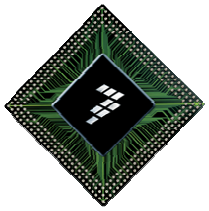
October 11, 2007

Pin Muxing Verification using Formal Analysis

CDNLive 2007 (Bangalore)



Sudhanshu Gupta, Neeraj Mangla, Nipun Mahajan

Freescale Semiconductor

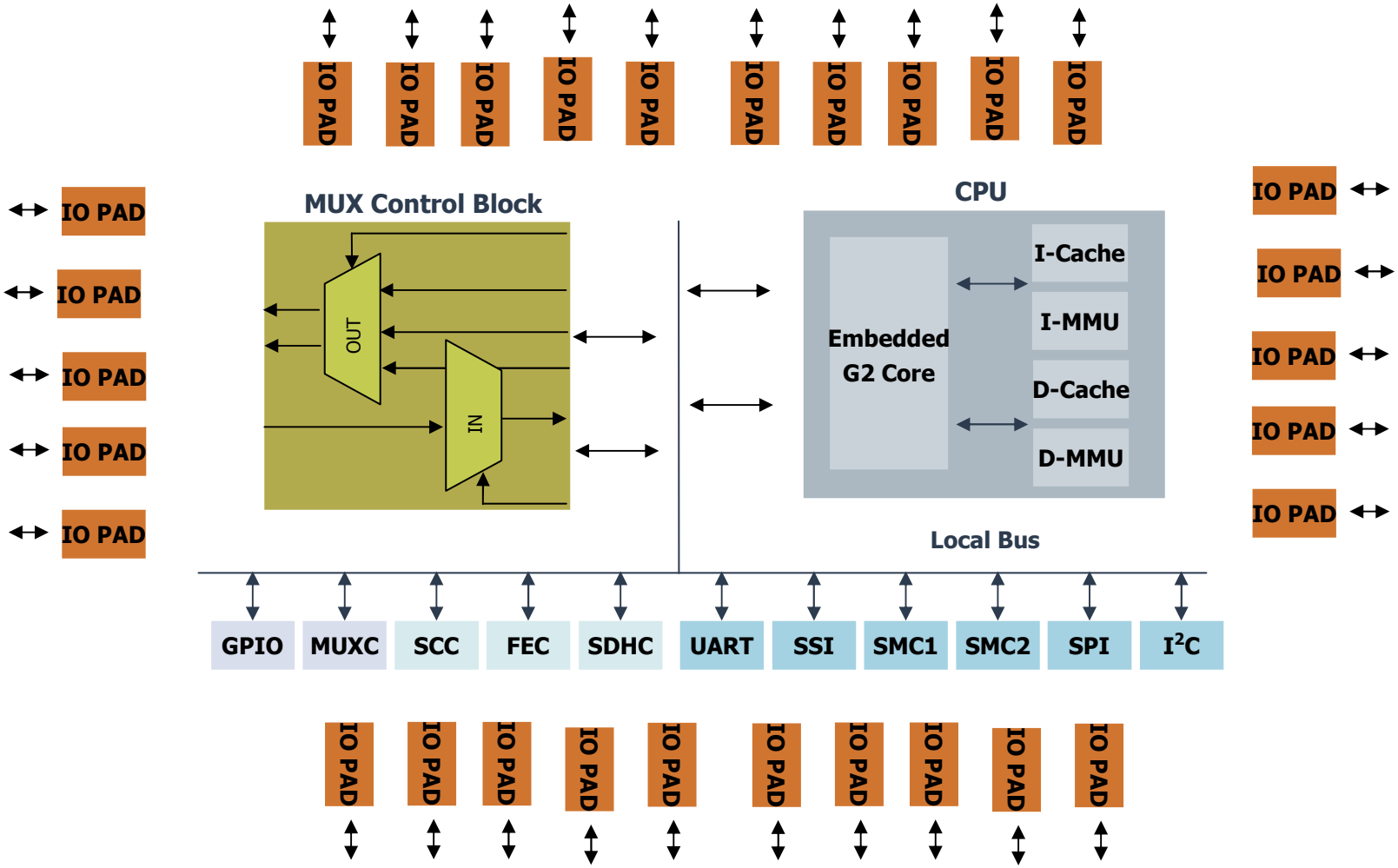
freescale™ semiconductor

Agenda

- IO Mux structure and verification complexity
- Existing approaches
- Limitations/Drawbacks with existing approaches
- Freescale Approach
- Single touch flow
  - Flow Implementation and details
  - Advantages
- Results
- Future enhancements and guidelines

freescale™ semiconductor

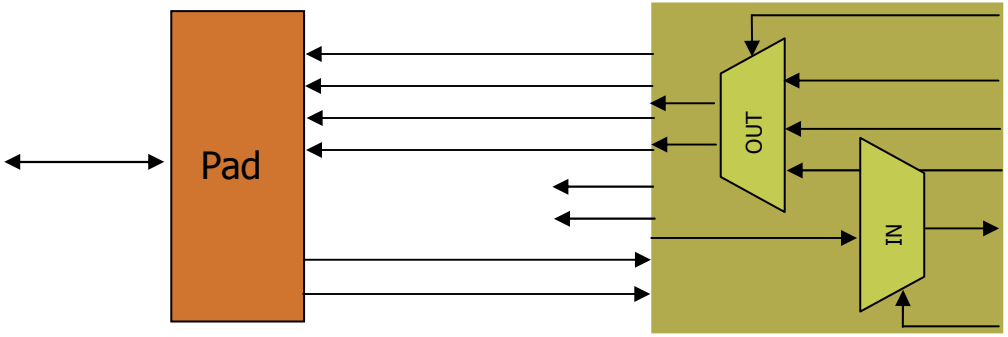
SOC IOMux Structure



freescale semiconductor

IOMux structure cont…

- Multiple pad type are present to meet different voltage and current requirements
  - Typically 4-5 types of pad are used
  - Each pad type has different control signals
    eg. sdr pads are slew controlled, so have slew rate control signals as input
  - All control signals are driven by Multiplexer logic
  - Separate multiplexer present for each pad
  - Multiplexer logic drives all control signal
  - Signals not required by pad type are left unconnected

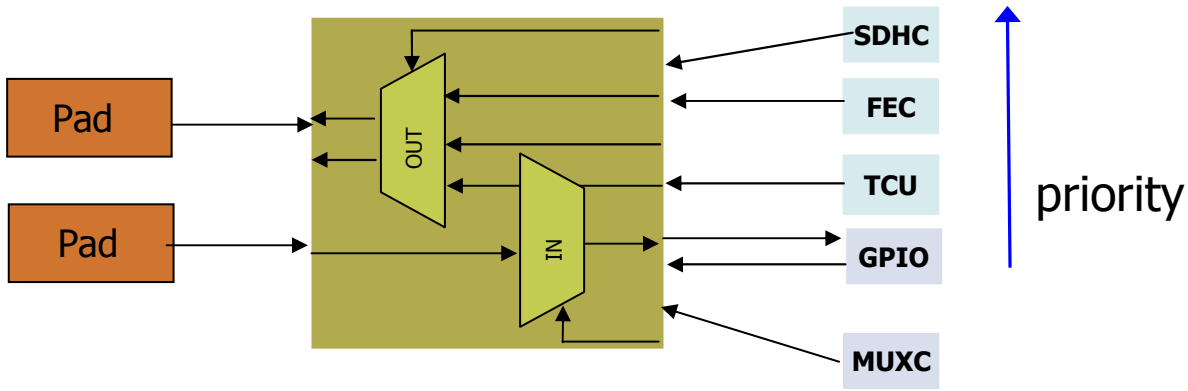


freescale™ semiconductor

IOMux structure cont…

- Multiplexer logic takes input from different modules available on the SOC
- DFT signals are also muxed in the same structure

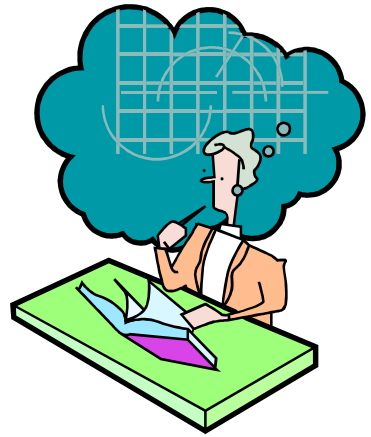
- Control signals are provided by a control block
- Multiplexer logic implements a priority muxing scheme
  - If port enable of two signals are active at the same time then the higher priority signals will be driven to the pad
- If multiple pads are trying to drive a module, a separate multiplexer logic sends the highest priority pad signal to the module



freescale™ semiconductor

Verification Complexity

- Huge coverage area
  - 5000+ nets
- DFT signals multiplexed on same Pins
- Covering all nets in functional mode not possible
  - Power related connections not covered
  - Drive strength and slew rate signals not covered
  - Pull up enable and pull up selects not covered
  - DFT multiplexing not covered
- Tedious job to write and run all testcases manually
  - 1000+ testcases to cover all scenarios
  - High quality testbench required
  - High Run and debug time
- 100% Coverage is not possible



freescale™ semiconductor

Traditional Methods in Use

- Dynamic simulation with directed testcases
  - Separate testcase to generate stimulus for each check
  - Separate Verilog checker to check connectivity
- Dynamic simulation with assertions
  - Assertions generated by automated script
  - Still require complete set of testcase to stimulate all assertions
- Random simulation with assertions
  - Still require some ways to check for the coverage of assertions

freescale™ semiconductor

Flaws in Traditional Flow

- Complete testbench setup is required
- Scan chain verification is not possible till a netlist is available
- None of the approach is single handed
- No prechecks (such as deadcode and FSM) are available
- Low confidence on assertion completeness

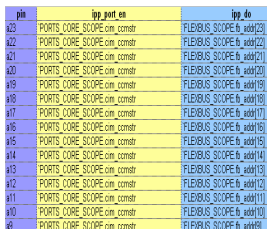
freescale™ semiconductor

Freescale Approach

- **Objective**
  - Remove stimulus and testbench generation effort
  - To have 100% coverage
  - To exploit formal tool for checking connectivity regressively
- **Inputs**
  - Connectivity information of IOMux in easily readable format i.e xls
  - Design RTL
  - Assertion Modules
- **Outputs**
  - Assertion set for all pads
    - Generated using xls sheet with the help of scripts
  - Separate batch file for each pad
    - This helps in running all pads assertion in parallel
  - Command file to fire the jobs on lsf
- **Results**
  - Script gives consolidated report about all the assertions
  - Command file to fire failed assertions

freescale™ semiconductor

Single touch Flow



Connectivity Info

```
module assert_fc;
begin
...
endmodule
```

Assertion Modules

```
module top;
begin
...
endmodule
```

RTL Design Modules

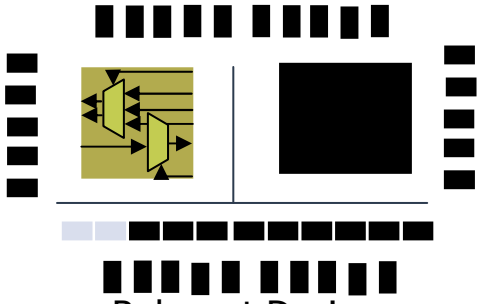
IOMUX_GEN

```
source setup.tcl
add_property_a1
prove
```

Pad Batch script

```
assert_fc pad_a1(
.ipp_do_vc(ipp_do_fc),
.ipp_obe( )'
);
```

Assertion module instantiation



Relevant Design

freescale™ semiconductor

IOMUX_GEN cont..

- Assertion Modules
  - Separate assertion module is written for each pad type
  - Models multiplexer's behavior using system verilog assertions

Inputs

- All the signals that are multiplexed on that pad
- Control signal
- Signals input to the pad
- Power signals

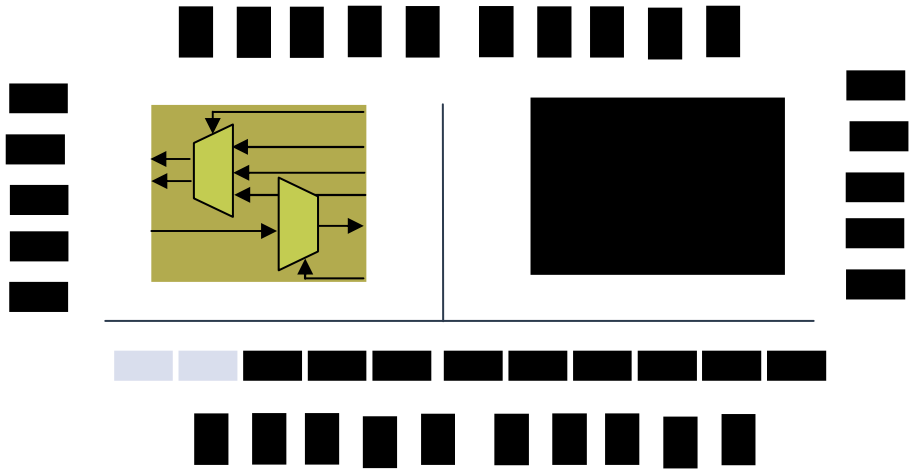
Checks

- Under all situations connectivity of all signals from module to pad is correct and vice versa
- All the Checks are performed from the module periphery
- Also checks for module integration

Stubbing the Design

- Connectivity checks are performed on the path
  - Modules output to the pad input
  - Pad output to the modules input
  - Control block to the multiplexer module
  - Power signals checks from pad output to the pad input

- Irrelevant logic is removed by IOMUX_GEN, resulting in design with
  - Reduced database size
  - Easy to handle big soc on Formal tools like IFV
  - Redundant logic cut off
  - Run Time reduced
  - Debug time reduced



freescale™ semiconductor

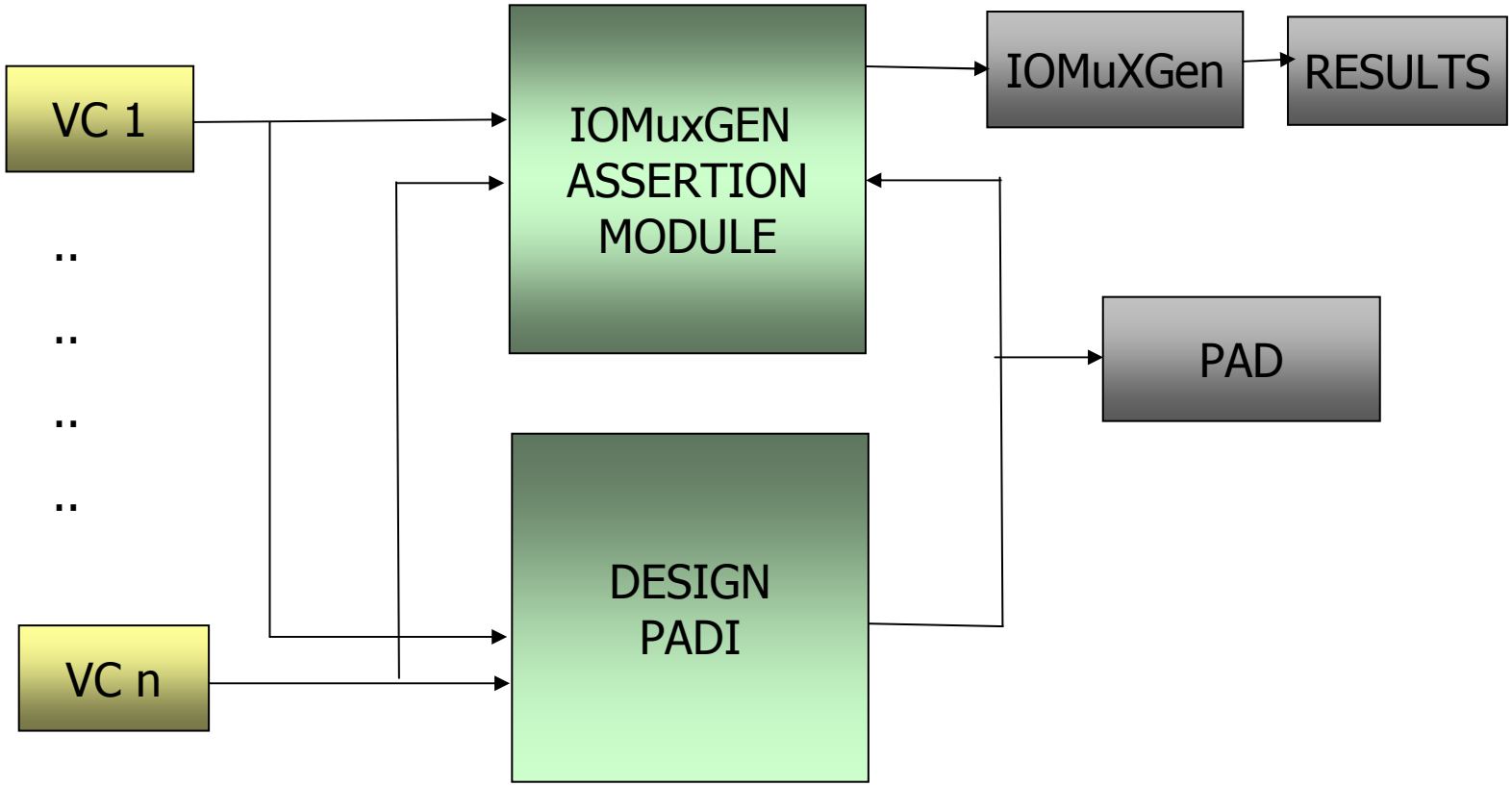
Sample assertion Module

```
module pad_sr_unit_internal(
  input clk,
  input w_priority,
  input ipp_do,
   input ipp_do_pad
  );

a_ipp_do: assert property
  (@(posedge clk)

  w_priority  |-> (ipp_do      ==
     ipp_do_pad ));

c_priority: cover  property
     (@(posedge clk)  w_priority);

endmodule

module pad_sr_unit
  #(parameter integer N = 1) (
  input           clk,
  input [N:1]   ipp_port_en,
  input [N:1]   ipp_do,
  input          ipp_do_pad,
 );
```

```
wire [N:0] highest_priority;
wire [N:1] w_priority;

assign highest_priority[0] = 1'b0;

genvar i;
 generate
  for (i=1; i<=N; i++)
   begin : g_

assign highest_priority[i] = highest_priority[i-1]
                                || ipp_port_en[i];

assign w_priority[i] = ! highest_priority[i-1] &&
                              ipp_port_en[i];

      pad_sr_unit_internal im(
        clk,
        w_priority[i],
        ipp_do[i],
        ipp_do_pad,
         );
   end
 endgenerate
endmodule
```


freescale™ semiconductor

Instantiating Assertion modules



VC – Virtual Component

freescale™ semiconductor

Running IFV

- Details of constraints
- Init files
- Scripts
- Pad specific constraints
- Reporting Results

freescale™ semiconductor

Running IFV

- Constraints
  - Resets
  - Defining cutpoints
  - Any SOC specific constraint

Example Constraint :

```
constraint -add -pin testbench.top.pll_ipg_soft_async_reset  = 0
```

- Init files
  - ifv_execute.tcl :Defines parameters like effort, engine, witness
  - reset.tcl : Defines all the resets of design
  - clock.tcl : Specifies the clocks in the design
  - write_cex.tcl : Dumps the waveforms for the failed assertions

Running IFV

- Pad specific Constraints
  - Constraints specific to a particular pad
  - Completely new feature implemented

  Example constraint

```
proc add_const_u2txd {}
                    {   cutpoint -add testbench.top.act_lo
                        const -add -pin testbench.top.act_lo=1'b0
                    }
```

- Steps to run IOMux Gen

  - **Padi.pl**: Takes in xls and generates assertion module instantiations, assertions and batch file for each pad
  - **Run_ifv**: Runs IFV picking up batch file for pad specified
  - **Results.pl**: Gives a consolidated report of passed and failed assertions

Examples Contd..

- Example Module Instantiation

```
pad_sr_unit #(2) pad_sr_bind_u2txd
                ( // clock
                    `ASSERTION_CLOCK,
                  // ipp_port_en
                    {  `PORTS_CORE_SCOPE.par_u2_txd
                       `PADRING_SCOPE.pad_test.ipp_ind
                    },
                  // ipp_do
                    {  `UART2_SCOPE.TXD,
                       1'b0,
                    },
                // pad - ipp_port_en, ipp_do,
                    `PAD_U2TXD.ipp_do,
                );
```

- Example Generated Property

```
proc add_properties_u2txd {} {
                assertion -add longjing.pad_sr_bind_u2txd.g_[2].im.a_ipp_do
                assertion -add longjing.pad_sr_bind_u2txd.g_[1].im.a_ipp_do
                                }
```

Examples Contd…

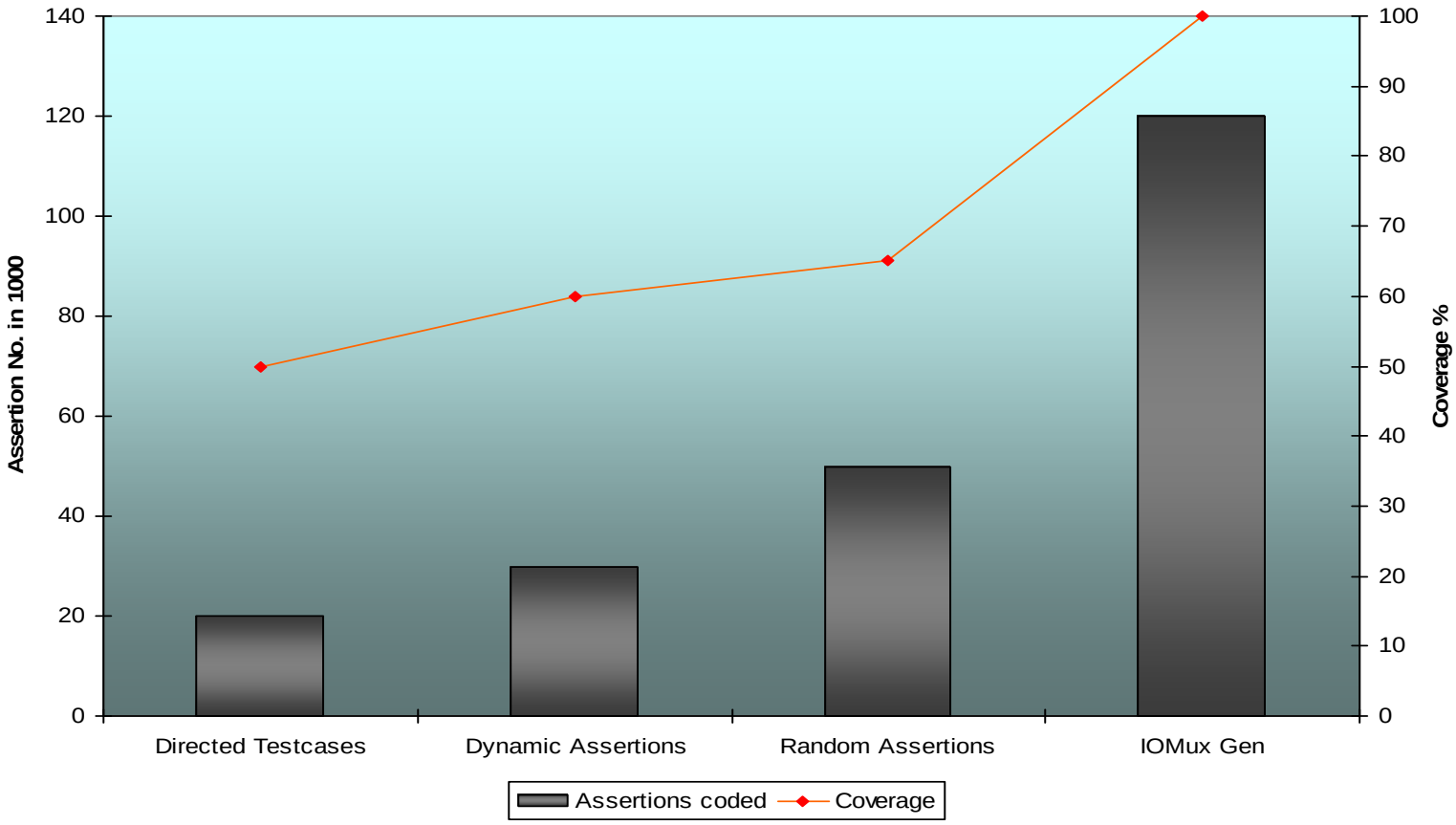
- Example Batch file

```
input ifv_startup.tcl
add_properties_u2txd
add_const_u2txd
input ifv_execute.tcl
```

- Example Results Summary

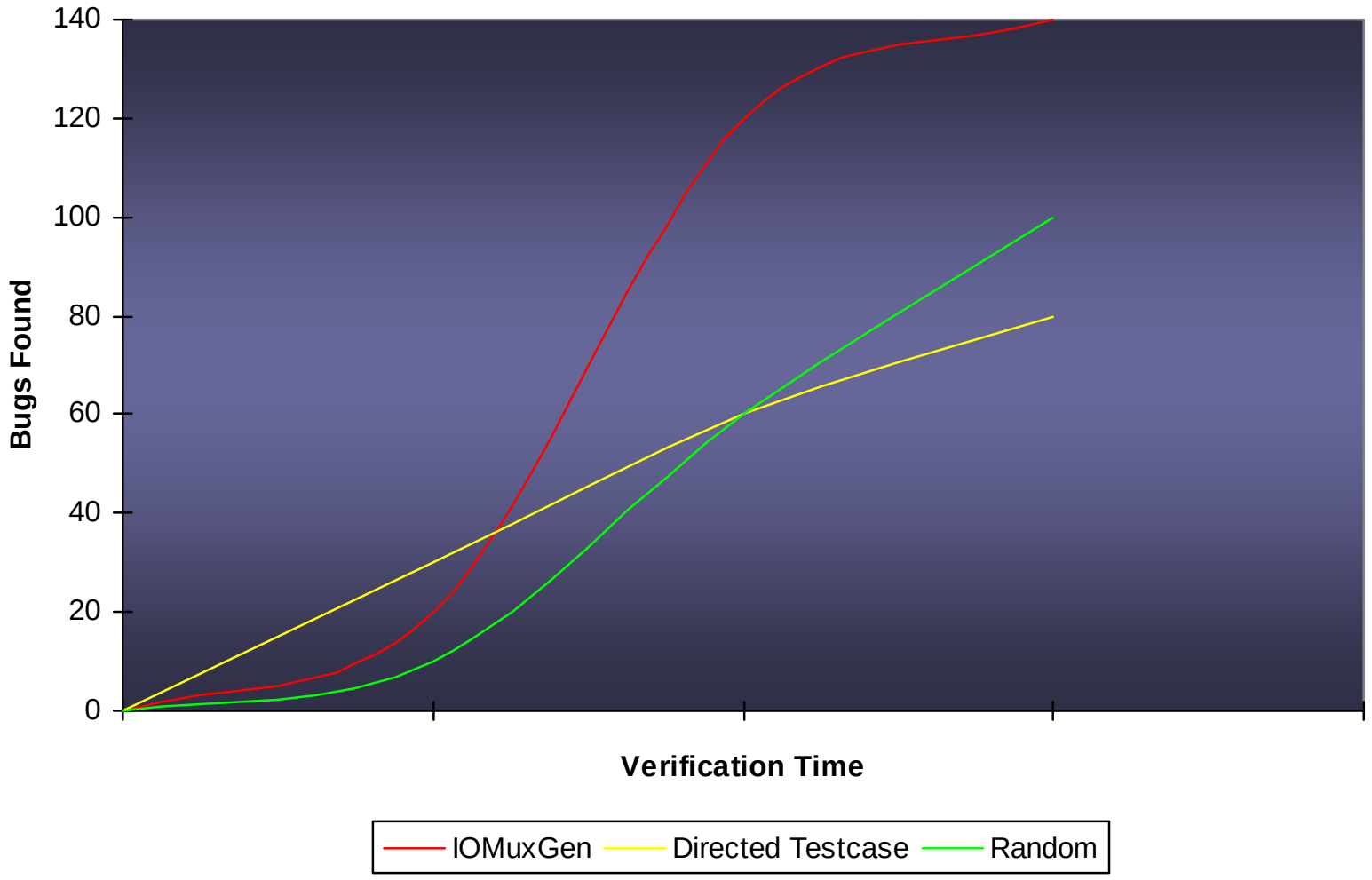
```
Summary
    Total pads:          174
    Total logfiles:      174
    Incomplete logfiles: 0
    Missing logfiles:    0

    Total properties expected: 7076
    Total properties actual:   5434
    Total properties passing:  5434
    Total properties failing:  0

    Total CPU time:    119038 sec (1984 min) (33 hr)
    Total real time:   193763 sec (3229 min) (54 hr)
    Average CPU time:  684 sec (11 min) / pad;  22 sec / property
    Average real time: 1114 sec (19 min) / pad;  36 sec / property
```

Comparison Of Results



freescale™ semiconductor

Efficiency



freescale™ semiconductor

Future Enhancements and Guidelines

- Future Enhancements
  - Automate the process of generation of Excel sheet
  - Extending it to other modules and complete SOC
  - Generation of testcases

- Guidelines
  - Standardization of IP level signal names
  - Delivery of top level signal names by each IP owner
  - Pad must have same name as top level pin

freescale™ semiconductor


freescale™
semiconductor