Maxtestbench - A technology to manage STIL pattern qualification in complex SoC Verification environment

Guillaume Costrel de Corainville / Mickael Broutin

ST Microelectronics

June 18, 2015

SNUG France

Agenda

Introduction

Environment and constraints

Stil2verilog flow to manage functional patterns

Achievement and Conclusion

Introduction

Introduction

- Two major categories can be defined in “test world”:
  - Structural test (SCAN methodology)
  - Functional test (functionalities verification)

- The aim of this presentation is to provide a possible flow convergence path between these two categories
  - To rationalize tooling
  - To reduce workload without pattern quality loss

Introduction

- Functional test overview:
  - IEEE1149.1 and IEEE1500 accesses

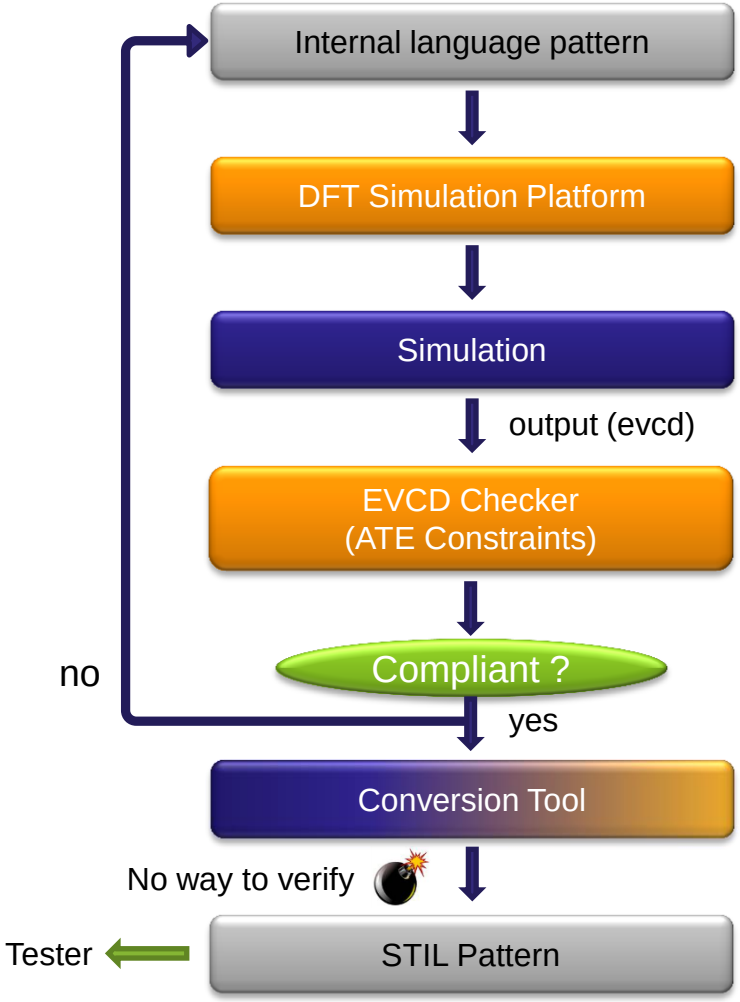
Functional test	Category	Case Nb = 300
Tap and 1500 controller	Direct access	21
Boundary scan	Direct access	6
IO compensation cell	Direct access	4
Process Monitoring Box	Direct access	12
Thermal Sensor	BIST/direct access	1
Memory	BIST	84
Memory repair	direct sequence	42
Memory Bitmapping	BIST	3
Memory Retention	BIST	16
PLL	BIST/direct access	30
DPHY	BIST/direct access	22
USB2PHY	BIST/direct access	17
MiPHY (PCIe)	BIST	26
LPDDRPHY	BIST	9
eFUSE	BIST/direct access	7
RNG	BIST/direct access	4
Power Controller	Direct access	2
OCC	Direct access	10

- Design overview:
  - Area of 85mm²
  - 4 digital voltage domains
  - 3700k DFFs
  - 1200 memories
  - Various PHYs
  - 200 implemented silicon test patterns

Environment and constraints

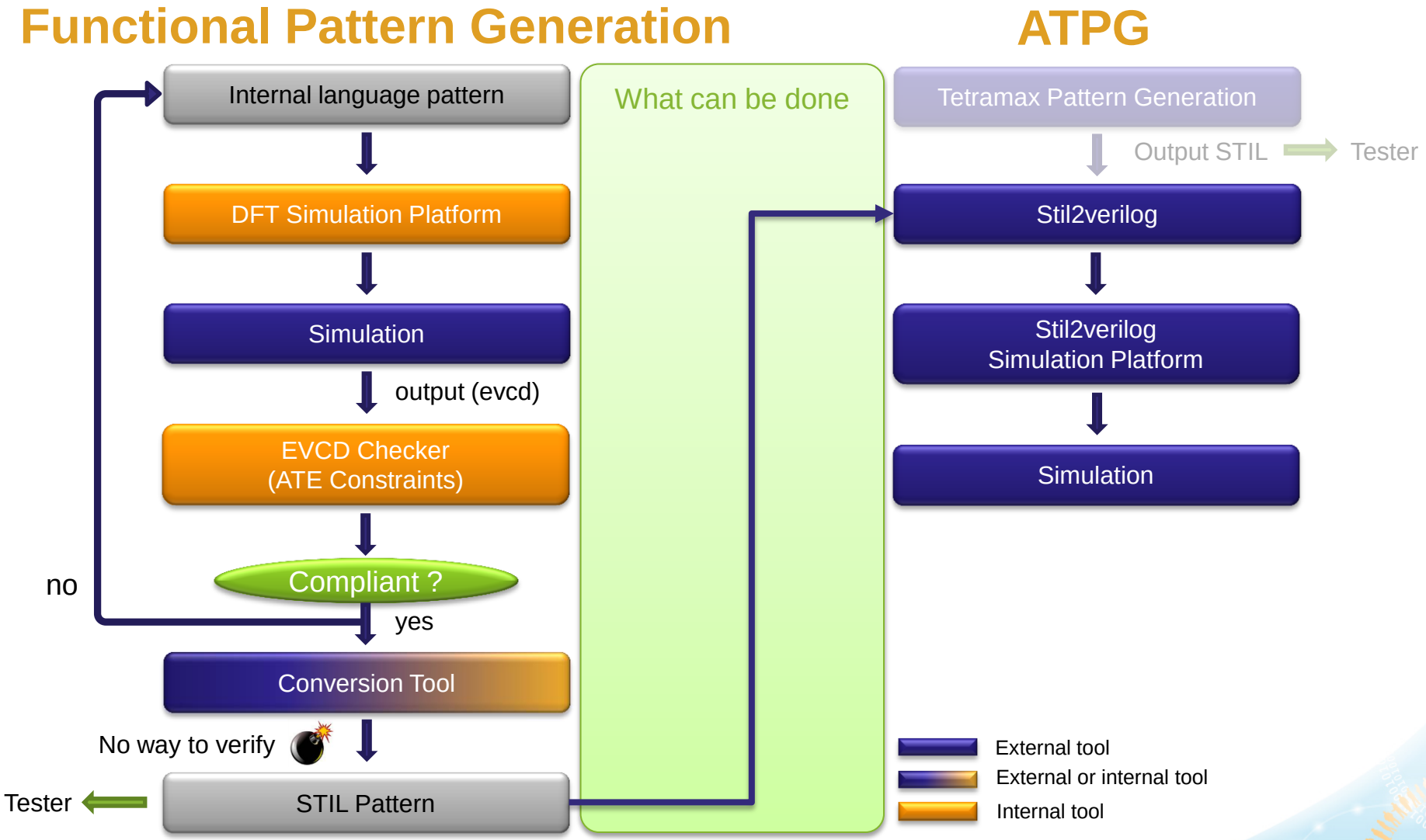
Environment

Functional Pattern Generation

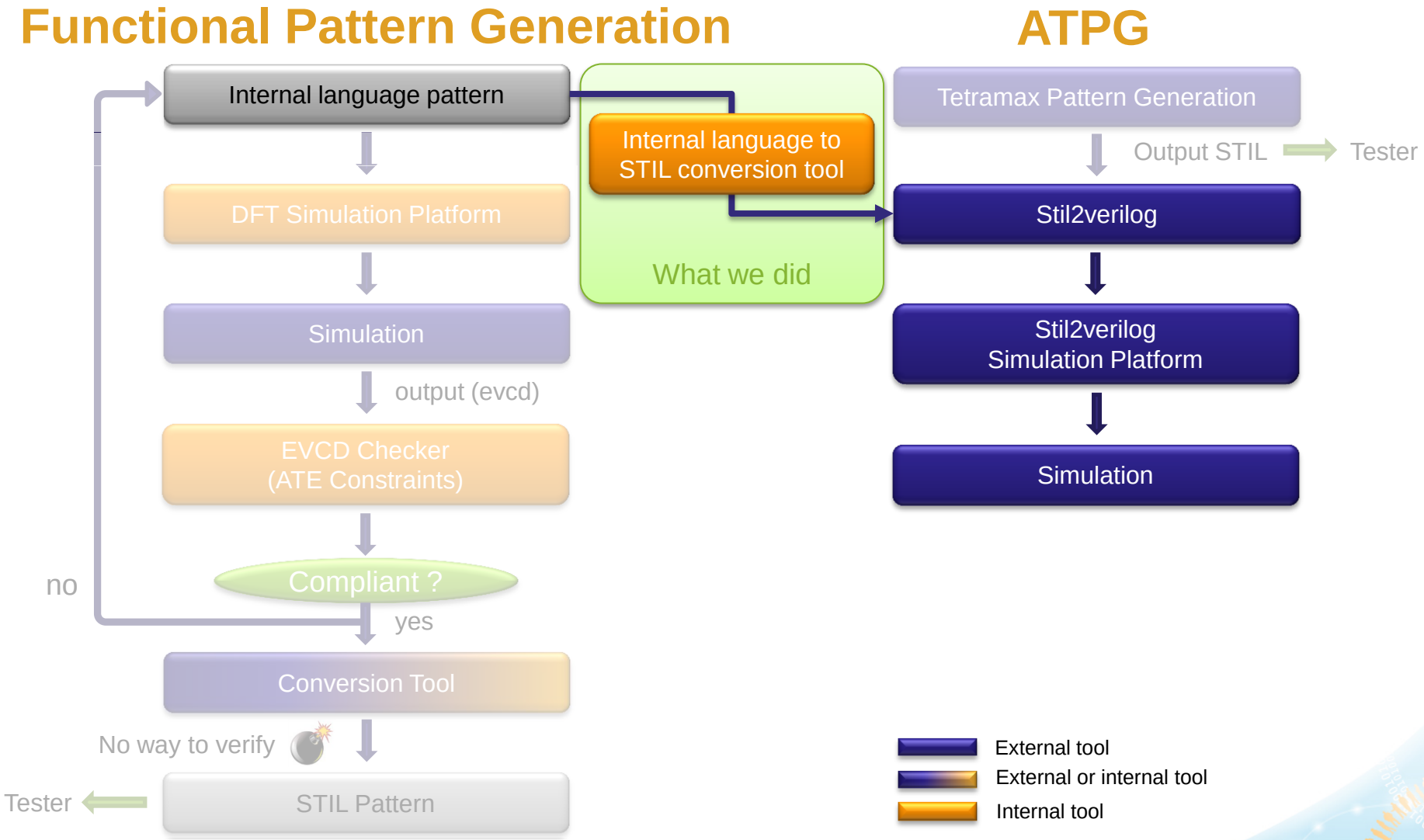


ATPG

Environment



Environment



Why was this flow used

Synopsys Users Group

pros

- No simulation artefact
- Quick STIL creation and validation

cons

- cycle based language
- conversion tool

Constraints

- Design data
  - Design elaboration run time ~5 hours
  - Elaboration snapshot ~80 Gb
  - 200 functional test pattern
- need to minimize as much as possible
  - Disk space usage
  - Regression run time
  - ➢ A single testbench must be created, avoiding specific elaboration per pattern to:
    - Save 1000 Hours of elaboration runtime
    - Save 16Tb disk space

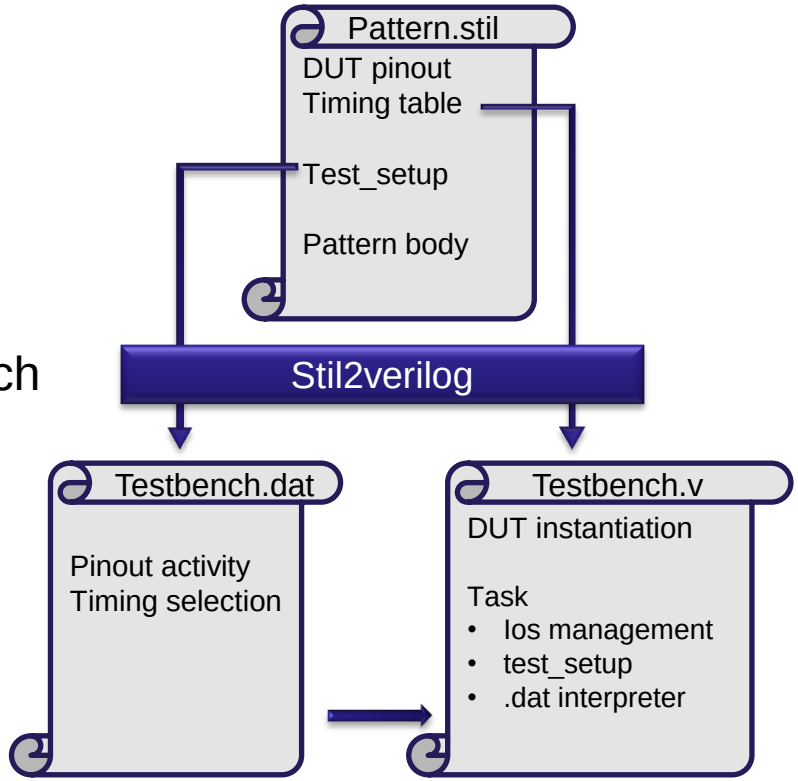
Stil2verilog flow to manage functional patterns

Stil2verilog way of working

- How it works ?
  - STIL pattern is used as input
  - Two output files are generated

(with the command *stil2verilog pattern.stil testbench –replace*)

- Verilog testbench
- .dat sequence file read by the testbench



Stil2verilog way of working

- How to manage functional pattern with stil2verilog?

- 2 possible approaches
  - Use STIL ATPG from Tmax and replace “Test setup” by functional test pattern
    - Functional pattern content is embedded in “Test setup”
  - Use directly functional STIL pattern and convert it
    - Functional pattern content is embedded in “pattern body”

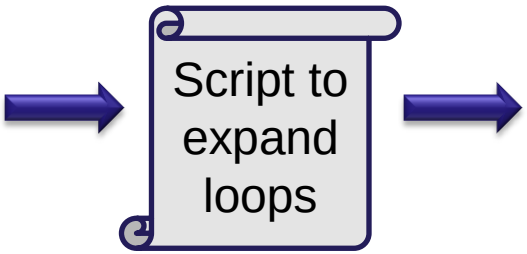
- Direct STIL conversion solution has been used
  - Advantage: Enable single testbench usage – “pattern body” is not hardcoded in testbench
  - Drawback: Out of box solution – Need to ensure Stil2Verilog usage with functional pattern

Updates for testbench genericity

- **Update 1**: Loops present in a pattern are hard coded in the generated testbench.v as tasks
  - **Impact**: Loop information is coded in the “testbench.v” instead of the “.dat” file
  - **Solution**: A script is run on the STIL pattern to replace the loops by the corresponding number of iterations of the vector looped.

Original STIL pattern

```
Loop 5{
      V{TAP_TCK=P; }//WAIT_CYCLE
}
```



Reworked STIL pattern

```
V{TAP_TCK=P;}//WAIT_CYCLE
V{TAP_TCK=P;}//WAIT_CYCLE
V{TAP_TCK=P;}//WAIT_CYCLE
V{TAP_TCK=P;}//WAIT_CYCLE
V{TAP_TCK=P;}//WAIT_CYCLE
```

Updates for testbench genericity

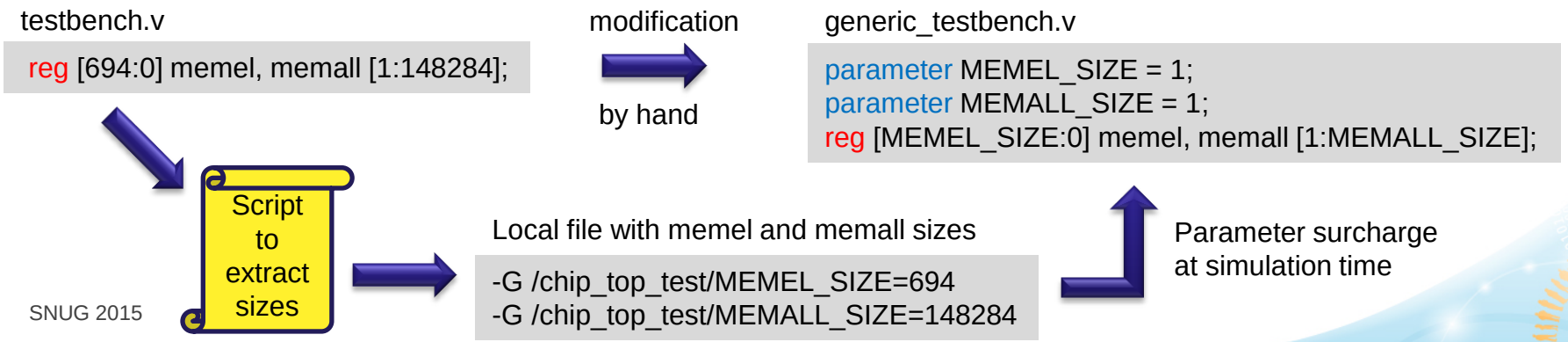
- **Update 2**: In some testcases, after translation to stil format, a check on outputs is done using _PO group, but not in all
  - **Impact**: The generated testbenches have not the same signal list
  - **Solution**: To generate a common signal list in all testcase testbenches, a dummy test on _PO is added at the end of each testcase.

Reworked STIL pattern

```
//dummy check to align all testbench signals
                V{_PO=XXXXXXXXXXXXXXX … XXXXXXXXXXXXXX;}
```

Updates for testbench genericity

- **Update 3**: memel and memall registers are hard coded in the generated testbenches, and are different for each testcase.
  - **Impact**: Parameter dependency of the ".dat" file content
  - **Solution**:
    - A script stores these registers' sizes in a local file for each testcase simulation directory.
    - The final generic testbench is modified with parameters replacing these registers' sizes
    - At simalation time, the parameters are surcharged.



Updates for testbench genericity

- **Update 4**: The TDATA file path is absolute
- **Impact**: It is impossible to run several simulations in parallel
- **Solution**: Update the TDATA FILE path from an absolute to a relative path. In our case, the update is

  `define TDATA_FILE "../testbench.dat"

Update for simulation

- **Update 5**: IOs management
  - **Impact**: IOs management is not fully covering functional pattern needs
  - **Solution**: For all the inout PADs, the task apply_XXX_default_WFT_WFT must be updated.

```
task apply_<YOUR_NAME>_default_\WFT_\WFT;
  reg [SIG_IDS_\W-1:0] sid;
  reg [SIG_IDS_\W-1:0] n;
  reg [SIG_IDS-1:0] s;
 begin
  for (sid=0; sid < SIG_IDS; sid=sid+1) begin
   n=0;
   case(sid)
    'd1, 'd2, 'd3, ..., GPIOxx
    begin
     if(ALLOUTSIGIDS[sid]==1'b1) begin
      for (n=0; n <SignalIDWidth[sid]; n=n+1) begin
       case ({TMPOUTSIGS_T[MAX_SIG\W*sid+n],TMPOUTSIGS_\[MAX_SIG\W*sid+n]})
        2'b10: begin   ALLINSIGS[MAX_SIG\W*sid+n] <= #(0) 1'bZ;   ALLOUTSIGS[MAX_SIG\W*sid+n] <= #(0) 1'bX;   ALLOUTSIGS[MAX_SIG\W*sid+n] <= #(30) 1'b0;  end
        2'b11: begin   ALLINSIGS[MAX_SIG\W*sid+n] <= #(0) 1'bZ;   ALLOUTSIGS[MAX_SIG\W*sid+n] <= #(0) 1'bX;   ALLOUTSIGS[MAX_SIG\W*sid+n] <= #(30) 1'b1;  end
        2'b1X: begin   ALLINSIGS[MAX_SIG\W*sid+n] <= #(0) 1'bZ;   ALLOUTSIGS[MAX_SIG\W*sid+n] <= #(0) 1'bX;   ALLOUTSIGS[MAX_SIG\W*sid+n] <= #(30) 1'bX;  end
        default: undef_wfc({TMPOUTSIGS_T[MAX_SIG\W*sid+n],TMPOUTSIGS_\[MAX_SIG\W*sid+n]}, sid, 0);
       endcase
      end
     end
     if(ALLINSIGIDS[sid]==1'b1) begin
      for (n=0; n <SignalIDWidth[sid]; n=n+1) begin
       case ({TMPINSIGS_T[MAX_SIG\W*sid+n],TMPINSIGS_\[MAX_SIG\W*sid+n]})
        2'b00: begin   ALLINSIGS[MAX_SIG\W*sid+n] <=   #(0) 1'b0;  end
        2'b01: begin   ALLINSIGS[MAX_SIG\W*sid+n] <=   #(0) 1'b1;  end
        2'b0Z: begin   ALLINSIGS[MAX_SIG\W*sid+n] <=   #(0) 1'bZ;  end
        2'bZ1: begin   ALLINSIGS[MAX_SIG\W*sid+n] <=   #(0) 1'b0;  ALLINSIGS[MAX_SIG\W*sid+n] <= #(33) 1'b1;  ALLINSIGS[MAX_SIG\W*sid+n] <= #(66) 1'b0;  end
        2'bZ0: begin   ALLINSIGS[MAX_SIG\W*sid+n] <=   #(0) 1'b0;  end
        default: undef_wfc({TMPINSIGS_T[MAX_SIG\W*sid+n],TMPINSIGS_\[MAX_SIG\W*sid+n]}, sid, 0);
       endcase
      end
     end
    end
```

Achievement and Conclusion

Achievement and Conclusion

- Even if not intended to, Maxtestbench can cover functional pattern verification

- This approach provides the following benefits:
  - Reduces test program ramp-up and resource debug effort by providing a Synopsys solution to qualify functional STIL pattern
  - Enables flow rationalization by reducing the number of steps and tools required for functional test pattern generation
  - Creates flow synergy between the functional and structural test worlds

Achievement and Conclusion

- What next…
  - Some minor workaround to manage testbench unicity
    - Some already obsolete in latest Stil2Verilog version
  - This flow could be used to create synergy inside SNPS tool suite
    - Stil2Verilog can be used for Tmax, SHS and SMS technology
  - One step is not standardized – “internal pattern language”
    - Not efficient pattern reuse format between groups/division of STMicroelectronics and external provider/customer
    - Looking for standardized approach… IJTAG (IEEE1687) could be an answer via Procedural Description Language (PDL)…

life.augmented

snug
Synopsys Users Group

Thank You