Front-End DFT

Rayi Giri Varshini

BTech 3rd Year Undergrad
IC Design & Technology

IIT Hyderabad

Agenda

1. C090 Common Capture Flop
2. P18 ATPG & Simulation
3. ATPG Chain Test
4. Pipeline Flop Insertion
5. Mismatches in Serial Sims
6. DFT/ATPG Issues
7. AI in VLSI
8. Projects at IITH

C090A REGBANK Common Capture Flop Modification

- Objective: To modify the RTL of Register Bank to have a common capture flop without change in update flops.
- In REGBANK, each register when addressed via the testbench, goes through capture and update flops.
- In Serial/JTAG mode (00), data is travelled as one bit at a time and in parallel (01) data, travels in chunks of 16 bits.
- Tools used: xceliumagile, synthesis

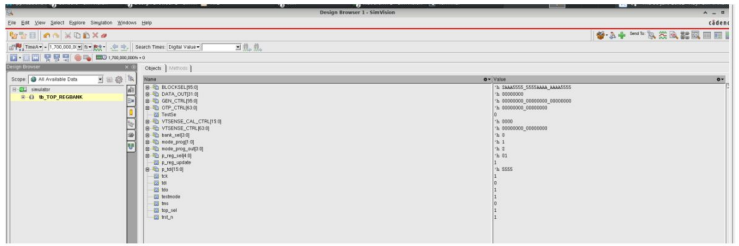
Register	Length	Cycles in Mode 01
BLOCKSEL	96	6+1
VTSENSE_CAL_CTRL	16	1+1
VTSENSE_CTRL	64	4+1
OTP_CTRL	64	4+1
GEN_CTRL	96	6+1

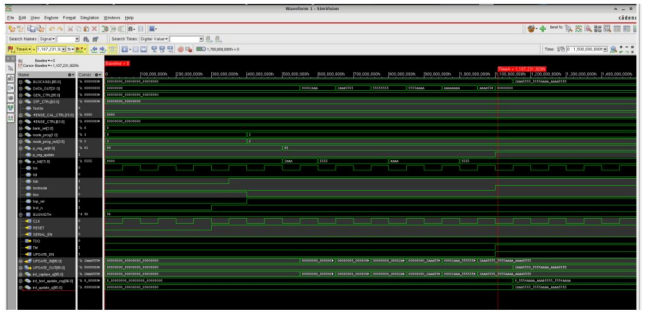
Plan and Modifications

- A common capture register of length 96 is required where each register will continue to have dedicated update registers.
- Separate modules capture & update are introduced, where capture is instantiated one time and update is instantiated 5 times. Effectively, 10 USER_FF modules reduce to 6 USER_FF.
- Instead of instantiating two clock gating cells in USER_FF, when only one CGT is used in USER_FF and adding the other CGT in capture and update modules.

Results for Simulation & Synthesis

For BLOCKSEL,





	REGBANK Unmodified	REGBANK with Common Capture Flop	REGBANK single CGT cell in USER_FF
Area	29884.983481 µm²	19833.005480 µm²	15153.466126 µm²
Power	8.1102e-02 mW	3.8494e-02 mW	2.3540e-02 mW
Timing (slack)	94.2 ns	92.78 ns	93.45 ns

Error Correction Codes (ECCs)

The techniques used to detect and correct errors in digital data essential for ensuring data integrity in communication systems, storage devices and memory systems.

REQUIREMENT: In REGBANK, as a common capture is added for all registers, if any error has occurred it would propagate leading to errors in the other registers too. So, we expect the ECC to correct single bit errors, correct double bit errors, detect triple bit errors.

Why ECCs?

- Whenever data is transmitted or stored, it maybe possible that data may become corrupted. This can take form of it flips (1 to 0 or 0 to 1). Error correction codes seek to find when an error is introduced into some data by adding parity/redundant bits info to data.
- Detects and corrects the errors so that time is not lost when some program is run outside working hours using cronjobs for say.
- As a common capture is present, one error could make all the remaining registers go wrong. Though we reset after every register the risk of error propagation increases without ECCs.
- Navigate the TAP (Timing, Area, Power) for the design without ECC in common capture and with it including the ECC and check which is better.

Hamming Code and BCH

•**Standard Hamming Code:** It is a popular ECC that can detect and correct single bit errors. It uses parity/redundant bits and XOR logic to detect and correct single bit error/data flip, enabling forward error correction, where errors are automatically corrected when read back.

•**Extended Hamming Code:** Additional overall parity bit is added to standard code to reliably detect double bit error resulting in single error correction/double error detection (SECDED).

•**Bose-Chaudhari-Hocquenghem (BCH) Code:** A class of cyclic advanced ECCs capable of correcting multiple errors within a block of data. Using reduction modulo, encoding, decoding, Galois field for single error correction/double error correction/triple error detection (SECDECTED).

- Can’t we further extend hamming code for triple bit error detection than going for BCH and other techniques?
  Yes, we can, but it becomes an inefficient way due to much increased logic.

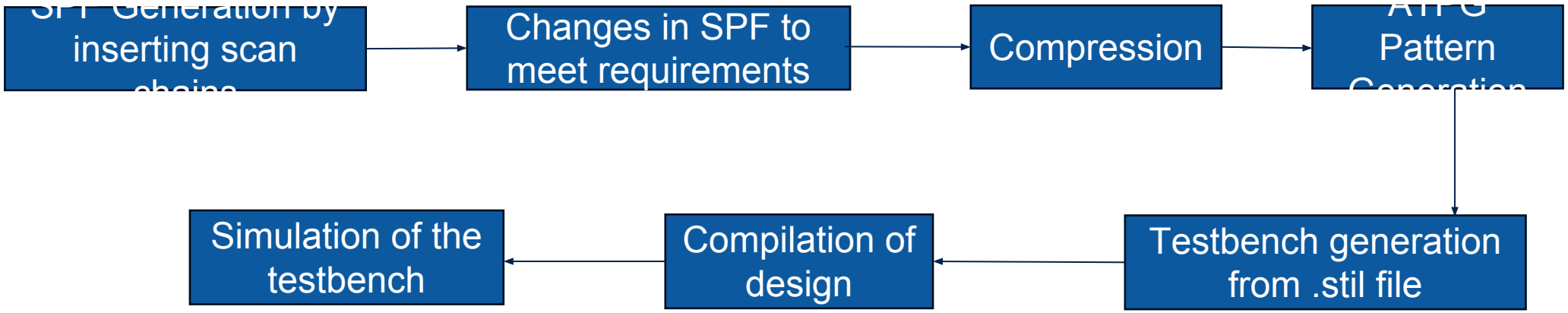
Overview on PROLIBP18 V6.0

- **Low Power Blocks:** MEMBLOCK1_SPHD_LOLEAK & MEMBLOCK1_SPREG_LOLEAK (0.5 V)
- **Modes:** LOGIC, LOGIC_MEM, LOGIC_COMP, LOGIC_MEM_COMP, ONLY_MEM, REGBANK
- **Fault types:** Stuck-at, Low-Speed, At-Speed
- **Pattern Modes:** Parallel (par), Serial (ser)
- **Process Corners:** Zero Delay, SDF (FF & SS)

- Tools used:
  tetramax for ATPG pattern generation
  xceliumagile for Simulation of testbench

ATPG Pattern Generation & Simulation

- Top Level ATPG Pattern Generation and Simulation for Zero Delay and SDF (FF & SS) for SPHD and SPREG for different modes, faults and pattern modes in P18.



Modifying BLOCKSEL[5:0], ADD[16], ADD[17], ADD[18], ADD[21], CTRL[0], CTRL[1] for the blocks, modes, fault-types, process corners, pattern modes.

Results: Achieved 0 mismatches with maximized fault coverage where all constraints are met for 5 MHz.

Modifications in SPF for ATPG Simulations

- Mismatches due to Analog Signals, manual modifications in test set-up, finding the optimal strobing values, etc.
- **Solutions Implemented:**

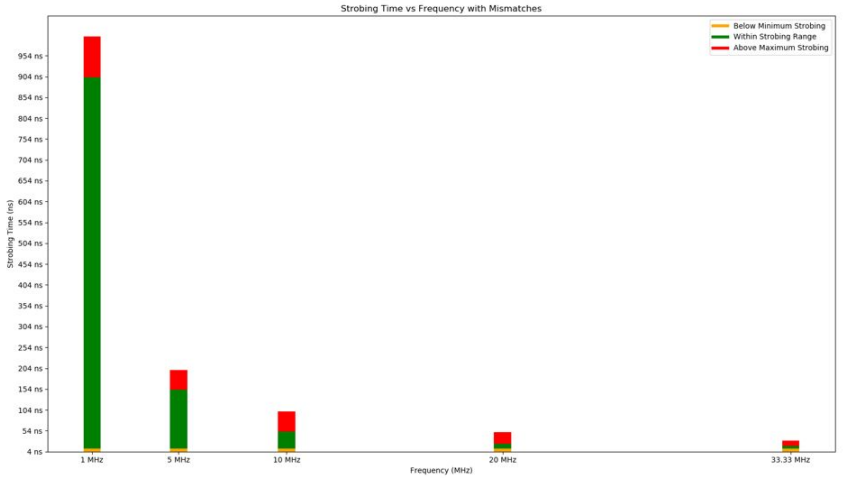
1. Removed the analog signals MEASRA_MB1 & MEASRA_MB2
2. Incremental flow is introduced to reduce the test time
3. Added two test vectors with clock turned off (dummy cycles) to meet setup/hold requirements
4. Vectors defining resets were added in the test-set up in Macrodefs
5. In high-trans, pre-shift cycle test-set up and PLL programming is added

ATPG Chain Test for maximum shift frequency

Objective:

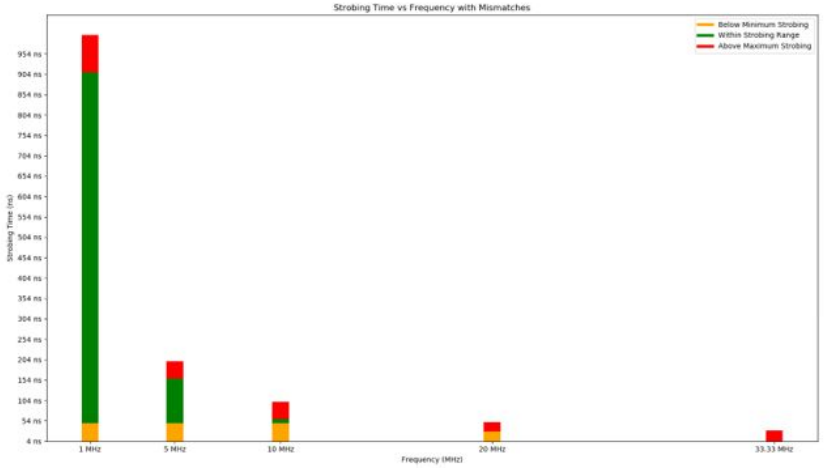
- To find the maximum shift frequency and strobing range for SPHD and SPREG low power blocks.
- Using python-based automation script

ATPG Command: run_atpg –only_chain_test



For FF corner

- Generate ATPG patterns for chain test then simulate for a given frequency, rise time, fall time, and strobing time.
- Compare and find the strobing range along with the maximum frequency of operation for the most optimum performance.



For SS corner

Observations in Chain Test

For FF trails,

Freq (MHz)	Time period, Tr, Tf (ns)	Strobing range - SPHD LOGIC_MEM_COMP (ns)	Strobing range - SPHD LOGIC_COMP (ns)	Strobing range - SPREG LOGIC_MEM_COMP (ns)
1	1000, 900, 930	12-902	10-902	11-902
5	200, 150, 180	12-152	10-152	11-152
10	100, 50, 80	12-52	10-52	11-52
20	50, 20, 40	12-22	10-22	11-22
33.33	30, 15, 25	12-17	10-17	11-17

For SS trails,

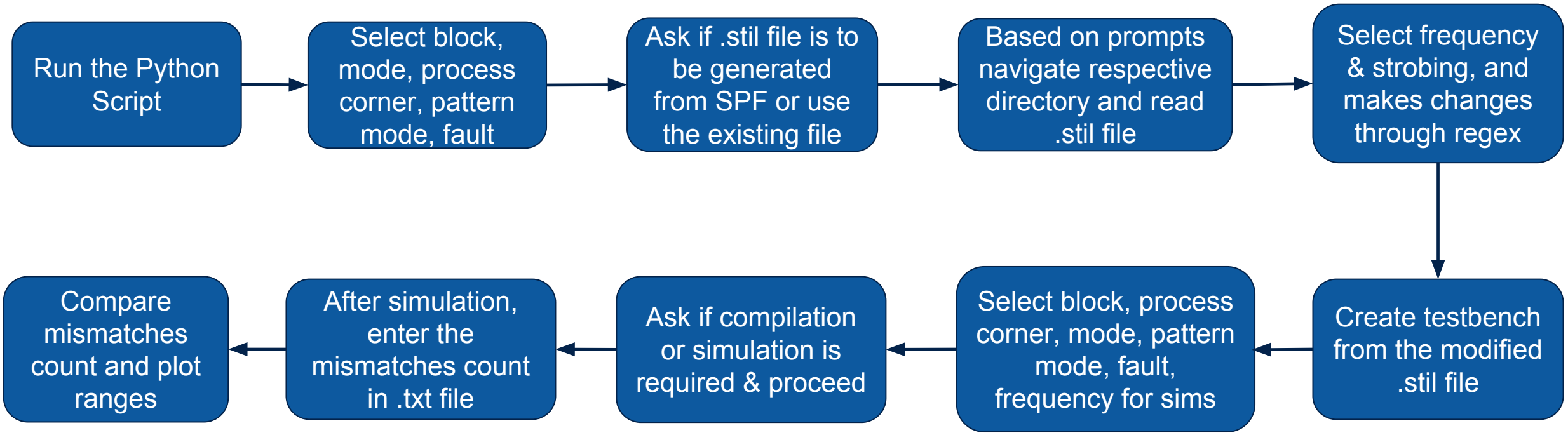
Freq (MHz)	Time period, Tr, Tf (ns)	Strobing range - SPHD LOGIC_MEM_COMP (ns)	Strobing range - SPHD LOGIC_COMP (ns)	Strobing range - SPREG LOGIC_MEM_COMP (ns)
1	1000, 900, 930	72-908	71-908	48-908
5	200, 150, 180	72-158	71-158	48-158
10	100, 50, 80	Fail	Fail	48-58
20	50, 20, 40	Fail	Fail	Fail
33.33	30, 15, 25	Fail	Fail	Fail

Results:

Maximum Shift Frequency:

1. SPHD block - 9.52 MHz
2. SPREG block – 12.5 MHz

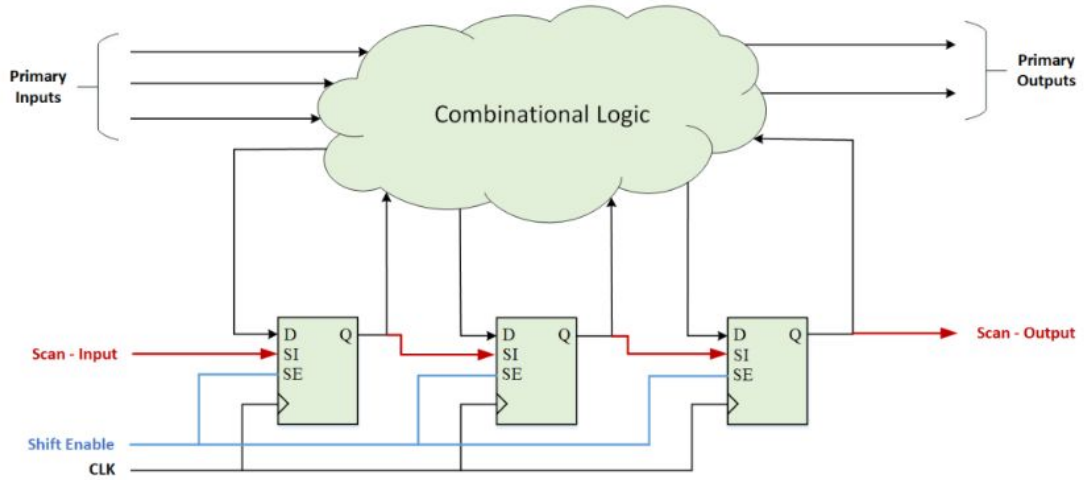
Automation flow for Max Chain Test Frequency



Capturing PI, PO maximum frequency

- Objective: In chain test, SI to SO frequency was found. Functional testing is done when SE=0 and PI, PO faults are added.
- Approach-1: Add only PI, PO faults using add_fault.
- Approach-2: Disable the shifting

- Conclusion: PI PO can be handled with dummy cycles.



In Approach-1,

For 3 possible hierarchies,

1. A_CORE/<signal>
2. <signal>
3. A_CORE/MEMBLOCK1_ST_inst/MEMBLOCK1_SPREG_inst/<signal>

For SPREG,

Frequency (Mhz)	Time period, Tr, Tf (ns)	Strobing (ns)	FF Mismatches	SS Mismatches (1)	SS Mismatches (2)	SS Mismatches (3)
1	1000, 900, 930	500	0	0	0	0
2	500, 400, 430	400	0	0	0	0
4	250, 180, 210	120	0	0	0	0
4.16	240, 180, 210	120	0	0	0	0
4.34	230, 170, 200	130	0	0	0	0
4.44	225, 175, 205	135	0	3484	5226	0
4.54	220, 170, 200	150	0	3571	5226	3571
5	200, 150, 180	145	0	5995	6097	6968
10	100, 50, 80	45	0	9057	15729	9057
20	50, 20, 40	20	0	9376	13949	9376
33.33	30, 15, 25	15	0	48272	48292	48272

Maximum frequencies: 4.34 MHz, 4.34 MHz, 4.44 MHz

Conclusion: Top Level PI, PO are critical paths
Solution: Dummy Cycles in PI, PO

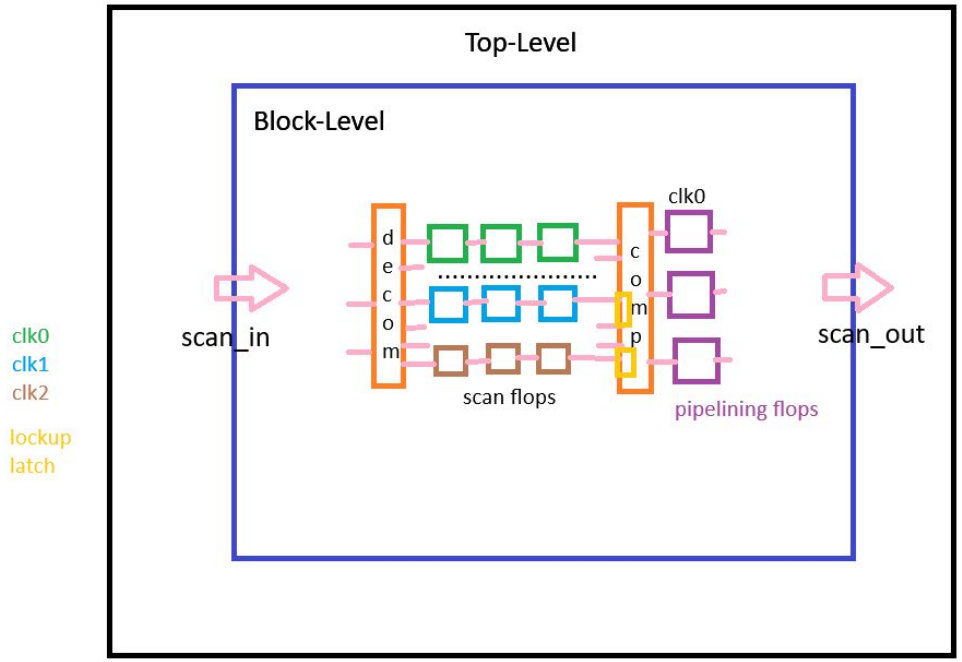
Pipelining Flops Insertion at Block Level

Objective: To insert pipelining flops in all modes at the output to enhance the shift frequency of SPHD and SPREG blocks.

Commands used:
set_dft_configuration -pipeline_scan_data enable
set_pipeline_scan_data_configuration -head_pipeline_clock CLK0 -tail_pipeline_clock CLK0 -head_pipeline_stages 0 -tail_pipeline_stages 1 -head_scan_flop true

Problems encountered: For DFT Insertion using TetraMAX, the design incorporates both compressed and non-compressed modes with multiple clocks (CLK0, CLK1, CLK2)

- Lockup Latches
- Compressor and Decompressor Logic



Observations & Results

Future Explorations:

- Top-Level Pipelining insertion
- In Codec logic, along with DFTMax, explore the other advanced tools like DFT Compression with serializer, DFT Ultra Compression, DFT SEQ Compression, which when used might not require lockup latches.

Mode	Scan Chains	Clock
LOGIC_MEM LOGIC	32	CLK0(28), CLK1(3), CLK2(1)
REGBANK	8	CLK0(1), CLK1(1), CLK2(6)
MEM	28	CLK0(27), CLK1(1)
LOGIC_MEM_COMP	32/165/4	CLK0, CLK1, CLK2
LOGIC_COMP	32/412/4	CLK0, CLK1, CLK2

```
Architecting Pipeline Structures
  Number of Head Pipeline Stages = 0
  Number of Tail Pipeline Stages = 1
Architecting scan compression mode LOGIC_MEM_COMP with base mode LOGIC_MEM
Architecting Load Decompressor (version 5.8)
  Number of inputs/chains/internal modes = 32/165/4
Architecting Pipelined Unload compressor (version 5.8)
  Number of outputs/chains = 32/165
  Information: Compressor will have 100% x-tolerance
Architecting scan compression mode LOGIC_COMP with base mode LOGIC
Architecting Load Decompressor (version 5.8)
  Number of inputs/chains/internal modes = 32/412/4
Architecting Pipelined Unload compressor (version 5.8)
  Number of outputs/chains = 32/412
  Information: Compressor will have 100% x-tolerance
```

Mismatches in Serial Sims due to Dummy Cycles

Objective: To find the root cause of mismatch of the pin DOUT[3] in serial simulation at top level in ATPG.

Observations:

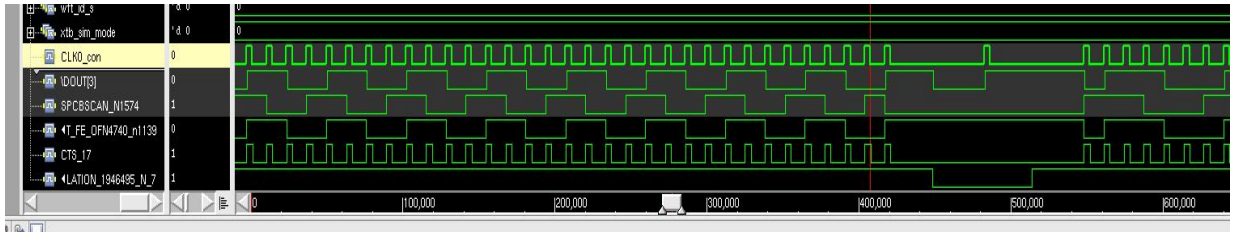
- All the mismatches are with scan cell 0 (the last scan - 685th) of the DOUT[3] scan chain with hierarchy,

A_CORE/A_MEMBLOCK1_ST_inst/MEMBLOCK1_SPREG_LOLEAK_inst/C13/mem_bist_wrapper_inst/COCO_SP/bist_collar_marchelement_q_reg_20x

- The mismatches are at the end of the patterns in scan chain of DOUT[3], which is the longest chain with no shadow cells of CLK1.

Procedure and Results

1. At the last shift, adding two dummy cycles V{"CLK0"=0; "CLK1"=0; "TCK"=0;}, where the clock doesn’t toggle but to meet the setup/hold violations to enter capture mode (min. 60ns required between scan enable and clk).
2. Due to the dummy cycles, trace back the flop in schematic viewer for the mismatching clock cycles, comparing the ATPG pattern cycles with the simulation results for the scan flop and DOUT[3] flop.



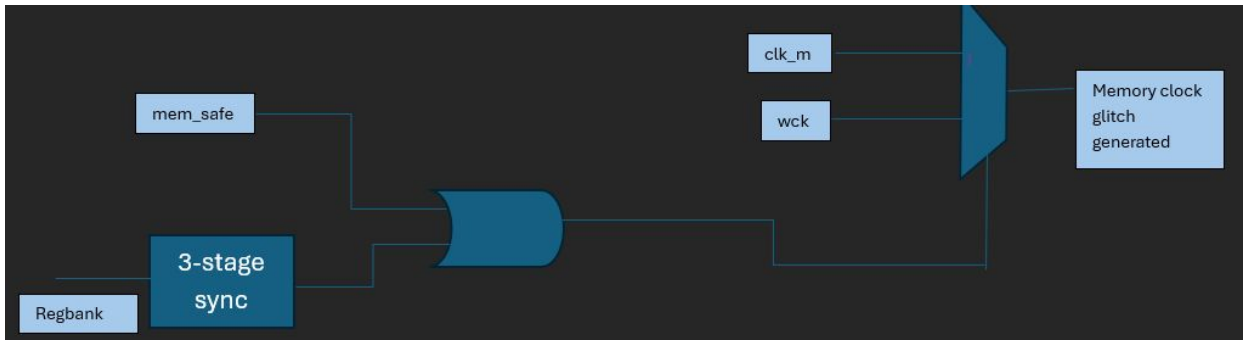
Results:

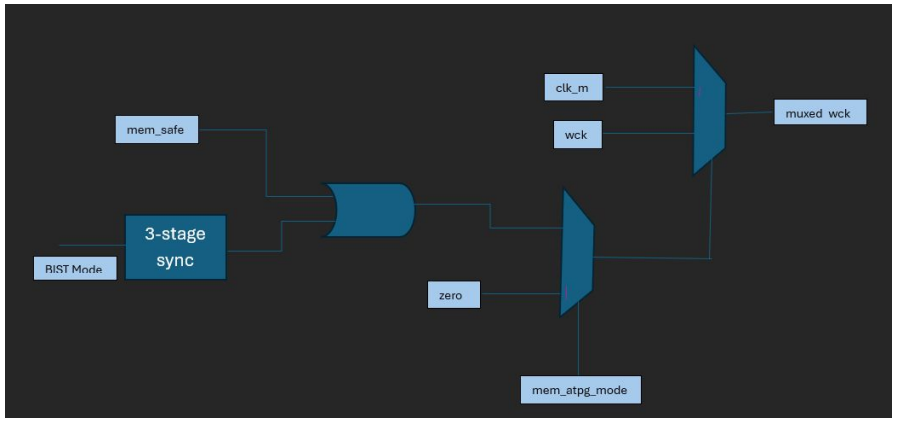
- Comparing the simulation results with patterns, the Q of the flop and DOUT[3] are same in the shift mode and the Q isn’t updating in the dummy cycles.
- No issue with the path between flop and DOUT.

BIST DFT/ATPG Design Issues

Issue 1:

- Glitch is generated on Memory Clock due the multiplexer controlled by read and write clocks causing unintended transitions in PROLIBP18 V3.0 CUT 28-31
- To bypass Synchronizer, in ATPG, a multiplexer with mem_atpg_mode pin as select line is used, which is not preset in the BIST.



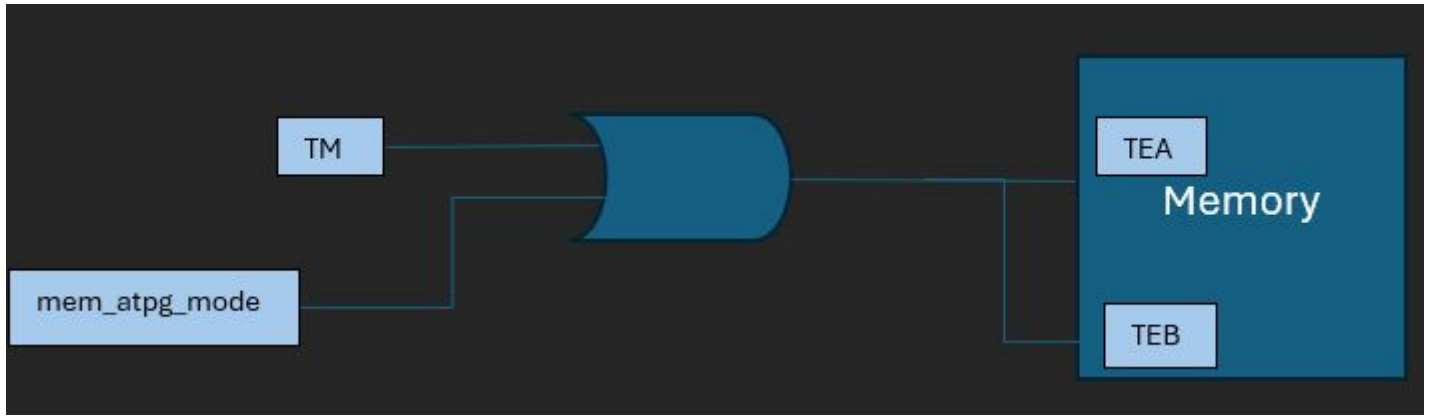


BIST DFT/ATPG Issues

Issue 2:

- Coverage drop by constraining the mem_atpg_mode pin to 1 in PROLIBP18 V5.0 RF2 Memory.

•mem_atpg_mode is recommended to be set as 1 in ATPG, but this leads to loss in coverage since TEA pin becomes 1 always, considering only TADD while ignoring ADD.



Verigy 93K ATE Pattern Conversion & Simulation

- Convert STIL ATPG patterns generated from DFT into ATE format compatible with Verigy 93K.
- Generate the testbench for comprehensive testing of the device for Compression and Non-Compression modes with pattern sets as LOGIC_MEM_atspeed, LOGIC_MEM_lowspeed, LOGIC_MEM_sa and Debug Patterns for MEMBLOCK_SPHD and MEMBLOCK_SPREG.
- Two operating conditions simulated:
  - **HS0LS0**: CTRL[1] = 0, CTRL[4] = 0 (FF corner)
  - **HS1LS1**: CTRL[1] = 1, CTRL[4] = 1 (SS corner)

Python version: 3.6.8 (default, Nov 15 2024, 08:11:39) [GCC 8.5.0 20210514 (Red Hat 8.5.0-22)]

Block	Mode	Fault Type	Simulation Status	Compare Operations	Simulation Start Time
MEMBLOCK1_SPHD_LOLEAK	LOGIC_MEM	atsp	Passed	21210619	2025-07-02 19:25:00
MEMBLOCK1_SPHD_LOLEAK	LOGIC_MEM	sa	Passed	8081203	2025-07-03 05:24:05
MEMBLOCK1_SPHD_LOLEAK	LOGIC_MEM	lowsp	Passed	23409	2025-07-03 05:22:49
MEMBLOCK1_SPHD_LOLEAK	LOGIC_MEM_COMP	atspd	Passed	5983889	2025-07-02 19:25:01
MEMBLOCK1_SPHD_LOLEAK	LOGIC_MEM_COMP	lowspd	Passed	4522621	2025-07-02 21:13:31
MEMBLOCK1_SPHD_LOLEAK	LOGIC	sa	Passed	12297167	2025-07-02 23:09:45
MEMBLOCK1_SPHD_LOLEAK	ONLY_MEM	sa	Passed	4875059	2025-07-03 05:20:13
MEMBLOCK1_SPHD_LOLEAK	LOGIC_MEM_COMP	sa	Passed	1808646	2025-07-02 22:37:44
MEMBLOCK1_SPHD_LOLEAK	REGBANK	sa	Passed	965933	2025-07-03 06:07:53
MEMBLOCK1_SPREG_LOLEAK	LOGIC_MEM	lowspd	Passed	21514528	2025-07-02 20:58:17
MEMBLOCK1_SPREG_LOLEAK	LOGIC_MEM	sa	Passed	18405818	2025-07-02 22:52:33
MEMBLOCK1_SPREG_LOLEAK	LOGIC_MEM	atspd	Passed	21877130	2025-07-02 19:25:01
MEMBLOCK1_SPREG_LOLEAK	LOGIC_MEM_COMP	sa	Passed	1404134	2025-07-02 20:19:07
MEMBLOCK1_SPREG_LOLEAK	LOGIC_MEM_COMP	lowspd	Passed	4054641	2025-07-02 19:59:12
MEMBLOCK1_SPREG_LOLEAK	ONLY_MEM	sa	Passed	2943775	2025-07-02 22:29:03
MEMBLOCK1_SPREG_LOLEAK	REGBANK	sa	Passed	925298	2025-07-02 22:35:19
MEMBLOCK1_SPREG_LOLEAK	LOGIC	sa	Passed	18405818	2025-07-02 20:27:30
MEMBLOCK1_SPREG_LOLEAK	LOGIC_MEM_COMP	atspd	Passed	6279448	2025-07-02 19:25:00

For FF Corner

For SS Corner

AI in VLSI

Mimics Human Intelligence of learning, reasoning, predicting & perceiving.
AI in CAD used tools for faster Convergence & Optimization

- AI Fundamentals in VLSI – AI, ML, DL, GenAI, LLM, NN (ANN, CNN, GNN, RNN)
- Benefits of AI in VLSI – Increased design efficiency & reduced manual effort, Improved accuracy & early defect detection, Cost reduction & enhanced performance, Scalability with growing design complexity.
- Challenges – Large & high-quality data requirement, Model interpretability & integration complexity, Computational resource demands, Security & trustworthiness of AI-driven designs.

Cntd: AI in VLSI

- AI Applications in VLSI – Automated PnR, Timing & Power Optimization, Yield Prediction & Process Optimization, Bug Detection & fault prediction, Dynamic Voltage/Frequency Scaling.
- Key AI Usage areas - Physical design (placement, routing, timing closure, sign-off), Analog design (transistor sizing, biasing, topology), Manufacturing process optimization, Design verification, FPGA emulation, testing
- Exploration Areas – Analog & digital design optimization, System-level multi-physics simulation, Technology & library development, Documentation & AI-assisted design consultation
- More on AI in VLSI - Link

Future Explorations

1. For Pipelining Flops, In Codec logic, along with DFTMax, the other advanced tools like DFT Compression with serializer, DFT Ultra Compression, DFT SEQ Compression, which when used increase the efficiency.

2. In REGBANK, Error Correction Codes (ECCs) logic implementation ensuring SCDCTD avoiding the risks of data flips.

3. AI in VLSI, implementing agentic flow in various stages of DFT like DFT Insertion, Pattern Generation flow, Backend floor planning, PnR, and flat sign-off (apart from automation), improving the work efficiency.

Projects by me at IITH

1. RTL Design for Independent Component Analysis (ICA) using CORDIC - Verilog RTL Design for FastICA including Orthogonalisation (GSO), Normalization, & Estimation and Update for 1024 samples using CORDIC doubly pipelining for 7D.

2. Quick Connect Multi-Sensor Project using Renesas RA6E2 MCU - Integrated various sensors like ZMOD4510, ZMOD4410 and HS4001 with Renesas RA6E2 MCU and connected with the Bluetooth module coded using e2studio for real-time monitoring of Temperature, Humidity, and various pollutants like TVOC, ETOH, CO2.

3. A 0.4V 0.93nW/kHz Relaxation Oscillator Exploiting Comparator Temperature Dependent Delay to Achieve 94-ppm/°C Stability for 65nm Technology - Virtuoso Design of the Relaxation Oscillator using PTAT – CTAT properties.

Thank You