



SHAPING THE NEXT GENERATION OF ELECTRONICS

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA





SHAPING THE NEXT GENERATION OF ELECTRONICS

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

Watt's Up with DDR5

Formal Verification Framework for Robust DRAM Power Management

Pradip Prajapati, Anshul Jain, Rocco Salvia, Mounica Kothi, Erin Rasmussen

Intel Corporation



Motivation

CONTEXT

DDR5 Double Data Rate 5

- Significant improvement over DDR4
- Advances memory landscape in AI, cloud and HPC

Higher Memory
Performance

Greater Power
Efficiency

Increased
Reliability

Improved Error
Correction

Data Rates 6400
MT/s

Capacity
128GB/DIMM

CHALLENGE

New Features & Modes

- Improves performance & power efficiency, **BUT**
- Major technical challenges in verifying **Power Management (PM) Flows**

Power States
Management

Active Power
Management

Idle Power
Management

Power Failure
Scenarios

Voltage
Regulations

Thermal
Management

RELEVANCE

Innovative FV Solution

- FPV based exhaustive Validation of PM flows
- Encouraging results; successful optimization of DDR5 Power Management

Exhaustive
Verification

Standard
Checkers

Cross-Flow
Coverage

Power Down
Modes

Refresh &
Self-Refresh

Achieve
Shift-left

Methodology

Design Features



Design Complexity

Power Down Mode

Active when no transactions are pending

Refresh Operation

Cells to be periodically refreshed to retain data

Self Refresh Operation

DRAM retains data w/o external clocking

Pre-requisites

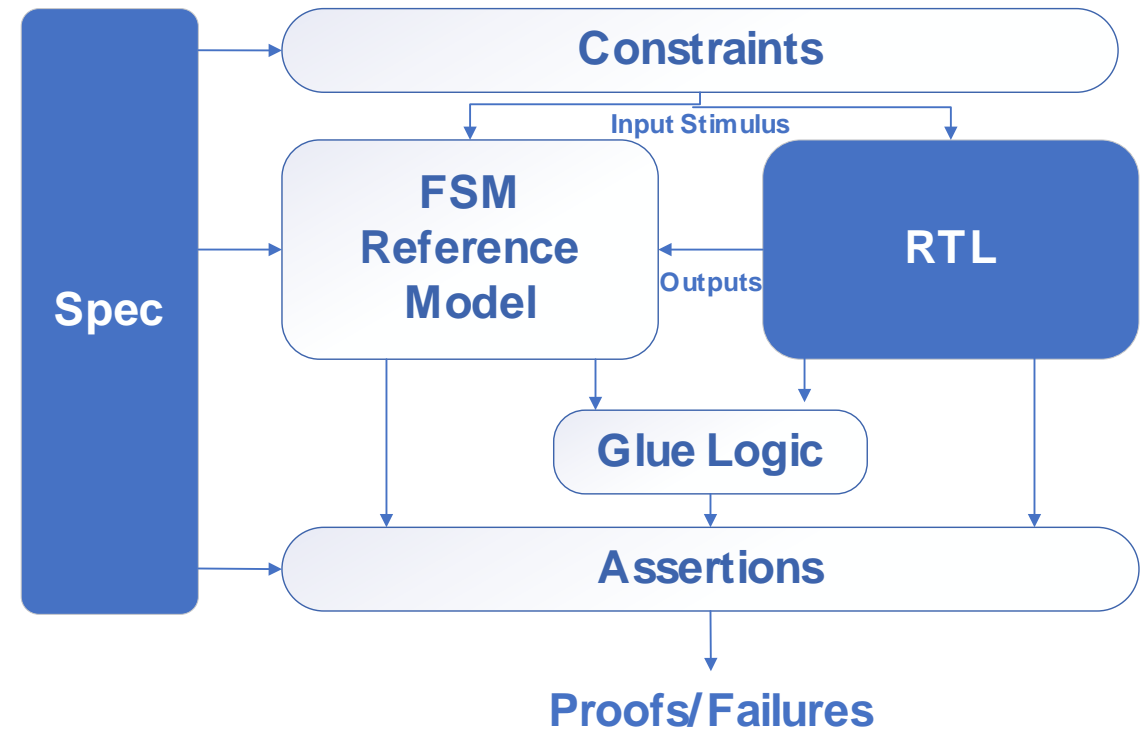
to enter any low power state

Multi-flows optimization

Maximize bandwidth while saving power

Formal Verification Strategy

FSM-based Reference Model + SVAs



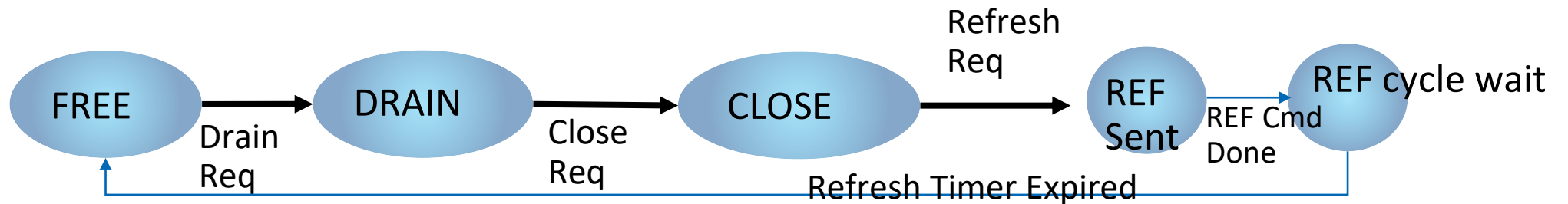
Formal Verification Implementation Details

FSM-Based Reference Model

- Based on **sequence** of **events** defined in **DDR5 specification**
- Keeps track of **tasks performed by DUT** using **outputs**
- Simplifies modelling of SVA properties by mapping an event in the sequence to a state

Example: Refresh FSM Model

1. Send request to scheduler to drain queues
2. Close all pages in target banks in memory
3. Send refresh to those banks
4. Wait for refresh cycle before sending next command
5. Allow new commands once timer expires



Formal Verification Implementation Details

- Power management block talks to other blocks through handshake protocols
- Check if all the requests and acknowledgements follow handshake
- Two widely applicable requirements are
 1. Handshake requests cannot fall before ack
 2. If there is no request, there should be no ack

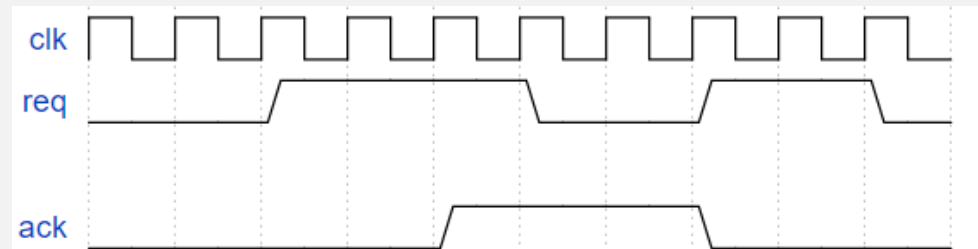
$req \ \&\& \ !ack \rightarrow \#\#1 \ req$
 $\!req \rightarrow \!ack$



1. Handshake Checker

Bug Example

- Bug scenario: Scheduler drain request sent by Refresh dropped before it is ack'd i.e. scheduler drained
- Root cause: Ack from a previous req overflowed to the next req. DUT incorrectly identified this as an ack for this request and dropped the request.



Formal Verification Implementation Details

2. Timing Checker

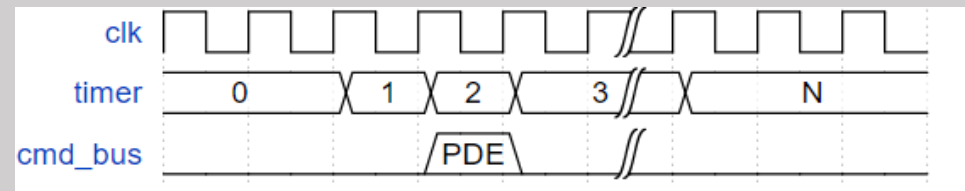
- DDR5 has many strict timing requirements between certain events to maintain integrity of data stored.
- Check if event B should not occur if event A has occurred in last N cycles. If two events meet timing requirements

$evenA_in_last_N_cycles \rightarrow !eventB$

(N: Parameter/Configuration Register)



- Spec: Before sending PDE (Power Down Entry) command scheduler should be blocked for at-least N cycles. (N is spec specified value)
- Bug Scenario: DUT sent a PDE on the command bus before scheduler was blocked for N cycles
- Root cause: There were two FSM paths to sending the PDE, where one path did not consider the timing requirement



Bug example

Formal Verification Implementation Details

- FSM related DUTs generally have strictly defined sequence of events.
- Check if a prerequisite event defined in the spec has not occurred the consequent event should also not occur.

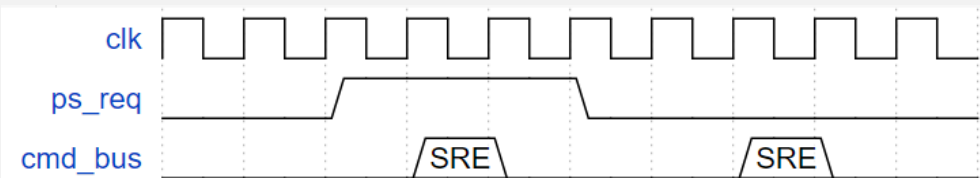
!prerequisite_event → !consequent_event



3. False Trigger Checker

Bug Example

- Spec: In the self refresh flows, if there is no pending power state request DUT should not send a SRE (Self Refresh Entry) request
- Bug Scenario: DUT sent SRE command twice, when request asserted only once.
- Root cause: An internal flag did not reset after the first SRE was sent, which caused the flow to trigger again



Formal Verification Implementation Details

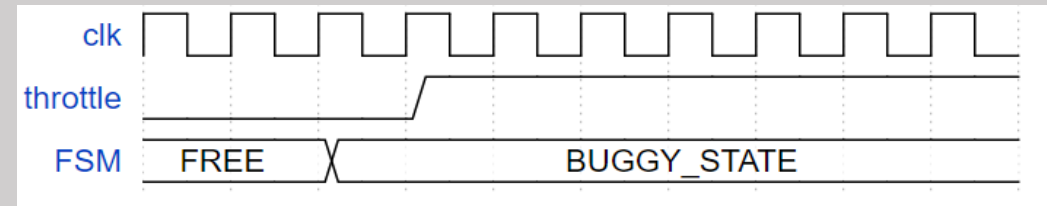
4. Prompt Trigger Checker

- DUT should not be stuck in a specific state significantly longer than required
- Check if, once the prerequisites to move to the next state are met, DUT makes the forward progress.
- Orthogonal to False trigger check.
- False-Prompt trigger combination central to exhaustive verification

all_prerequisite_events → consequent_event



- Spec: When the DIMM is throttling, DUT should send PDE (Power Down Entry) command
- Bug Scenario: When the DUT is in a specific state, it incorrectly ignores the throttling indication and does not trigger the PDE flow
- Root cause: In this state, DUT waits for other events to complete which are not prerequisites



Bug example

Formal Verification Implementation Details

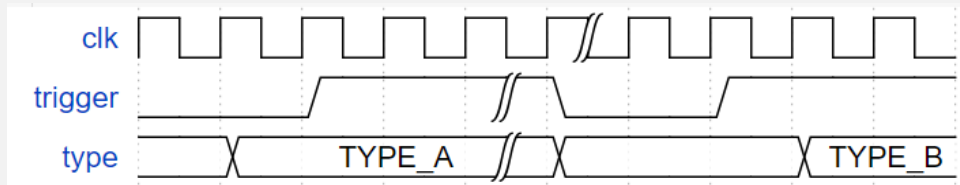
- Many common tasks between various flows of the DUT
- Same tasks should execute in a consistent manner across flow/states unless specified by spec.
- Not any standard checker type, format may differ for different tasks



5. Inconsistency Checker

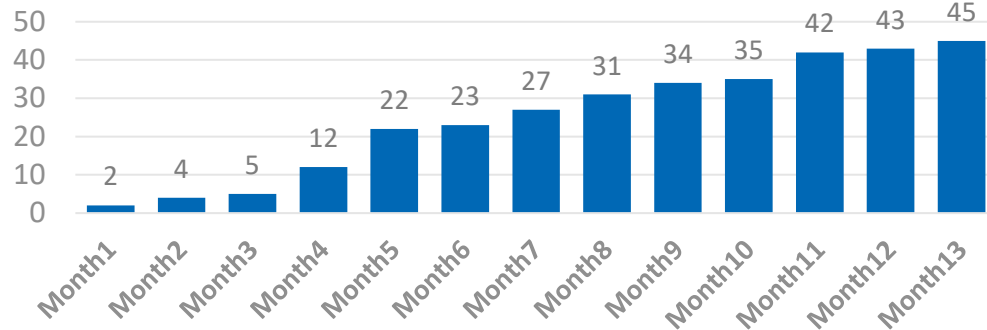
Bug Example

- Bug scenario: In two different states, DUT determined refresh type in different manner. This was not specified in the spec.
- Root cause: In one state refresh type was determined before refresh trigger. In another state refresh type was determined after refresh trigger.



Results

Bug Trend



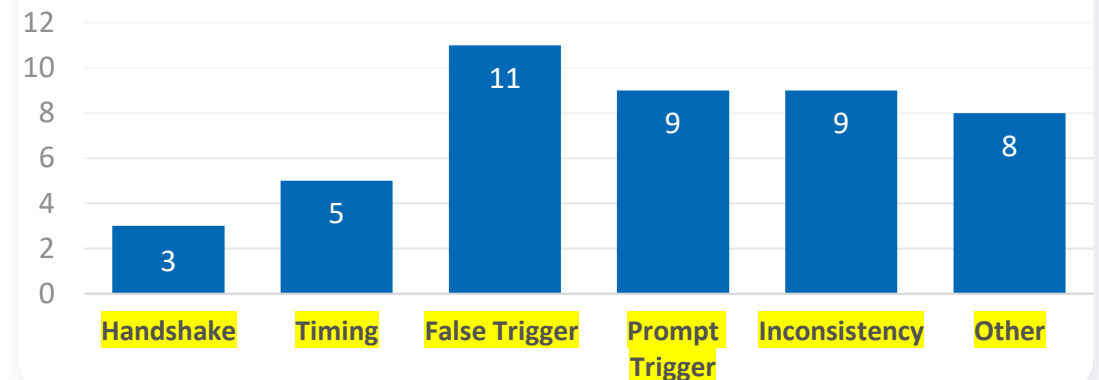
Collateral Generated

- 3 FSM reference models
- Total 46 FSM states
- Assertions: 300
- Constraints: 311
- Functional Covers: 611

Resources & Effort

- One dedicated engineer with 50% BW
- Results over 54 work weeks
- Total engineering hours ~1100
- Around 25 hours/bug

Bug Classification



Summary

- 1 Successful Verification of DDR5 Power Management using **Formal Technology**
- 2 Detailed **Formal Verification Framework** for validating Power Management Flows
- 3 **Encouraging Results** both in terms of Bugs Found and Sign-off Achieved
- 4 **Improved Accuracy & Efficiency** of DDR5 DRAM Memory Controller