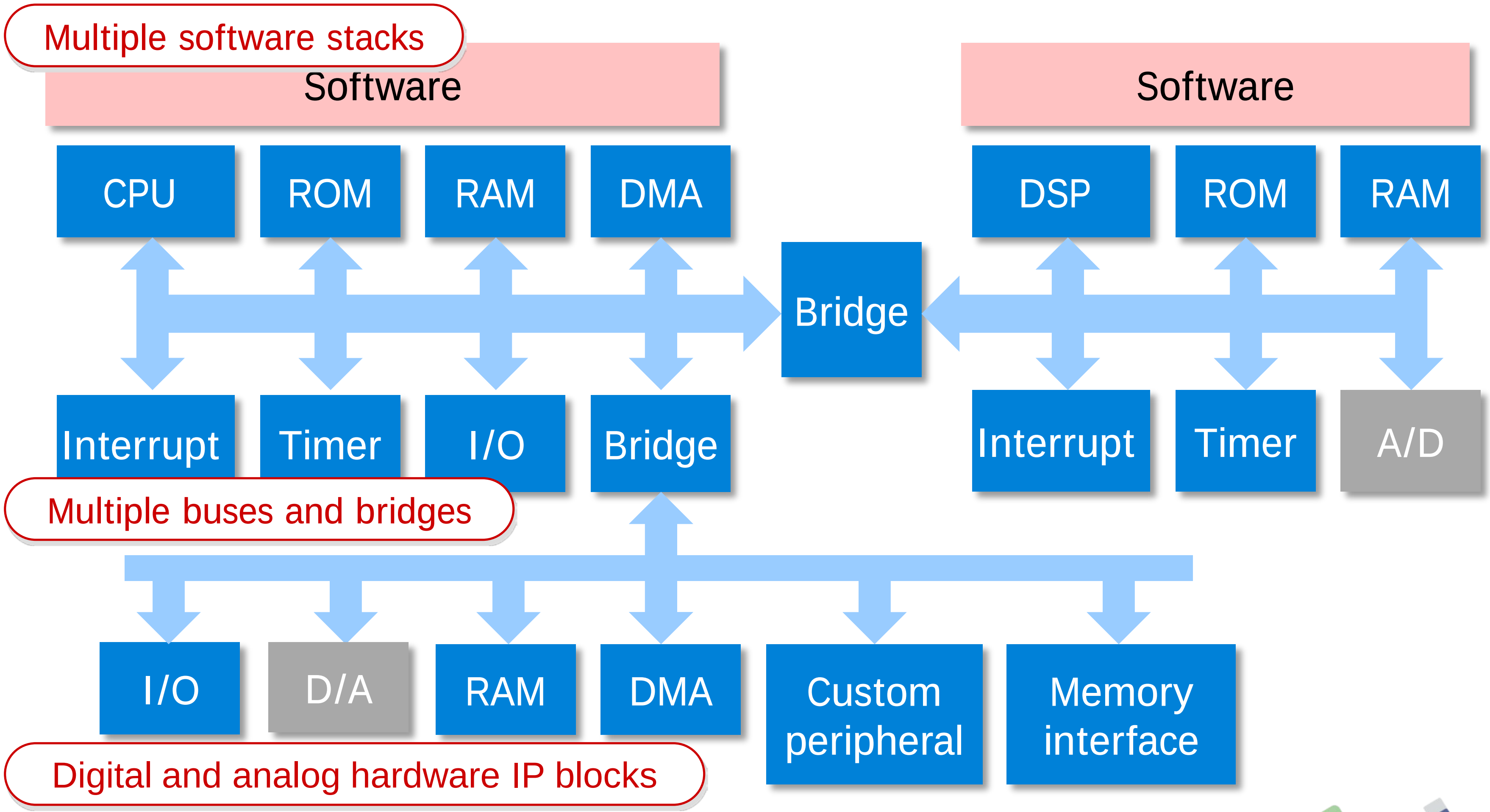
Approach using a C++ library to create abstract hardware/software models with less detail than RTL to enable early analysis & software development

- C++ enables
  - Draw on vast libraries
  - Common language with software team
- Open-source enables
  - Wide distribution for very large teams
  - Share with customers

- Less detail means
  - Fast creation (earlier)
    - More experiments
    - Early performance analysis
  - Fast simulation
    - Allows software development
    - Verification reference model

What is SystemC?

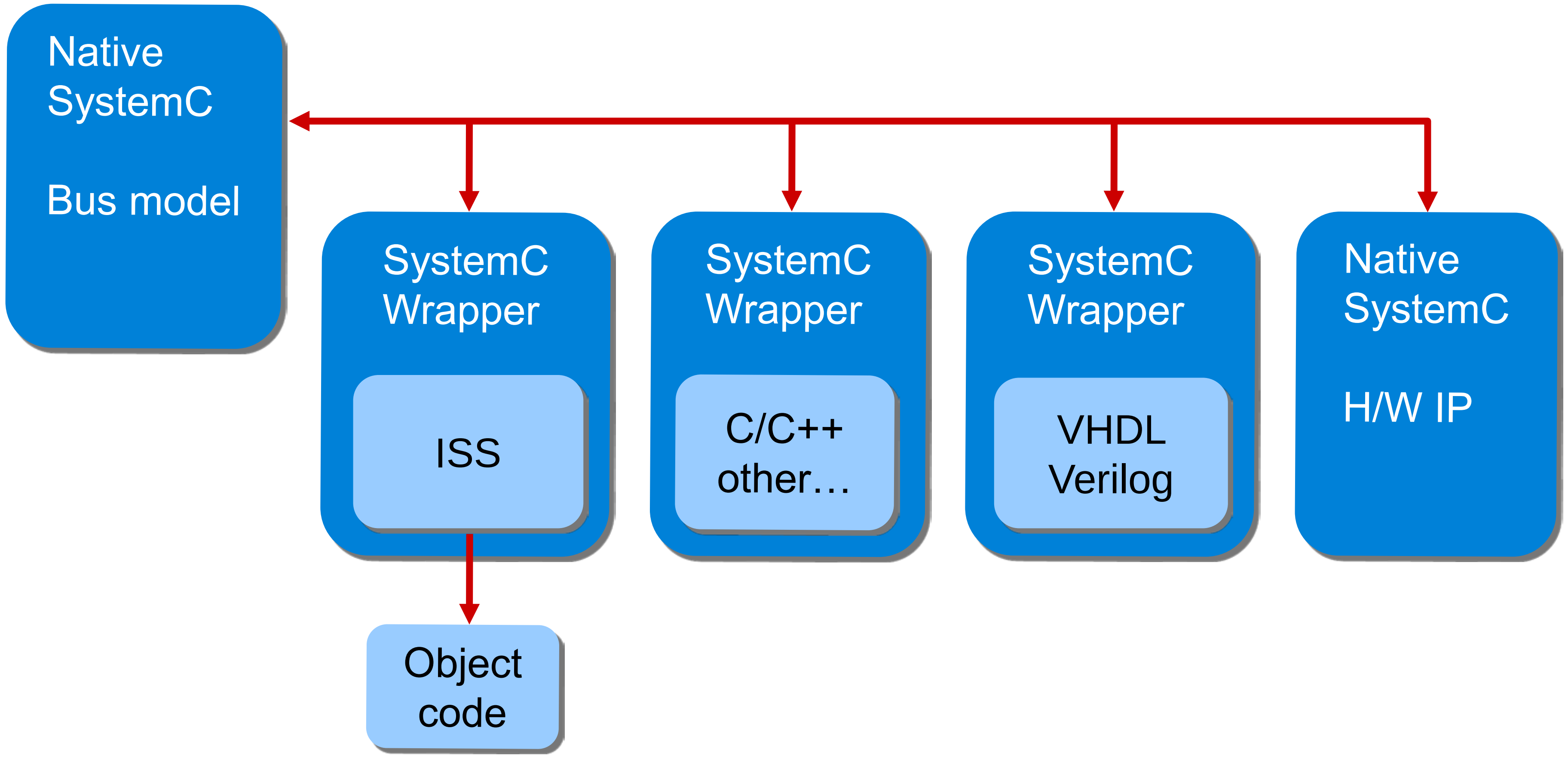
- Systems at RTL level simulate too slowly...



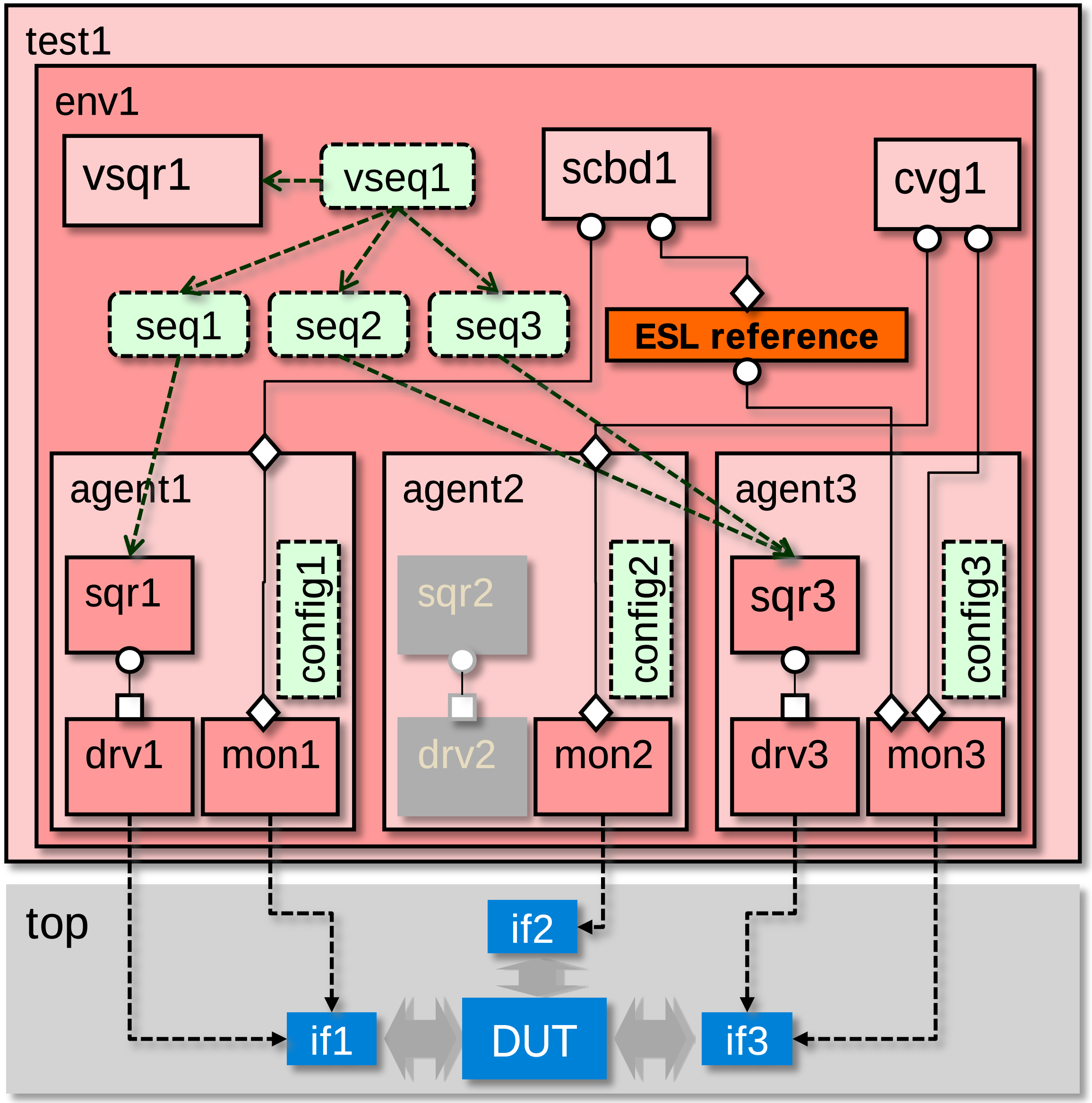
Sponsored By:

Co-existence

- Mixed abstraction levels play well



Co-existence with UVM



Side by Side: Modules

(Containers for blocks of code)

SystemVerilog

```
module Design
( input  logic [7:0] d
, output logic [7:0] q
);
    …
endmodule: Design
```

SystemC

```
SC_MODULE(Design) {
    sc_in <sc_lv<8> > d;
    sc_out<sc_lv<8> > q;
    …
};
```

DVCon 2013 Design & Verification Conference & Exhibition

Sponsored By: accellera SYSTEMS INITIATIVE

Side by Side: Data

SystemVerilog

```
logic [3:0] l;
int         i;
bit         b;
string      txt;
typdef struct { int a, b; } S;
S           s = ‘{1,2};
time        t;
```

SystemC

```
sc_lv<4> l;
int      i;
bool     b;
string   txt;
struct S { int a, b; };
S        s{1,2};//C++11
sc_time  t;
```

Side by Side: Containers

SystemVerilog

```
T1 fixedArray[N];
T1 dynamicArray[];
T1 associativeAry[T2];
T1 queue[$];
```

SystemC

```
std::array<T1,N> fixedArray;
std::vector<T1> dynamicArray;
std::map<T2,T2> associativeAry;
std::deque<T1>  queue;
```

Side by Side: Conditionals

Sponsored By:

SystemVerilog

```
if (EXPR) STMT1
else      STMT2

case (EXPR)
  EXPR: STATEMENT
  default: STATEMENT
endcase
```

SystemC

```
if (EXPR) STMT1
else      STMT2

switch (EXPR) {
 case CONST: STATEMENT; break;
 default: STATEMENT;
}
```

Side by Side: Loops

Sponsored By:

SYSTEMS INITIATIVE

SystemVerilog

```
while(EXPR) STATEMENT
do STATEMENT while (EXPR);
for (int i=0;i!=max;++i) STMT
forever STATEMENT
foreach (CONTAINER[i]) STMT
```

SystemC

```
while(EXPR) STATEMENT
do STATEMENT while (EXPR);
for (int i=0; i!=max; ++i) STMT
for(;;) STATEMENT
for (auto i:CONTAINER)STATEMENT
```

Sponsored By:

Side by Side: Processes

SystemVerilog

```
input      clock;
input  int d;
output int q;

always_ff @(posedge clock)
begin :REGS
  q <= d;
end
```

SystemC

```
sc_in<bool> clock;
sc_in<int>  d;
sc_out<int> q;

SC_METHOD(REGS);
sensitive << clock.pos();
…
void REGS(void) {
    q->write(d);
}
```

Side by Side: Fork/Join

SystemVerilog

```
fork
  begin STATEMENTS… end
  begin STATEMENTS… end
join
```

SystemC

```
FORK
 sc_spawn([&](){STATEMENTS…}),
 sc_spawn([&](){STATEMENTS…})
JOIN
```

Side by Side: Dynamic Processes

Sponsored By:

SystemVerilog

```
process h;
fork
  begin
    h = process::self();
    STATEMENTS…
  end
join_none

wait(h.status !=
       process::FINISHED);
```

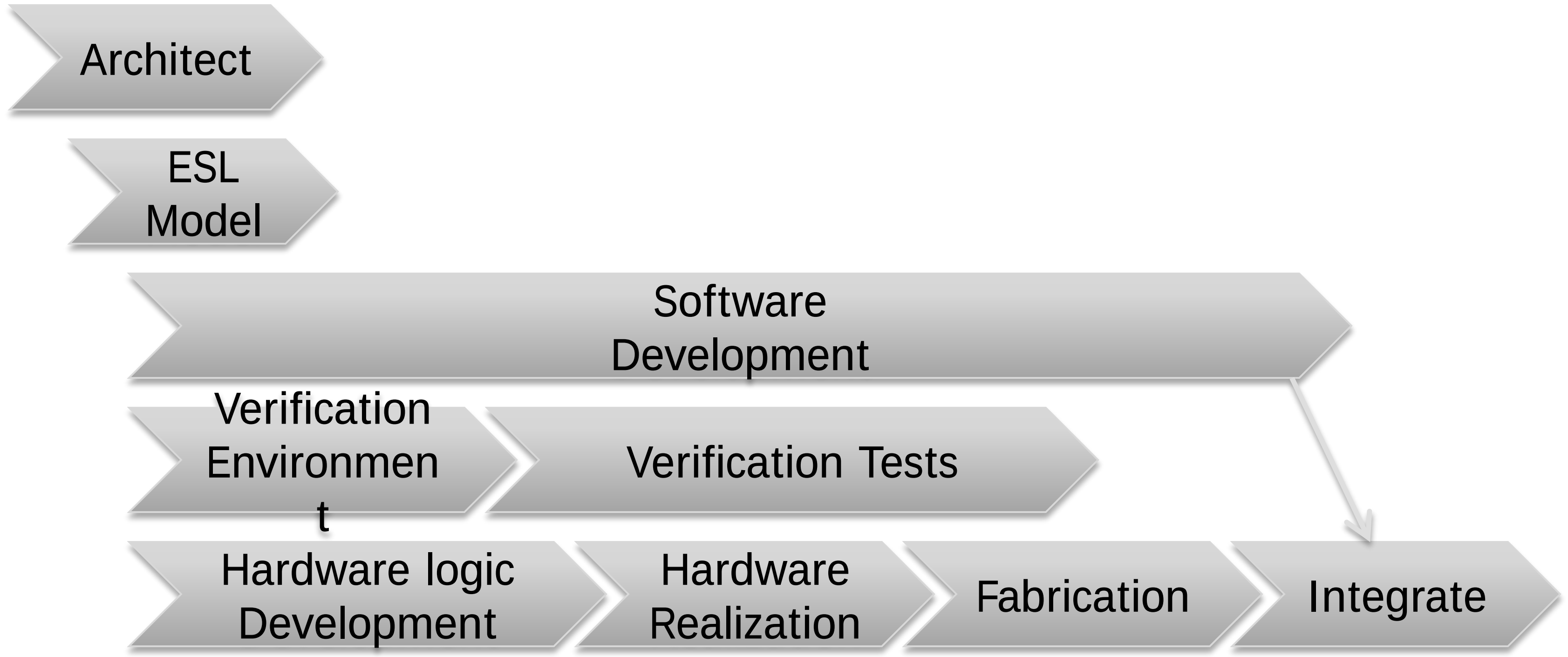
SystemC

```
auto h = sc_spawn([&](){
  STATEMENTS…
});










wait(h.terminated_event());
```

A Project Schedule

Sponsored By:

- Rationales for selecting language



Open issues

- Interoperability between SystemVerilog & SystemC
  - Need TLM 2.0 standard interface
  - Need configuration controls (for tools & models)
- Common register abstraction
- Native C++ DPI

Sponsored By:

SYSTEMS INITIATIVE

Concluding Remarks

- Different needs – different languages
  - Architecture
  - Software
  - Verification
  - Hardware
- Co-existence and interoperability required
  - Enable the entire team
  - No surprises
- Education key
  - Understand the goal
  - Learn to appropriately use the language

Sponsored By:

Thank You