A User's Experience with SystemVerilog

Jonathan Bromley and Michael Smith

Doulos Ltd

Ringwood, U.K. BH24 1AW

jonathan.bromley@doulos.com
michael.smith@doulos.com

Objectives

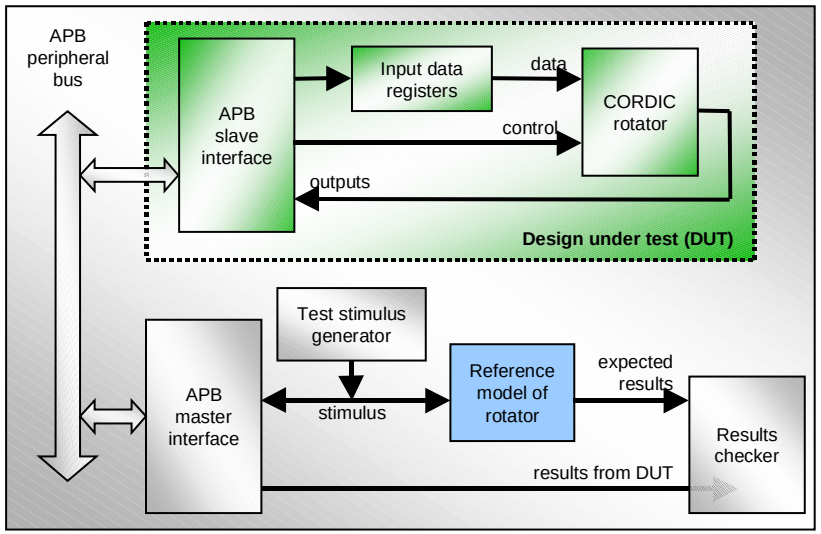
Jonathan Bromley
Michael Smith
DOULOS

- Practical use of SystemVerilog
  - Synopsys tools (VCS, Design Compiler)
- Aim for no-compromise SystemVerilog methodology
  - New data types wherever appropriate
  - Use variables everywhere - no nets!
  - Interfaces for better abstraction of interconnect
  - Transaction-driven test fixture, assertion checking
- Awareness of the landscape
  - Work around tool limitations - it’s very early days yet!
  - Note upcoming SystemVerilog 3.1a changes/enhancements

Outline

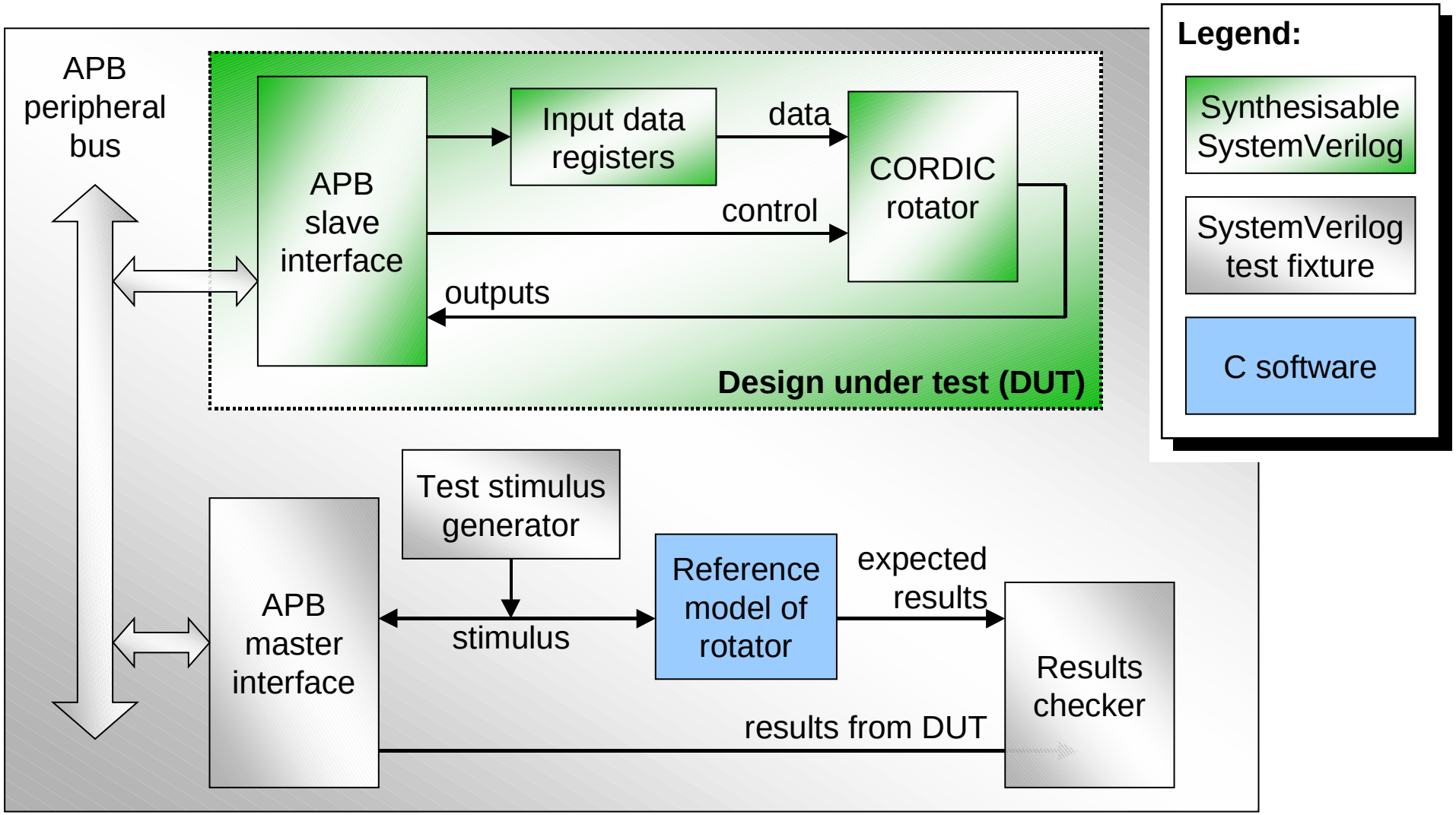
Jonathan Bromley
Michael Smith
DOULOS

- Overview of design and test fixture
- Using SystemVerilog in the design
- Challenges in bus modelling
- Using SystemVerilog in verification
- Challenges in transaction-based verification



Design and Test Fixture

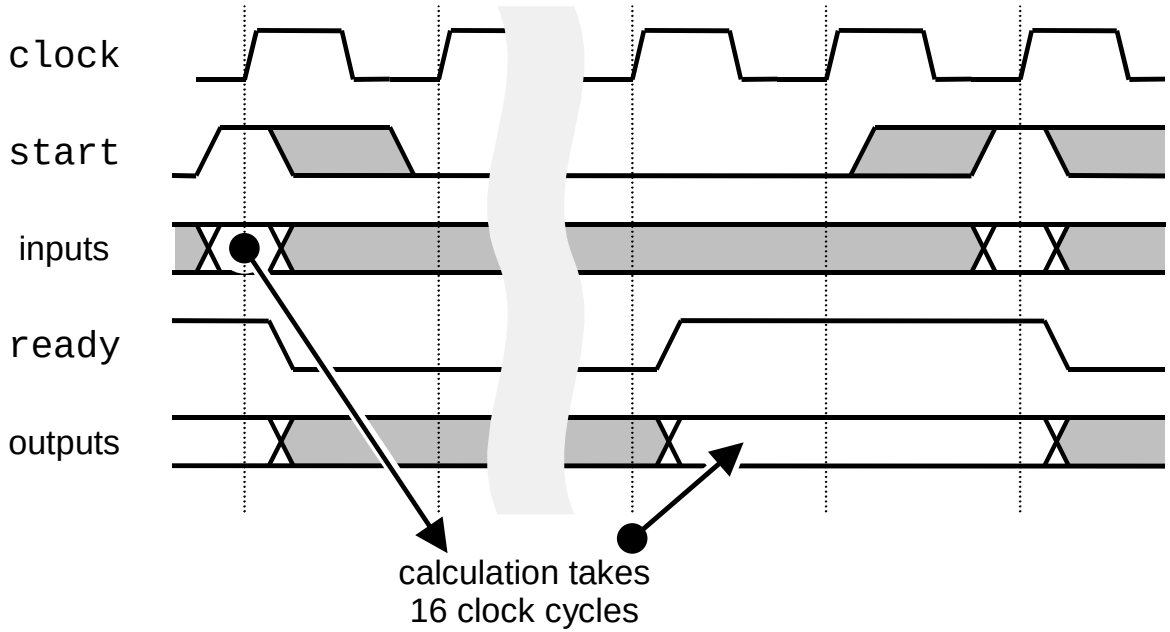
Jonathan Bromley
Michael Smith
DOULOS



Practical Problems That Make It Interesting

Jonathan Bromley
Michael Smith
DOULOS

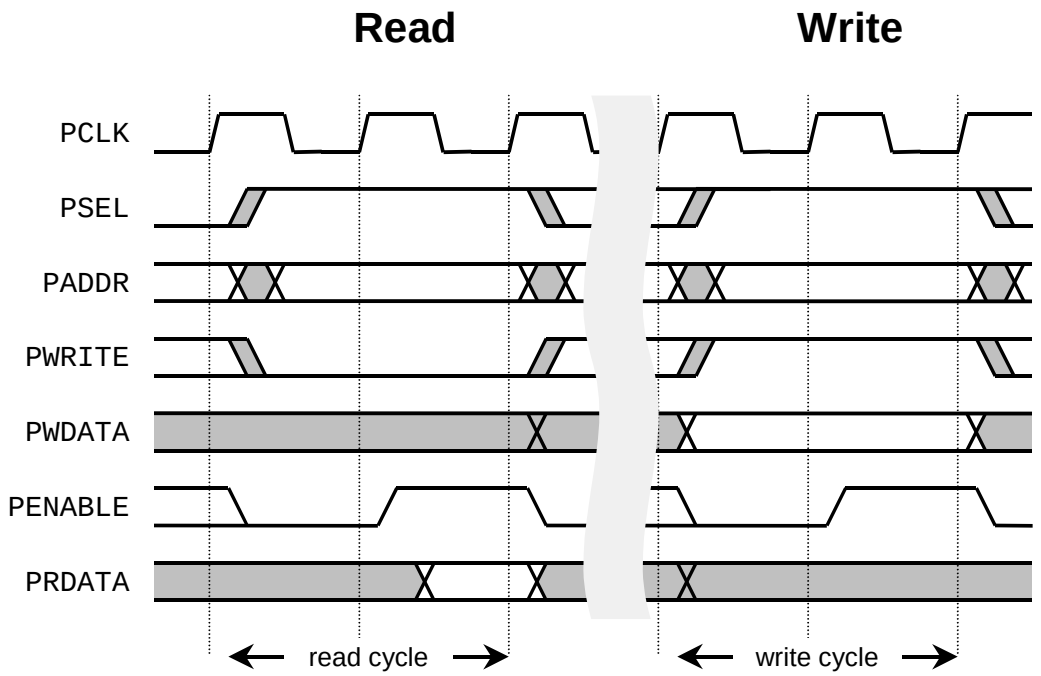
- Design needs 16 clock cycles to perform a calculation
  - start/ready handshake flags appear in bus-visible registers
- Test fixture must mimic software behaviour
  - Set up inputs
  - Hit start
  - Poll until ready
  - Read outputs



APB Bus

Jonathan Bromley
Michael Smith
DOULOS

- APB is a simple synchronous bus
  - Intended for small numbers of slow peripherals
  - Only one master
  - Only two kinds of bus cycle:
  - Every transfer takes 2 clock cycles
- Part of AMBA specification (ARM Ltd)
- Tailored for on-chip applications



Verilog-2001 At Last!

Jonathan Bromley
Michael Smith
DOULOS

- Life is better…
  - signed values, signed arithmetic
  - localparam
  - ANSI-style port declarations
  - multi-dimensional arrays
  - generate
  - named parameter association
- We have avoided recommending Verilog-2001 in training…
  - patchy tool support
- …but support is now complete in many tools
- Irritations…
  - can’t define localparam within generate block
  - multi-dimensional array ports ???

Synthesisable SystemVerilog

Jonathan Bromley
Michael Smith
DOULOS

- Data types to match the problem
- packed struct, union for different registers on common bus
- Structural driver on variable
  - full power of variable data type system available throughout a design
- Note: can't mix structural and procedural drivers on fields of a packed object

```
typedef struct packed {
  logic [15:10] Opcode;
  logic [9:0] Operand;
} T_Instr;

typedef struct packed {
  logic [15:1] Junk;
  logic Ready;
} T_StatusWord;

module ...
  T_Instr Instr;
  T_StatusWord Stat;
  logic [15:0] Databus;
  ...
  assign Databus =
       Adr[0] ? Instr: Stat;
```

Synthesisable SystemVerilog

Jonathan Bromley
Michael Smith
DOULOS

- Inquiry function `$bits()` for parameterisation via types
  - Makes VHDL users very happy!
- Where to put this typedef?
  - SV3.1a will offer *packages*

```
typedef struct packed {
  logic [15:10] Opcode;
  logic [9:0] Operand;
} T_Instr;
```

```
module ...
  localparam
    Width = $bits(T_Instr);
  T_Instr Instr;
  logic [Width-1:0] Databus;
  ...
  assign Instr = Databus;
```

Interfaces for Design

Jonathan Bromley
Michael Smith
DOULOS

- Interface port is like a handle to a module instance

```
module Top;
  Rst_Itf R1;
  Device D1(R1.RTL);
  Rst_Itf R2;
  Device D2(R2.RTL);
```

R1

```
interface Rst_Itf;
  logic Reset;
  modport RTL (input Reset);
  ...
```

D1

```
module Device (
  Rst_Itf.RTL  Wires
);
  always @(Wires.Reset) ...
```

R1.Reset

R2

```
interface Rst_Itf;
  logic Reset;
  modport RTL (input Reset);
  ...
```

D2

```
module Device (
  Rst_Itf.RTL  Wires
);
  always @(Wires.Reset) ...
```

R2.Reset

Interfaces for Design

Jonathan Bromley
Michael Smith
DOULOS

- Interface port is like a handle to a module instance

```
module Top;
  Rst_Itf R1;
  Device D1(R1.RTL);
  Rst_Itf R2;
  Device D2(R2.RTL);
```

Cross-module references - *synthesis issues?*

R1

```
interface Rst_Itf;
  logic Reset;
  modport RTL (input Reset);
  ...
```

R2

```
interface Rst_Itf;
  logic Reset;
  modport RTL (input Reset);
  ...
```

D1

```
module Device (
  Rst_Itf.RTL  Wires
);
  always @(Wires.Reset) ...
```

D2

```
module Device (
  Rst_Itf.RTL  Wires
);
  always @( Wires.Reset ) ...
```

Interfaces for Design

Jonathan Bromley
Michael Smith
DOULOS

```
module Top;
  Rst_Itf R1;
  Device D1(R1.RTL);
  Rst_Itf R2;
  Device D2(R2.RTL);
```

Cross-module references - *synthesis issues?*

modport controls visibility and direction - *not yet respected in VCS7.1*

R1

```
interface Rst_Itf;
  logic Reset;
  modport RTL (input Reset);
  ...
```

R2

```
interface Rst_Itf;
  logic Reset;
  modport RTL (input Reset);
  ...
```

D1

```
module Device (
  Rst_Itf.RTL  Wires
);
  always @(Wires.Reset) ...
```

D2

```
module Device (
  Rst_Itf.RTL  Wires
);
  always @(Wires.Reset) ...
```

Interfaces in the Test Fixture

Jonathan Bromley
Michael Smith
DOULOS

- Opportunity for transaction-level modelling via `import task`

```
module Top;
  Rst_Itf R;
  TestCase T(R.Beh);
  Device D(R.RTL);
endmodule
```

```
module Device (Rst_Itf.RTL Wires);
  always @(posedge Wires.Reset)
    ...
```

```
module TestCase (Rst_Itf.Beh RstGen);
  initial begin
    RstGen.Pulse(20);
    ...
```

```
interface Rst_Itf;
  logic Reset;
  task Pulse(input time T);
    ...
  endtask
  modport Beh (
   import task Pulse;
  );
  modport RTL (input Reset);
endinterface
```

Interfaces in the Test Fixture

Jonathan Bromley
Michael Smith
DOULOS

- Opportunity for transaction-level modelling via `import task`

```
module Top;
  Rst_Itf R;
  TestCase T(R.Beh);
  Device D(R.RTL);
endmodule
```

```
module Device (Rst_Itf.RTL Wires);
  always @(posedge Wires.Reset)
    ...
```

```
module TestCase (Rst_Itf.Beh RstGen);
  initial begin
    RstGen.Pulse(20);
    ...
```

```
interface Rst_Itf;
  logic Reset;
  task Pulse(input time T);
    ...
  endtask
  modport Beh (
   import task Pulse;
  );
  modport RTL (input Reset);
ace
```

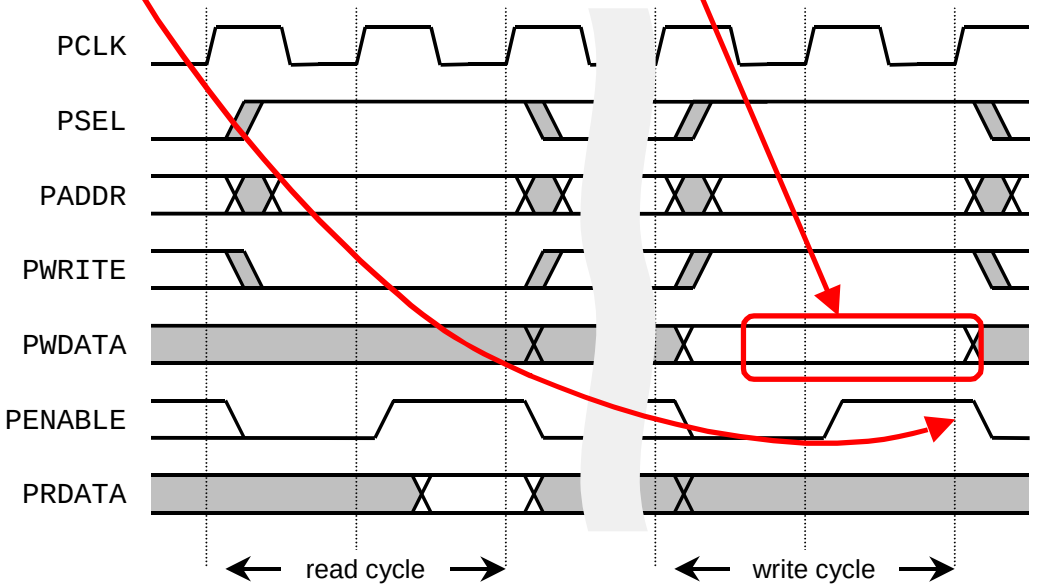
Just like traditional BFM -
but don't need to know its instance name in the hierarchy

Assertions

Jonathan Bromley
Michael Smith
DOULOS

- Natural way to describe bus activity, transactions
- Permanently active checker

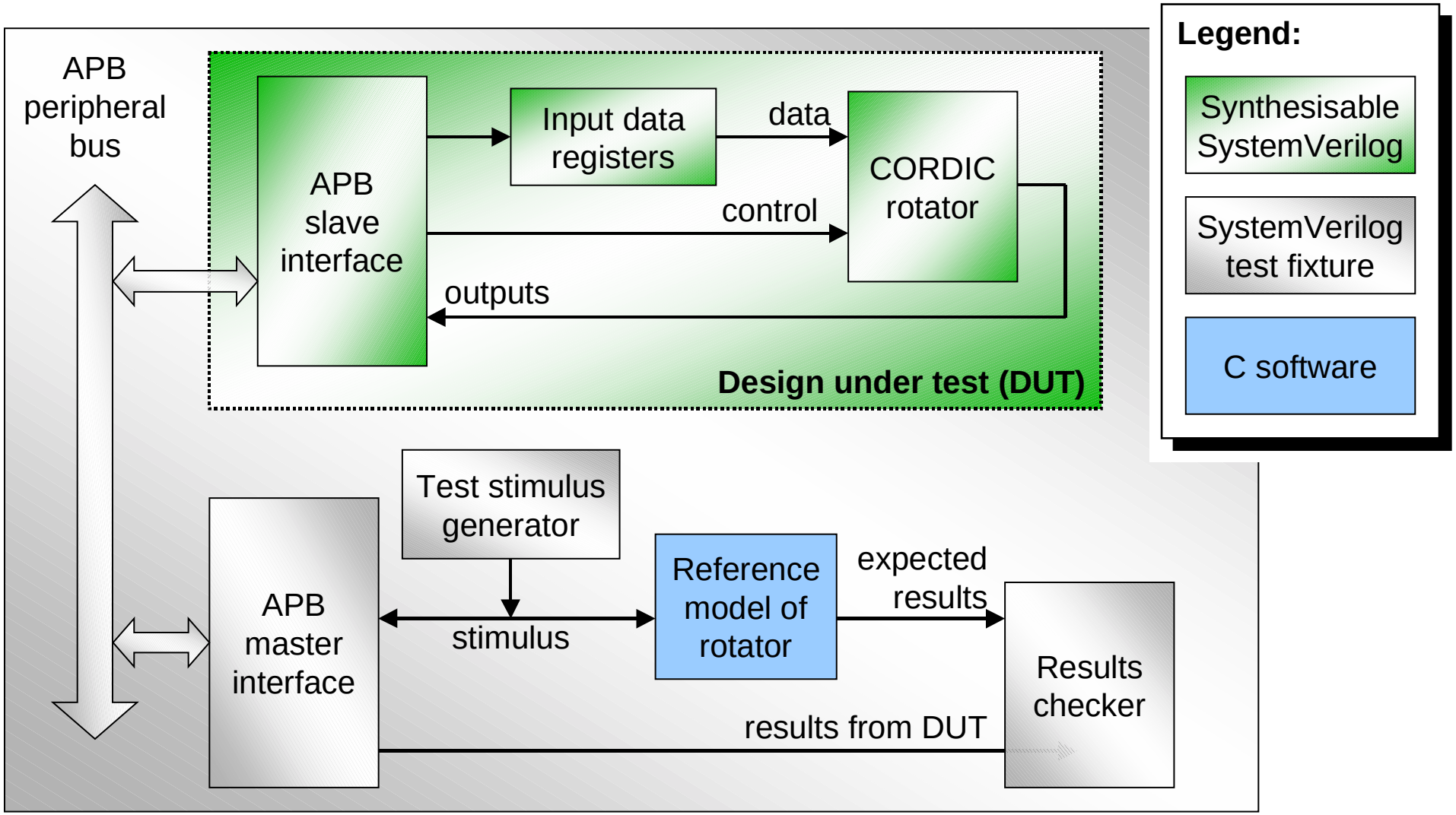
```
property WriteDataStable;
  @(posedge PCLK) ((PENABLE && PWRITE) |-> $stable(PWDATA));
endproperty
```



- Coverage - did it happen?
  - automatic, inflexible in SV3.1
  - 3.1a adds `coverpoint`

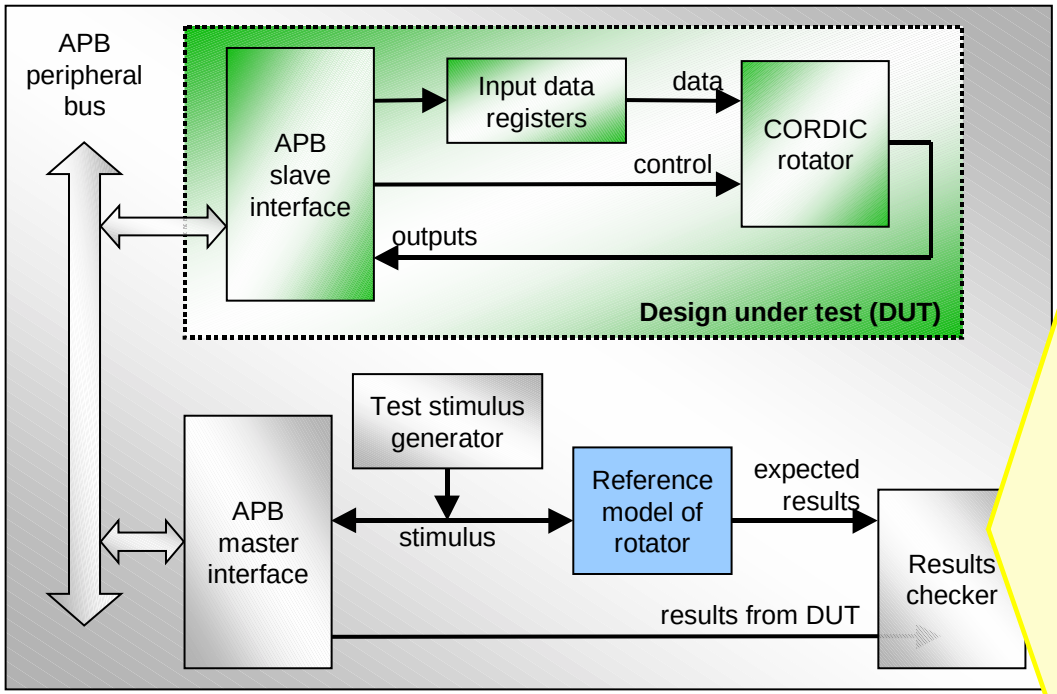
Sample Project Revisited

Jonathan Bromley
Michael Smith
DOULOS



Easier C Interface

Jonathan Bromley
Michael Smith
DOULOS




```
extern "C" void
  vlog_CORDIC_model (
    input  int reduceNotRotate,
    input  int angleIn,
    input  int xIn,
    input  int yIn,
    output int angleOut,
    output int xOut,
    output int yOut
  );
```

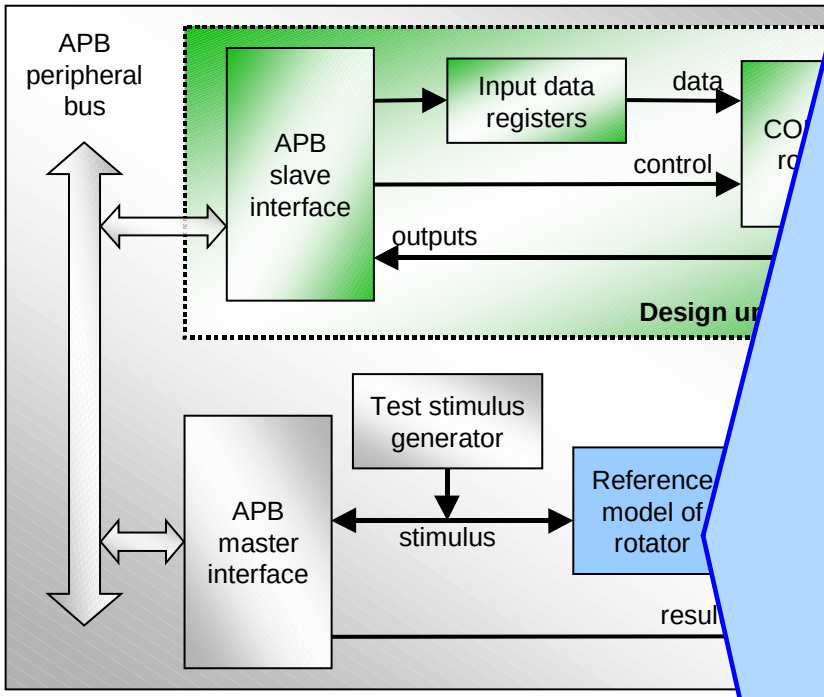
```
// Evaluate expected result
vlog_CORDIC_model(
  reduce, angle, x, y,
  exp_angle, exp_x, exp_y
);
```

This is VCS 7.1's "DirectC" - SystemVerilog3.1 has similar DPI

No PLI linkage tables!

Easier C Interface

Jonathan Bromley
Michael Smith
DOULOS



C helper functions

```
#define scalar_0 0
#define scalar_1 1
#define scalar_z 2
#define scalar_x 3
```

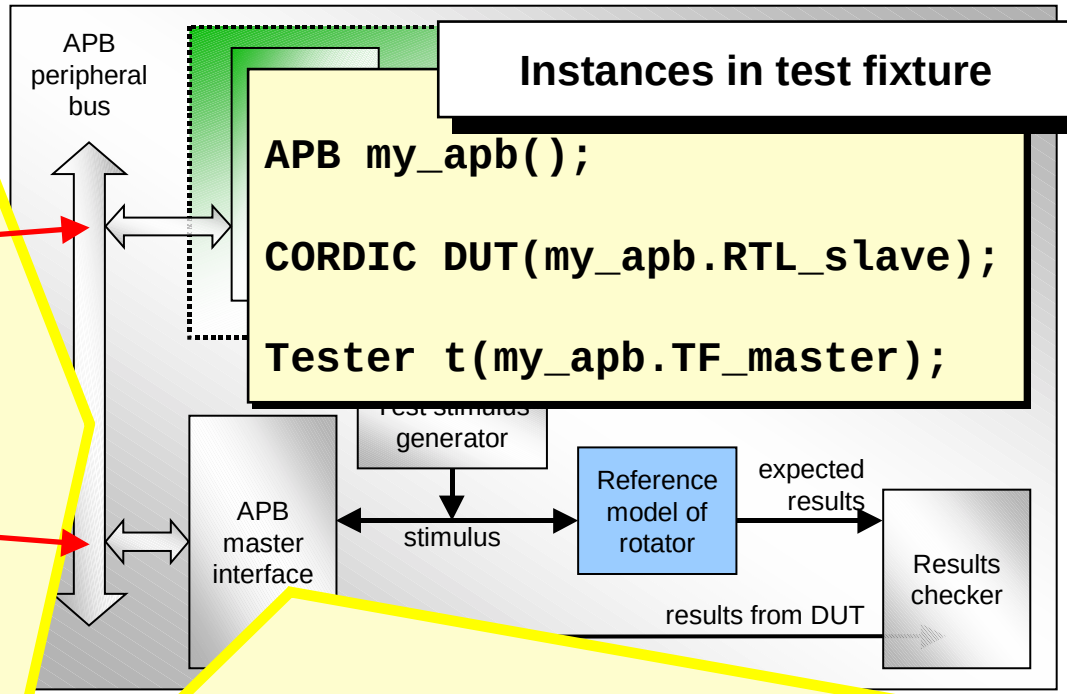
```
void vlog_CORDIC_model (
  scalar reduceNotRotate,
  int  angleIn,
  int  xIn,
  int  yIn,
  int* angleOut,
  int* xOut,
  int* yOut
) {

  double a, x, y, xR, yR, s, c;

  // Get trig values from Verilog info
  x = verilog_to_trig(xIn);
  y = verilog_to_trig(yIn);

  if (reduceNotRotate == scalar_1 ) {
    // Determine rotation to reduce y to 0
    a = -atan2(y, x);
```

**Plain C code -
no need for PLI calls to
get argument values!**

Interfaces and Modports

Jonathan Bromley
Michael Smith
DOULOS

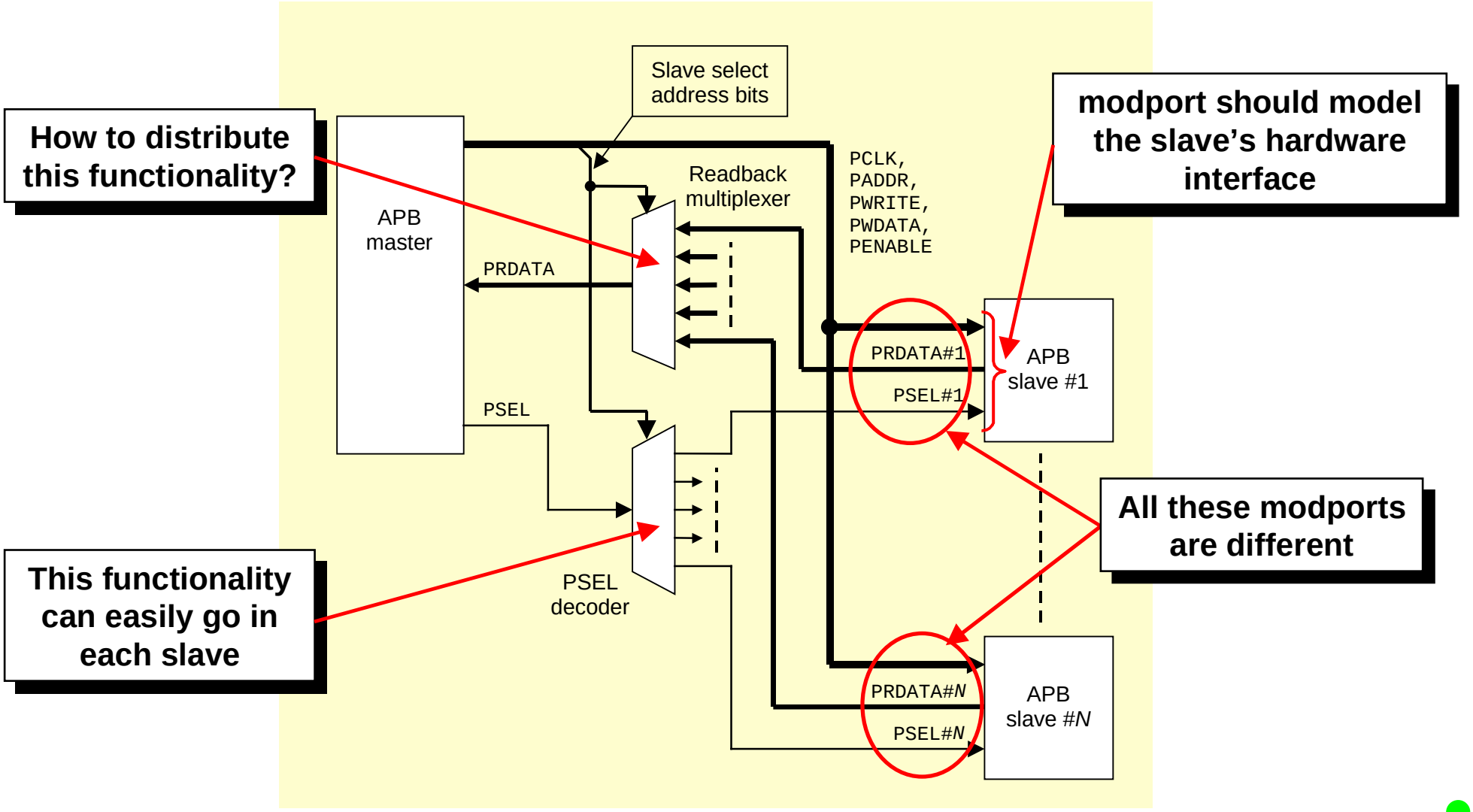
```
interface APB;

  logic PCLK, PSEL, ...

  modport RTL_slave (
    input PCLK,
    input PSEL, ...
  );

  modport TF_master (
    import task read,
    import task write
  );

  task read (
    input  [15:0] adrs,
    output [15:0] data )
    ...
  endtask

endinterface
```




```
module Tester(APB.TF_master Bus);
  initial begin : TestSequence
    logic [15:0] status;
    ...
    Bus.read(16'hFFE3, status);
    ...
  end
```

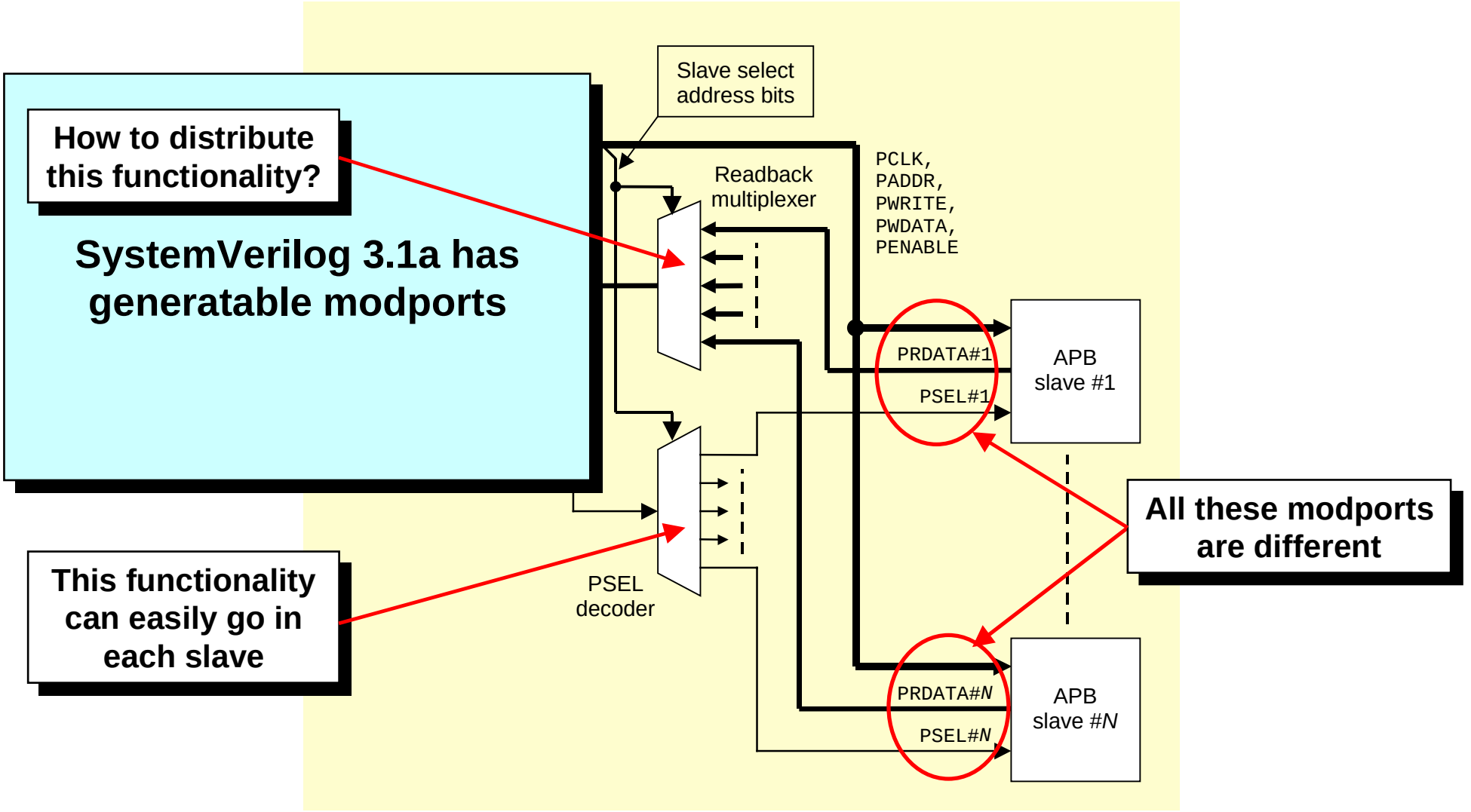
Interfaces, Modports and APB

Jonathan Bromley
Michael Smith
DOULOS



Interfaces, Modports and APB

Jonathan Bromley
Michael Smith
DOULOS



Conclusions

Jonathan Bromley
Michael Smith
DOULOS

- SystemVerilog 3.1 is useful now…
  - incremental improvements such as `.*`
  - `typedef`, `struct`, `union`, `packed`
  - **`$bits()`**
  - simple interfaces **Synthesis?**
- Constrained-random stimulus is important…
  - not investigated in this example
- We look forward to the full functionality of interfaces **Synthesis?**
- DirectC is good, DPI will give same benefits AND standardisation
- Assertions are powerful and convenient
  - but we need SV3.1a `cover`, `coverpoint` to match capabilities of specialised testbench automation tools