Architectural Formal Verification of System-Level Deadlocks

Mandar Munishwar
Qualcomm Atheros, Inc.

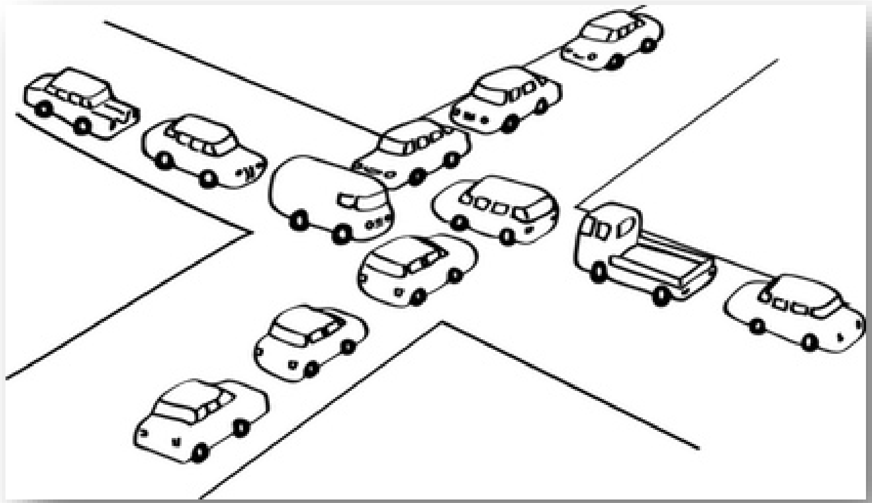
Vigyan Singhal
Oski Technology, Inc.

Agenda

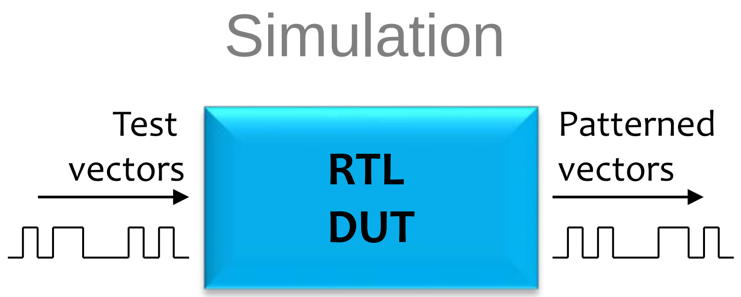
- What is System-level Deadlock ?
- Design Under Verification
- Architectural Formal Verification Methodology
- Results

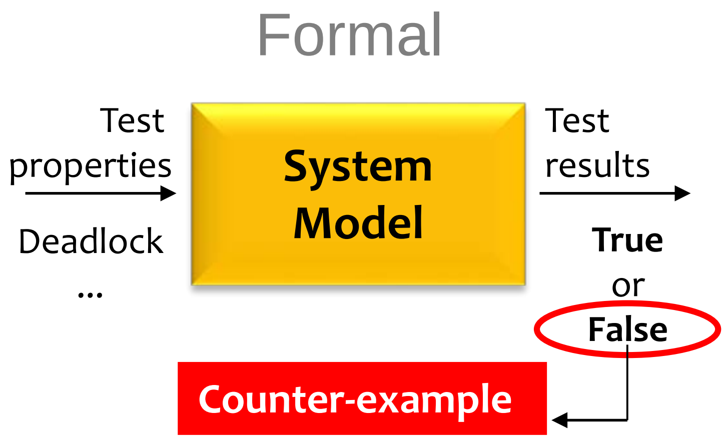
System-Level Deadlocks

- Possible forward progress issues:
  - Deadlock
    - Cyclical dependency between two or more processes
    - Mutually blocking each other
    - Each waiting for resources that can never become available
  - Livelock
    - Processes appear to be progressing
    - But can never terminate or reach their final state
  - Starvation
    - Process never gets a required resource
    - It always loses competition for share resources



Formal Verification is Needed

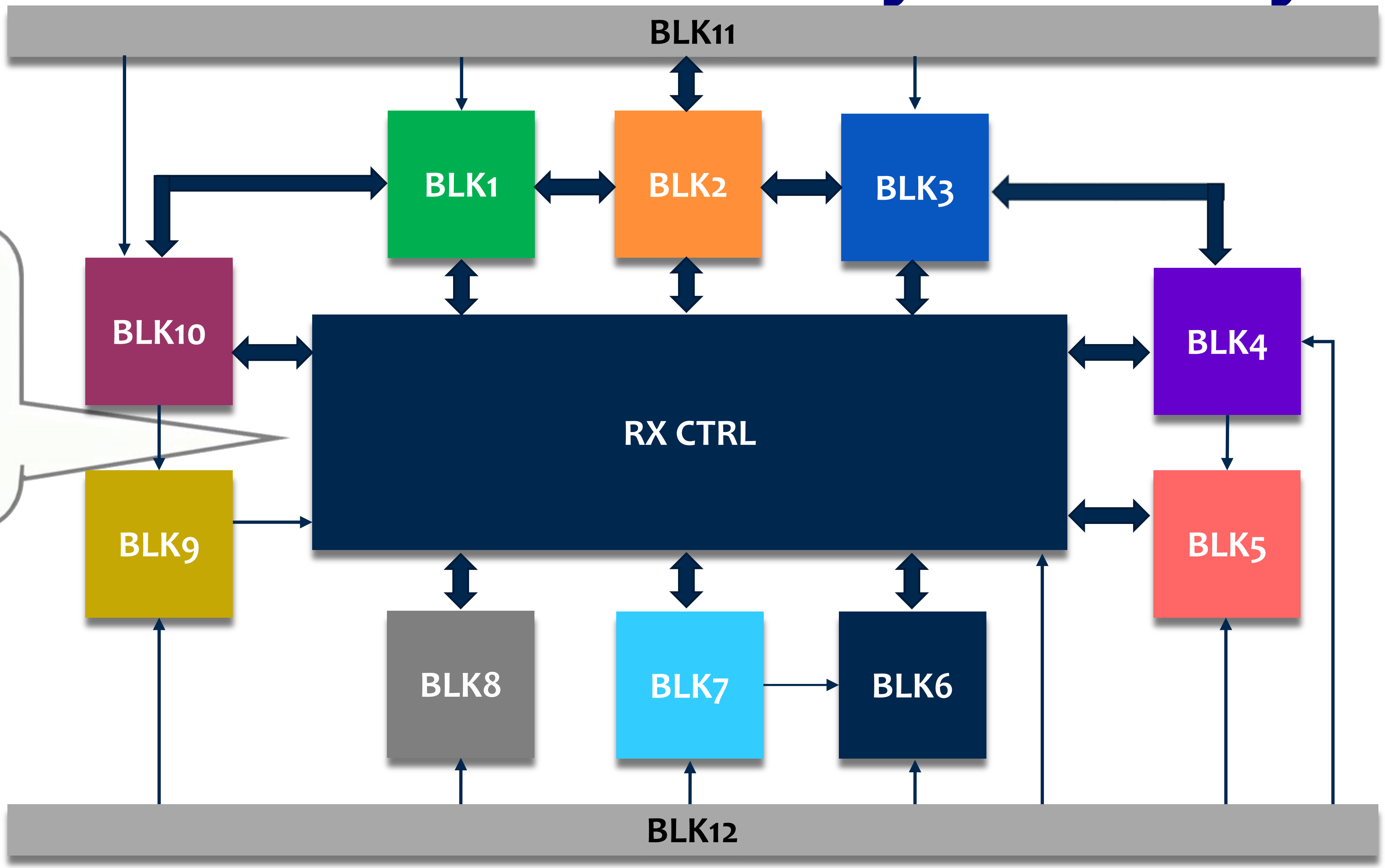




- Cannot cover all possible cases
  - Possibility of missing subtle corner-case bugs
- Bugs found late-stage
  - More expensive to fix

- Exhaustive
  - Equivalent to simulating all possible scenarios
  - But, system-level complexity too much for RTL formal
- Sign-off
  - Abstract models required
  - No bug left behind if done properly

Wireless SOC PHY Layer Sub-System



Central controller sequences operations of all other blocks

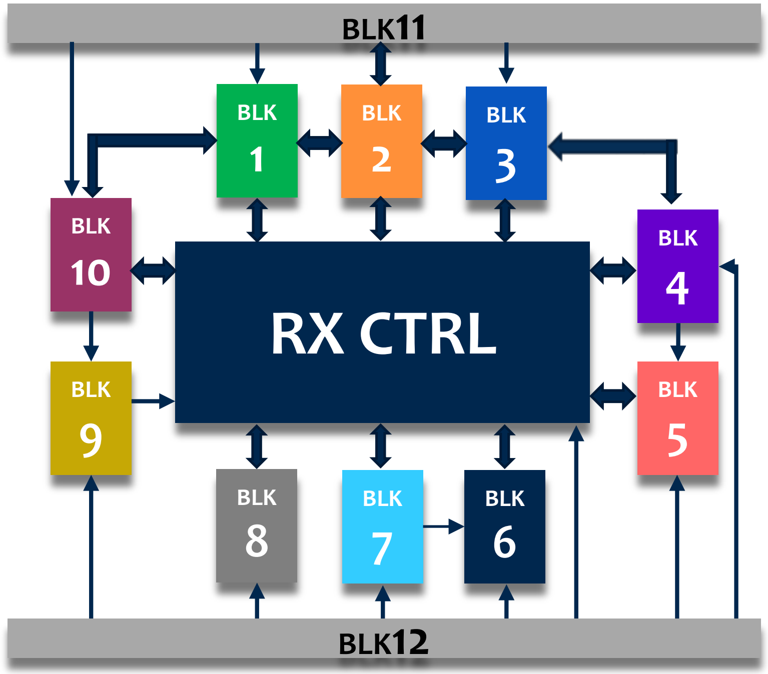
Background on DUT

Controller of next Generation WiFi Core for Mobile SoC

1. Feature Rich WiFi Subsystem
   - 2x2 MIMO – High throughput
   - Beamforming - extended range
   - 802.11AC Wave2 compliance

2. Makes very challenging to verify
   - Block is deep inside the design - limited observability/controllability
   - Interacts with Multiple block controllers
   - Susceptible to dead-locks

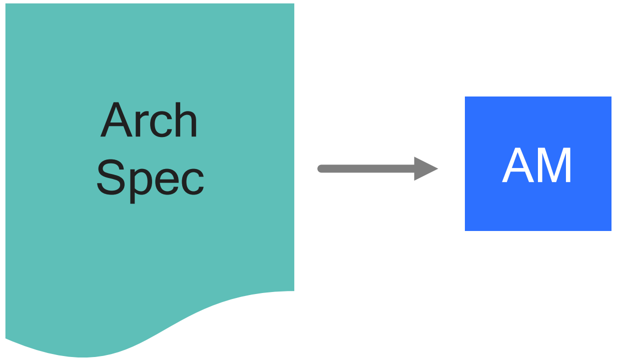
3. Formal is the way to build confidence and de-risk, but …
   - Too big a system for Traditional formal (FPV) to handle …

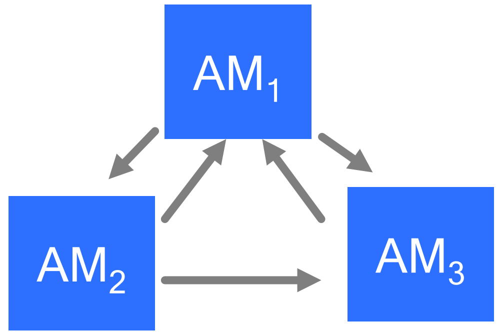


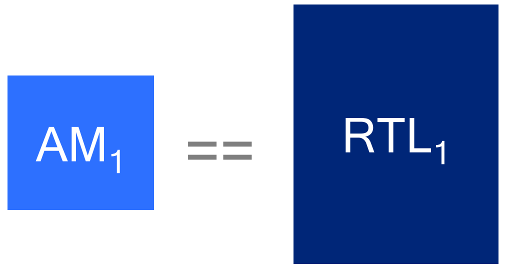
Architectural Formal Verification

Three-Step Process

1. Develop Architectural Models (AMs) for the Blocks in the System
   - Assume the block-level “contract” that contributes to the system-level requirement
   - Abstract internal details not relevant to the system-level requirement
     - E.g. For deadlock, model only the passing of control between block
   - Allow non-deterministic latencies
2. Formal verification of System of Block AMs
   - Prove that collection of block-level contracts imply system-level requirement
     - E.g. Prove forward progress checks at the system-level
3. Validate Block AMs versus RTL
   - Check that assumptions made about block AMs are true of the RTL
   - Any mismatch is a problem

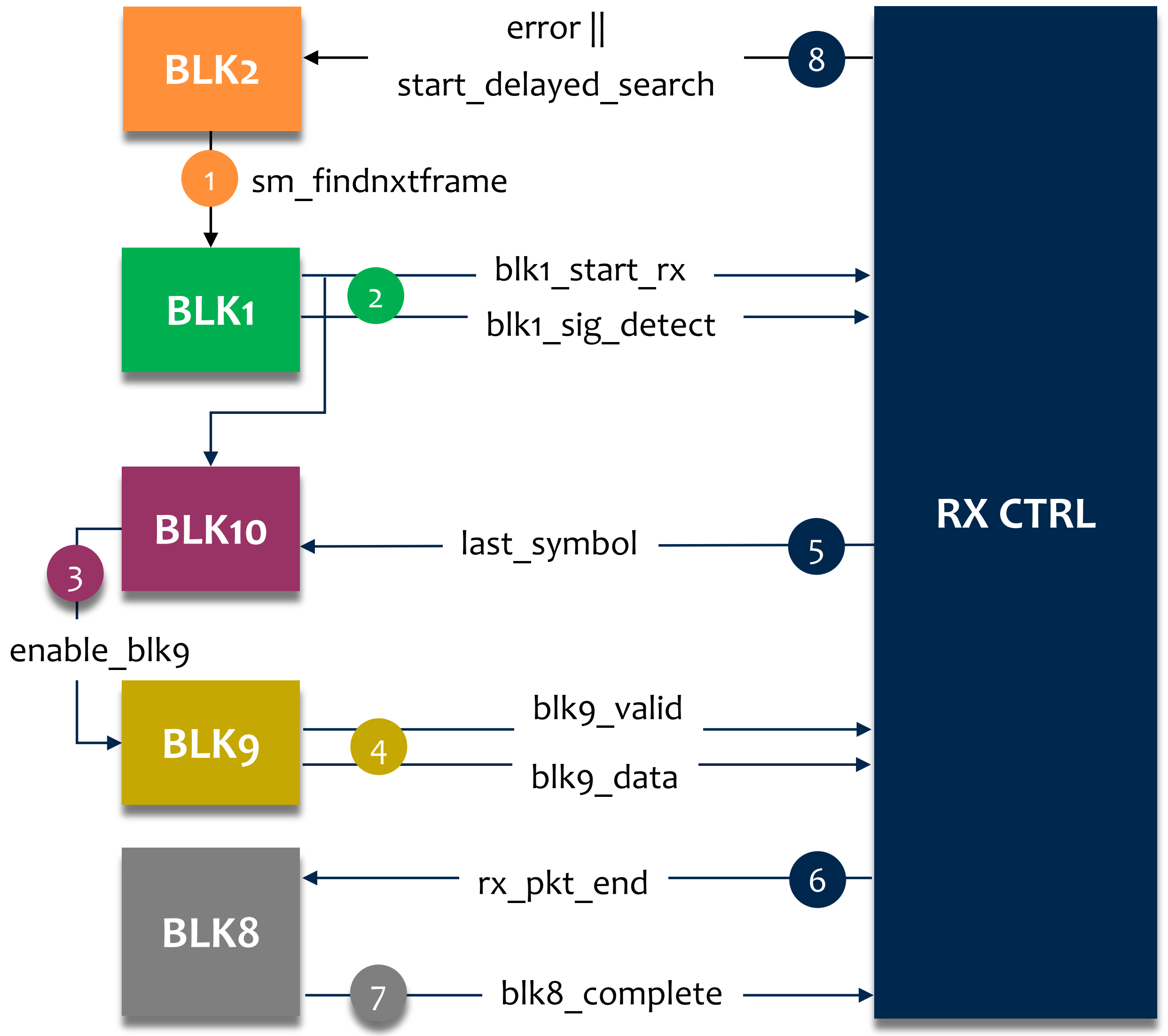






Example Packet Control Flow

Step #	Description
1	BLK2 asks BLK1 to find next packet in the channel
2	Indicates start of a new RX frame; End of BLK1 operation; Trigger BLK10
3	BLK10 enables BLK9; It has counters to add delays to make sure that BLK9 data arrives after data from BLK1
4	BLK9 sends data
5	RX CTRL notifies last symbol information to reset BLK10
6	RX CTRL notifies packet end to BLK8
7	BLK8 sends data; BLK8 sends a complete flag to mark end of data
8	Indicate completion/termination of frame



Step 1: Block-Level Architectural Modeling

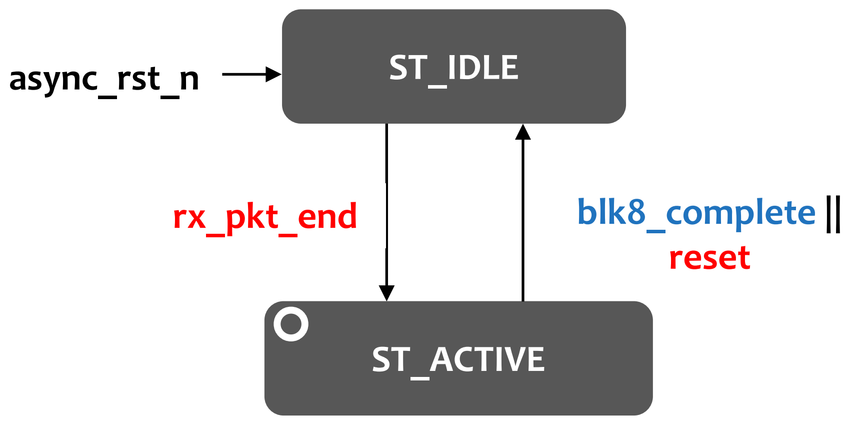
Example: BLK8

- FSM tracks the control state of the block
- SVA properties (assumes) model the output behavior

BLK8 (AM)

Inputs: Red

Outputs: Blue

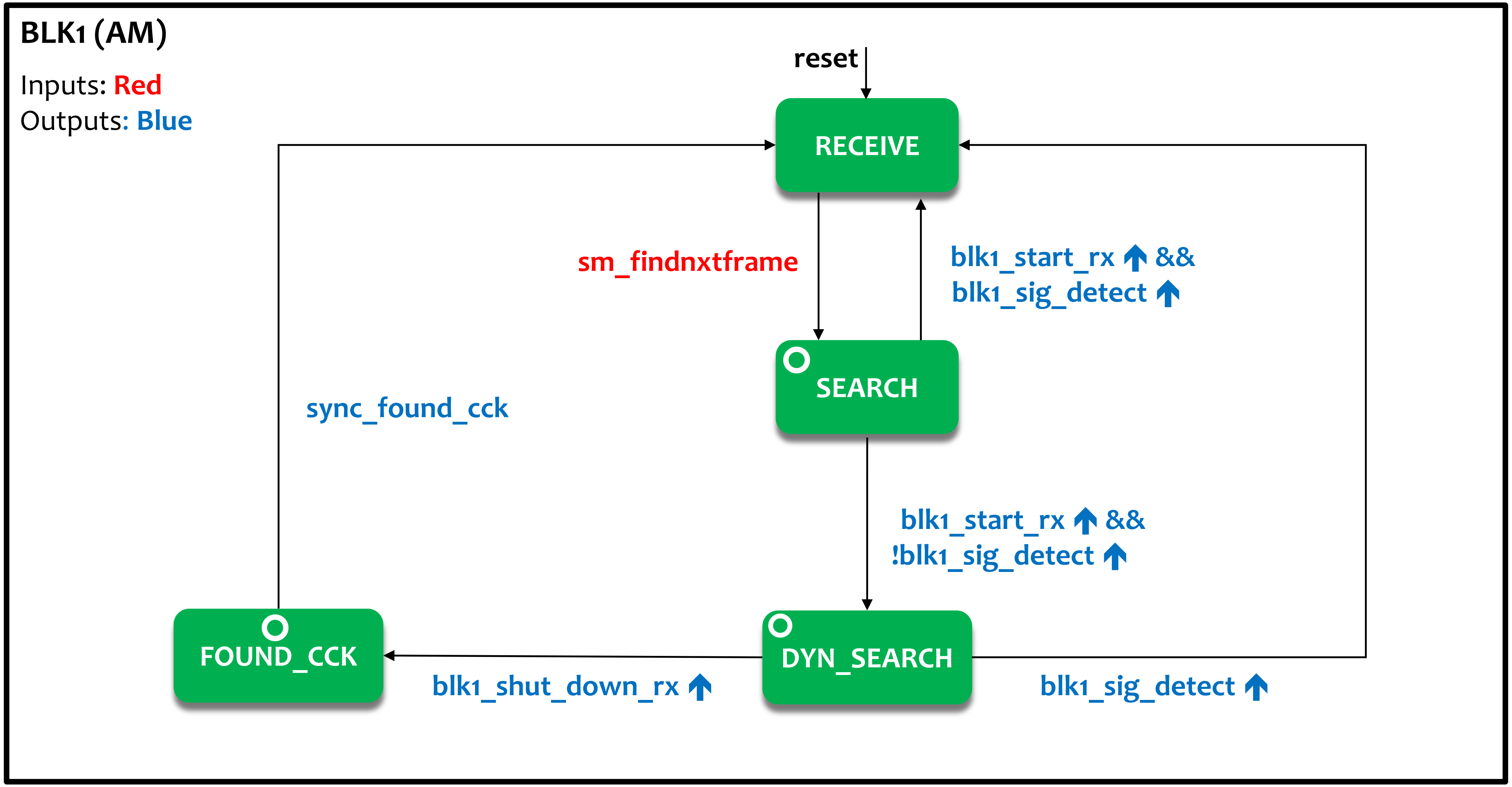


Property 1:
blk8_complete should not be asserted in ST_IDLE state

Property 2:

-
-
-

Example: BLK1 Architectural Model



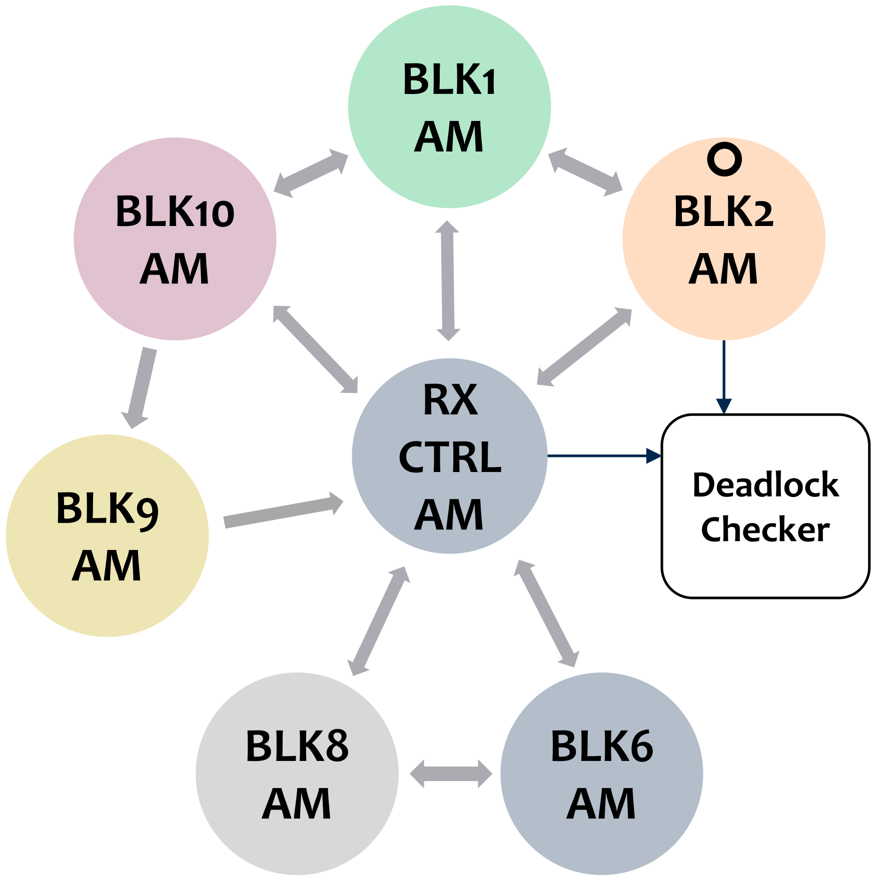
Step 1: Block-Level Architectural Modeling

Results

- Simply the act of developing the AMs exposed some architectural specification issues
- Example:
  - Spec included recovery mechanism for shutdown during Rx frame reception
  - Recovery after shutdown during Tx frame was not specified
  - Discovered an inconsistency in the contract between the MAC and PHY

Step 2: System-Level Verification

System Model



A subset of AMs is shown here for simplicity

- Assembly of block AMs form system-level model
- Models all possible flow control operations between the blocks and the central controller
- End-to-end checker for deadlock

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Step 2: System-Level Verification

Deadlock Checker

```
// Reset counter on either of
// 1. error
// 2. start_search
if (error || start_delayed_search_ofdm) begin
   counter <= 'd0;
end
else begin
   if (tx_frame) begin
      counter <= 'd0;
   end
   else if (sm_findnxtframe) begin
      counter <= 'd1;
   end
   else begin
      counter <= (counter == 'd0) ?
                    'd0 : (counter + 1'd1);
   end
end
```

```
phy_deadlock_a: assert property (
   @(posedge clk) disable iff (rst)
   (counter < TIMEOUT)
);
```

- Timeout safety property for deadlock
- Count up until:
  - Frame is completed successfully, OR
  - Frame is terminated due to an error
- Count should not exceed max upper bound on length of WiFi frames

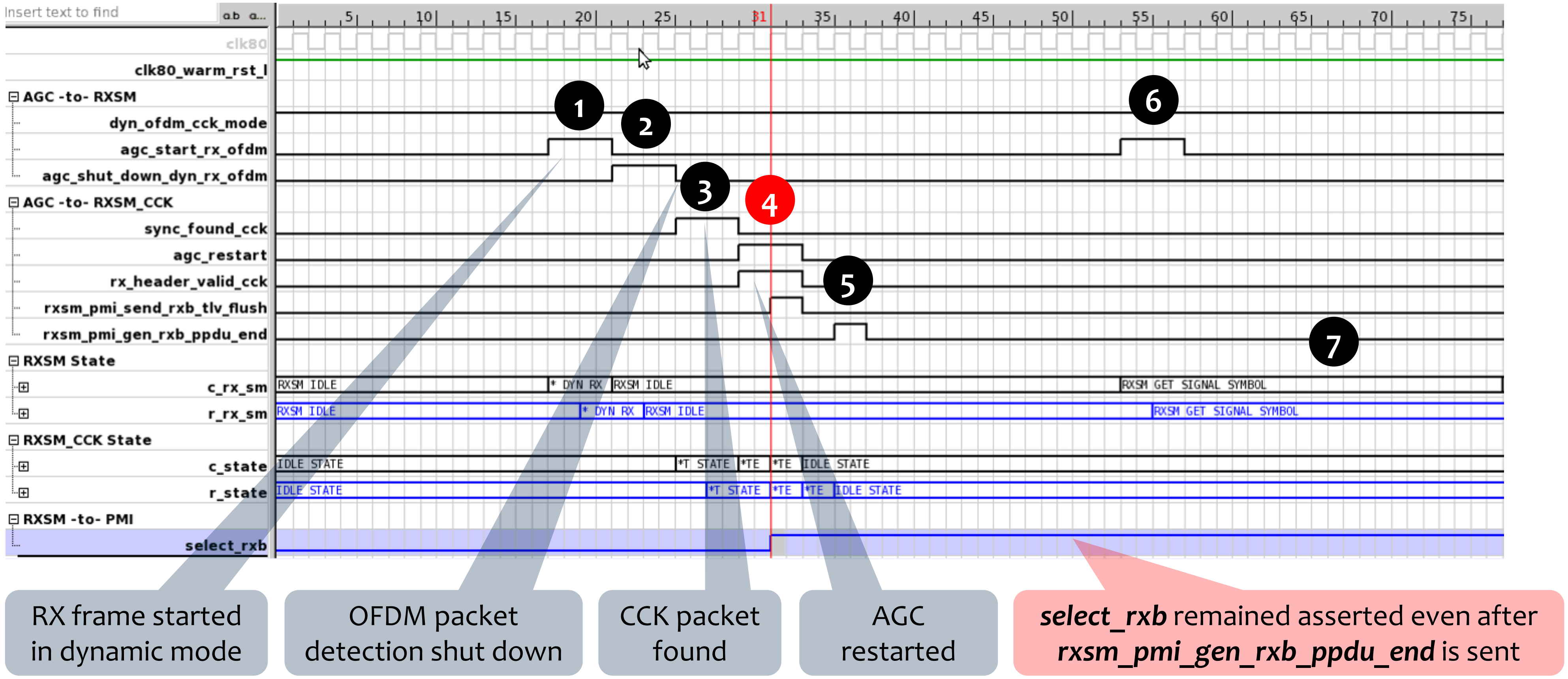
Step 2: System-Level Verification

Abstractions and Coverage

- Real time limit is too long for formal to handle (microseconds)
- Abstraction:
  - Reduce the time required proportionally for each block to process the frame
  - Overall frame time limit reduced to <200 clocks
- Bounded proof coverage used to sign-off on the system-level checker
- Measure depth of witness waveforms for functional and code coverage goals of the system-level model
- Ensure witness wave depths are less than bounded proof depths

Step 2: System-Level Verification
Results

- Found system-level deadlock bugs
- Example: Terminated packet causes RX_CTRL to become stuck in a wait state forever



RX frame started in dynamic mode

OFDM packet detection shut down

CCK packet found

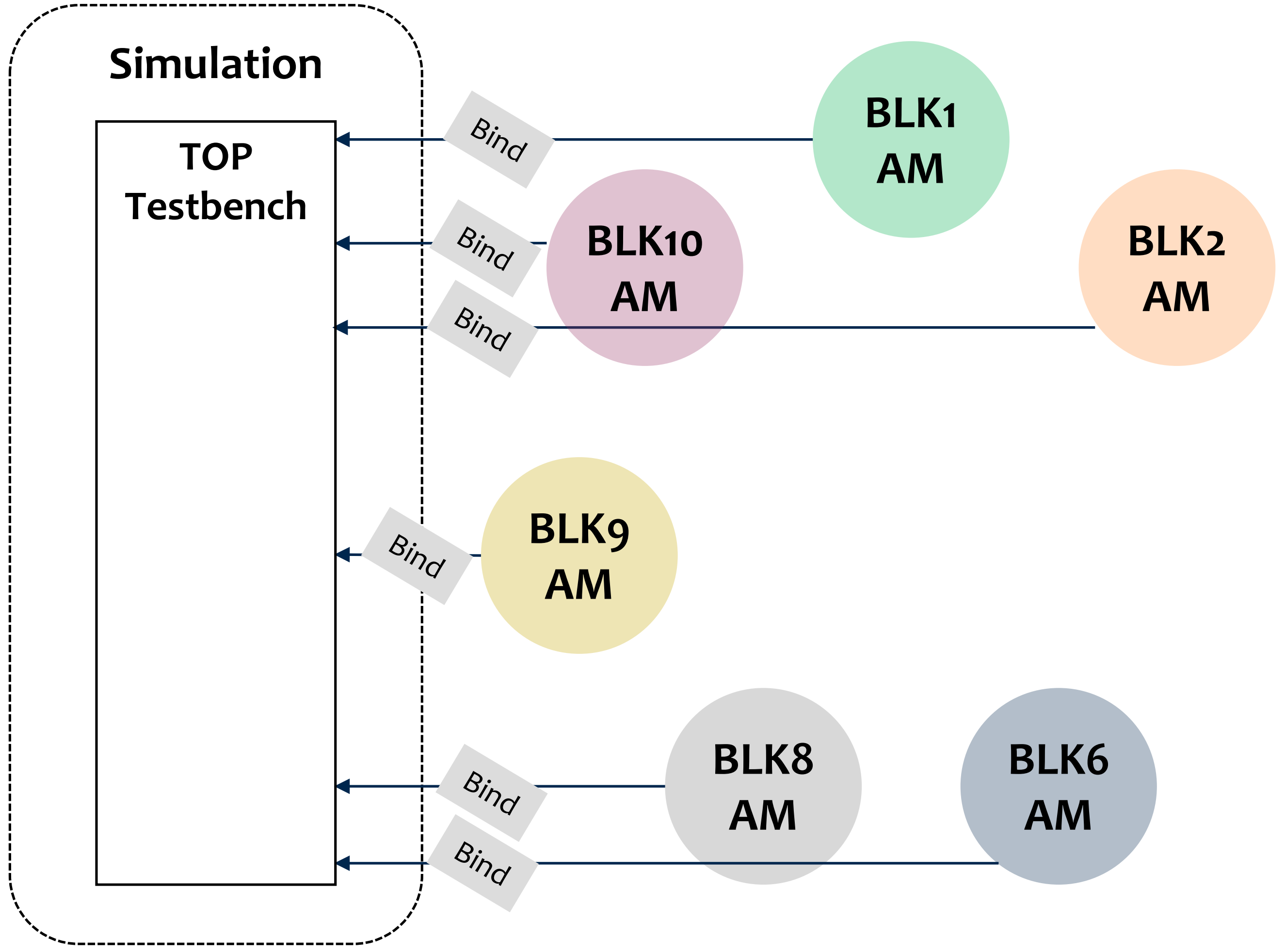
AGC restarted

select_rxb remained asserted even after ***rxsm_pmi_gen_rxb_ppdu_end*** is sent

Step 3: Validate AMs Versus RTL

Simulation Method

- Bind AMs to RTL in simulation environment
- Low effort approach
- Sufficient for relatively simple AMs

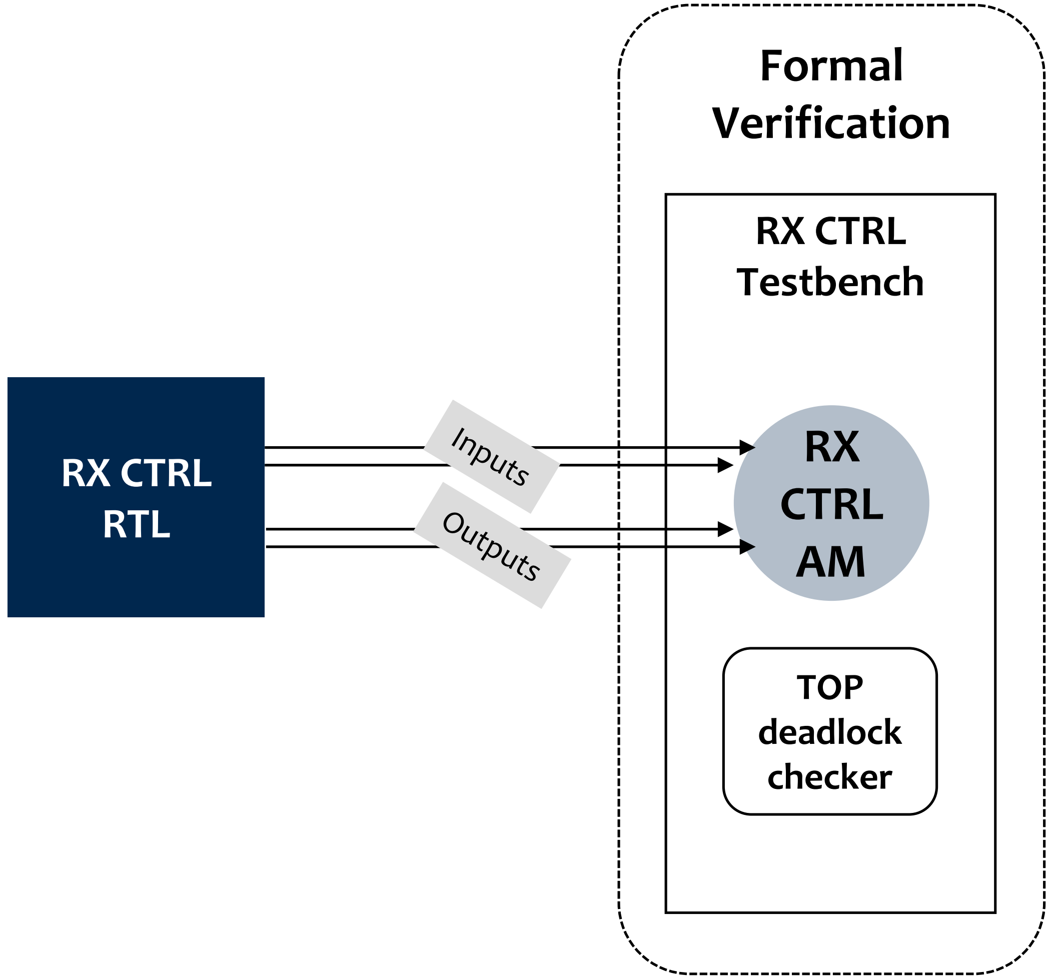


A subset of AMs is shown here for simplicity

Step 3: Validate AMs Versus RTL

Formal Method

- Formally verify assumptions made in AM
- Prove SVA properties on the RTL
- Abstractions use for large RTL counters
  - Packet length counter
  - Wait counter
- Bounded proof coverage used to sign-off



Example: RX_CTRL Architectural Model

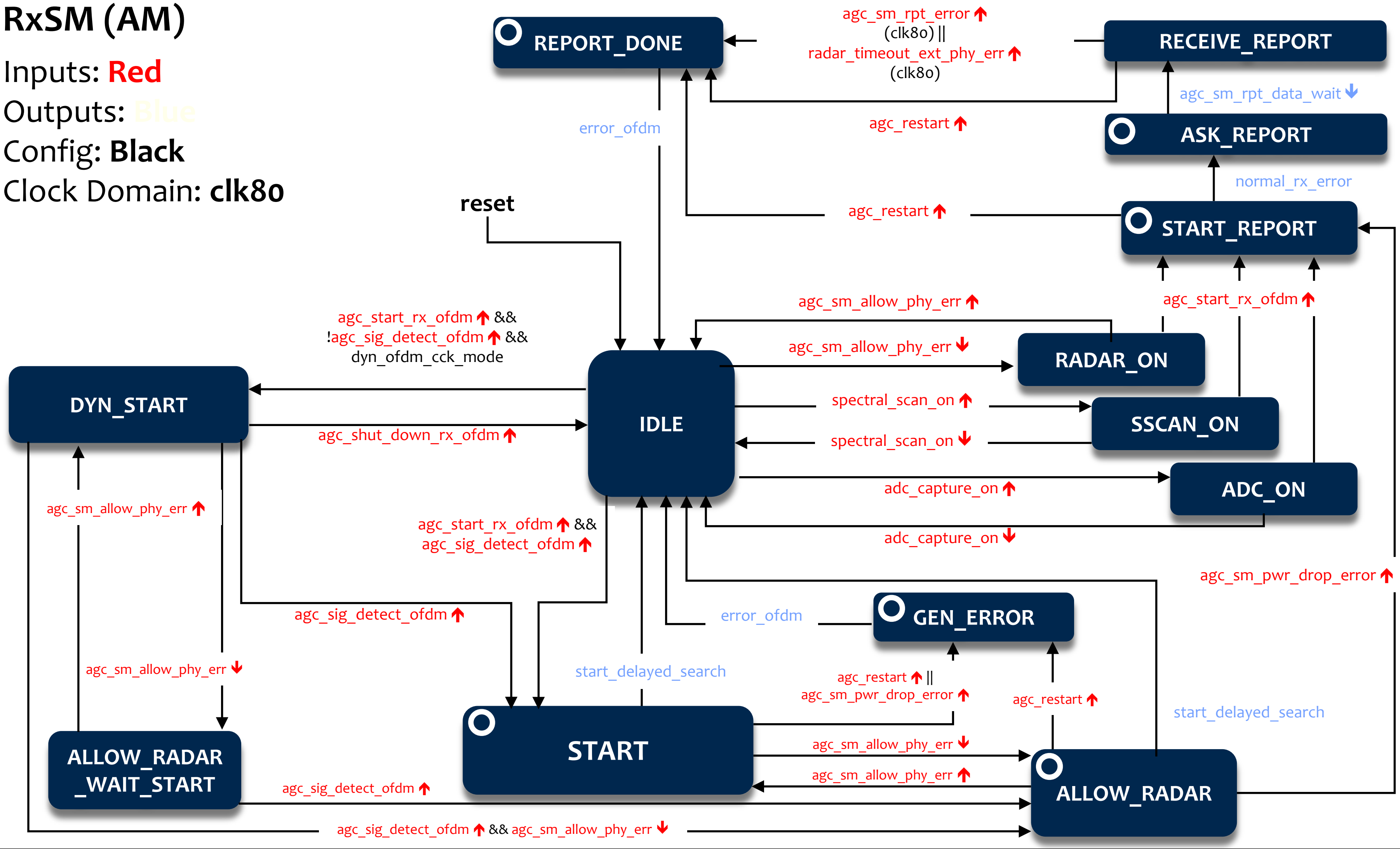
RxSM (AM)

Inputs: **Red**

Outputs: **Blue**

Config: **Black**

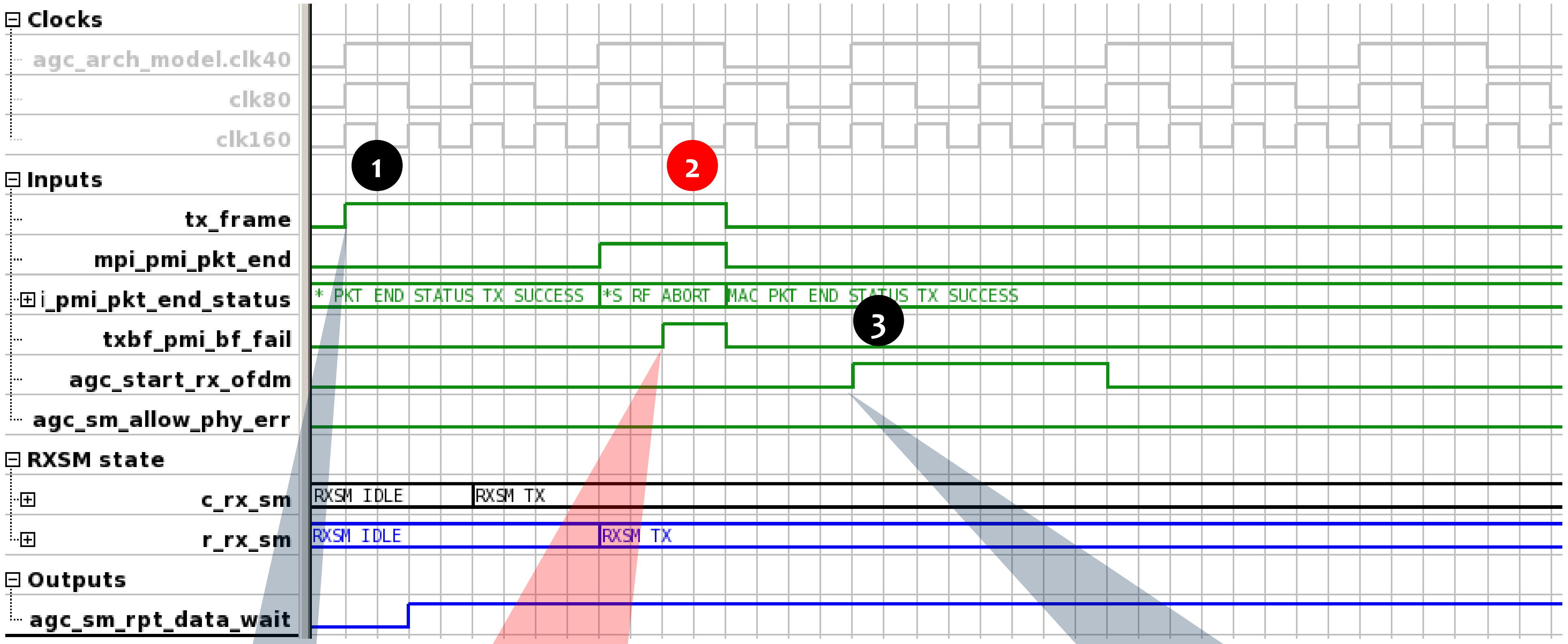
Clock Domain: **clk80**



Step 3: Validate AMs Versus RTL

Results

- Corner-case bug example: Packet end and fail conditions occur simultaneously causing deadlock



TX frame started

mpi_pmi_pkt_end and ***txbf_pmi_bf_fail*** occurred simultaneously

A new RX frame could not be detected since RX_CTRL is stuck in TX state now

Summary

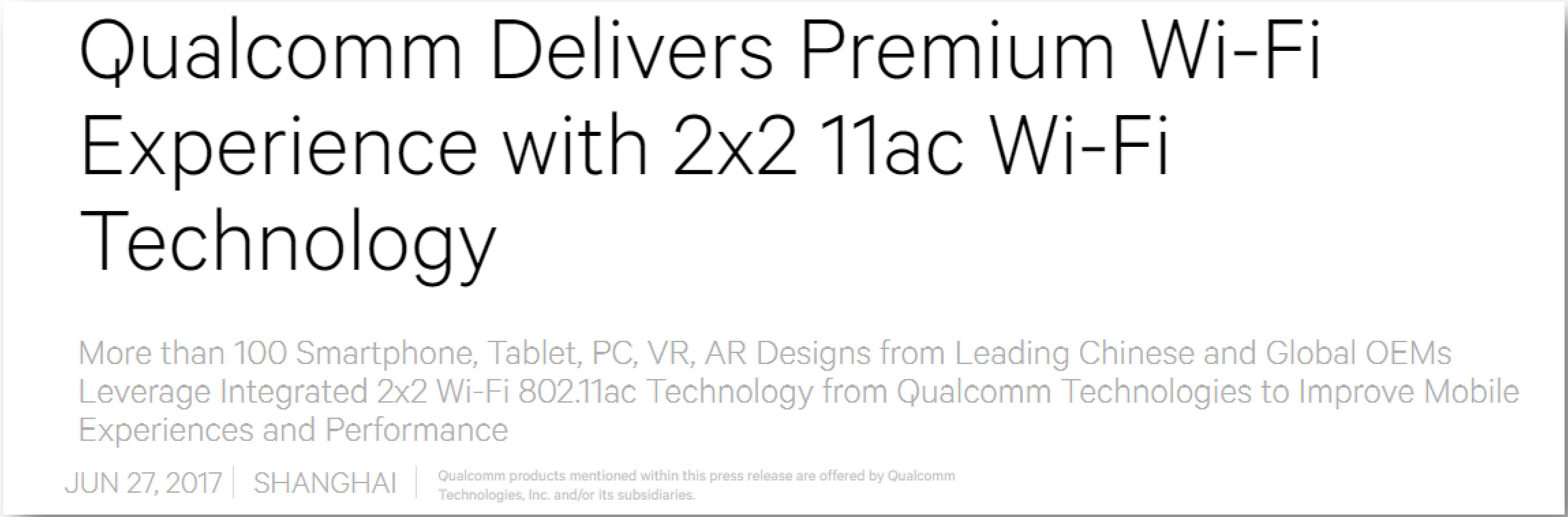
- Introduced a methodology to formally prove system level requirements using architectural models
  - Addressed concerns surrounding adequacy of simulation coverage using formal to exhaustively cover corner cases
  - Addressed issues of formal tools inability to handle high-complexity design by building Architectural Models of RTL blocks
  - Ensured correctness of AMs by verifying them against respective RTL
- Successfully deployed the methodology for a Wireless SoC
  - Proved the absence of deadlocks in the PHY sub-system
- Improved the quality of silicon
  - **Found 9 hard-to-find bugs** which, left undiscovered, could have led to performance issues and/or unplanned chip re-spin

Acknowledgements

- Anshul Jain – Oski Technology, Inc.
- HarGovind Singh – Oski Technology, Inc.

- Cedric Choi – Qualcomm Atheros, Inc.
- Naveed Zaman – Qualcomm Atheros, Inc.

Epilogue



Qualcomm Delivers Premium Wi-Fi Experience with 2x2 11ac Wi-Fi Technology

More than 100 Smartphone, Tablet, PC, VR, AR Designs from Leading Chinese and Global OEMs Leverage Integrated 2x2 Wi-Fi 802.11ac Technology from Qualcomm Technologies to Improve Mobile Experiences and Performance

JUN 27, 2017 | SHANGHAI | Qualcomm products mentioned within this press release are offered by Qualcomm Technologies, Inc. and/or its subsidiaries.