2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION JAPAN

Holistic Verification of Bus Health Monitor in Automotive SoC using BHMVC and ParaHunter

Gaurav Kumar Yadav, Abhisek Hota, Prashantkumar Sonavane,
Samsung Semiconductor India R & D

SAMSUNG

accellera SYSTEMS INITIATIVE

Bus Health Monitor(BHM)

- Automotive SoCs require stringent safety measures
- Fault occurring at one component can result in failure at another component which is called ***Dependent Failure***
- Bus (Interconnect) is backbone of every SoC
- Fault in Bus can lead to potential dependent failures
- Parameter and Registers based configurable **"Bus Health Monitor" (BHM)** IP is introduced to run predefined periodic test patterns to monitor Bus health
- BHM Capable of detecting **90% >** permanent faults in the Bus which is required for Functional Safety ASIL-B Certification of Automotive SoC
- A UVM based **"Bus Health Monitor Verification Component" (BHMVC)** has been developed to support plug and play verification of the BHM instances across SoC

Verification Challenges & Solution

Challenges:

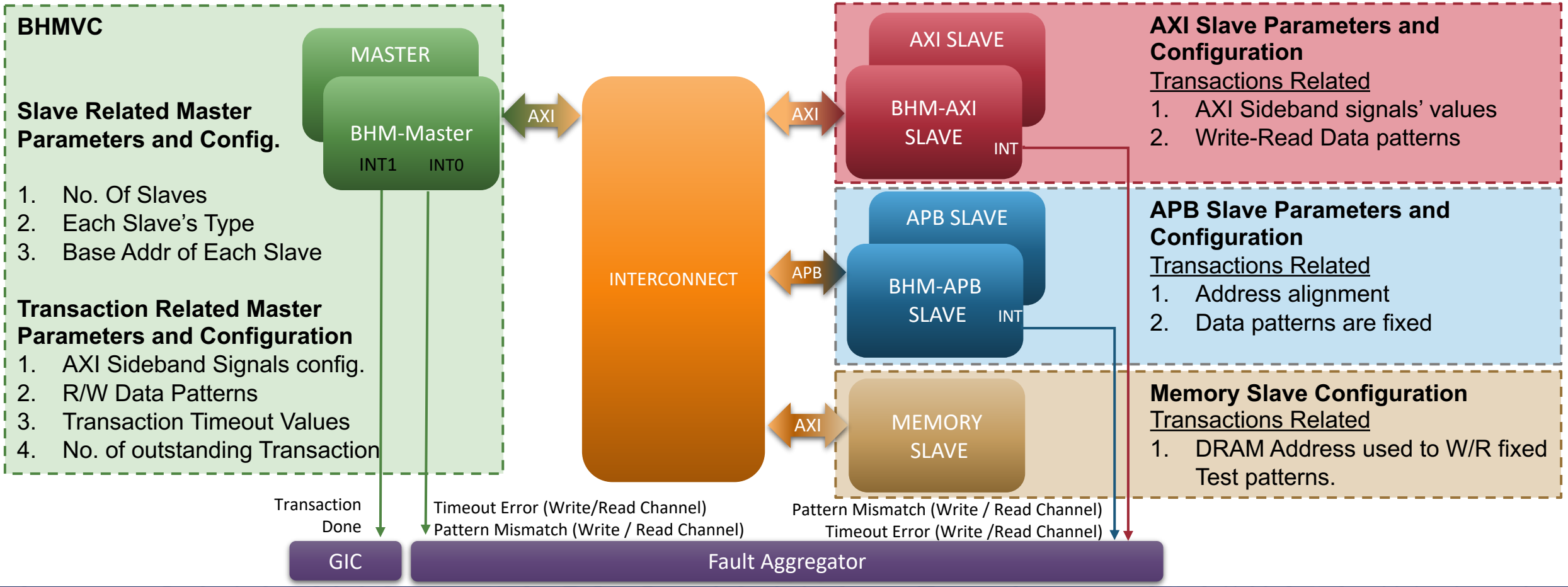
- Verification of all BHM Masters(~15) ↔ Slaves(~40) combinations at SoC
- For all BHM instances total ~800+ Design Parameters to be reviewed/verified
- Considerable use-case understanding and test case development time for each combination. Probable Human-Error.

Solutions:

- Highly customizable and Plug & Play BHMVC for SoC DV
- Exhaustive coverage of all the supported BHM parameters and configurations
- Randomization to explore the corner cases
- Supporting Stress testing to verify BHM operation along with SoC traffic (use-case)
- Along with BHM's functional checks, BHMVC supports BHM's Test of Diagnostic Verification
- Extensive in-Built functional checkers to verify all the correct functionality
- Sanity Check support for fast SoC TB bring-up e.g. clock, reset checks & overall integrity checks

BHMVC usage in SoC Verification

- BHM Master/Slave Modules are instantiated along with each Master/Slave respectively
- UVM based Plug N Play BHMVC is deployed to verify each BHM Master to Slave combination

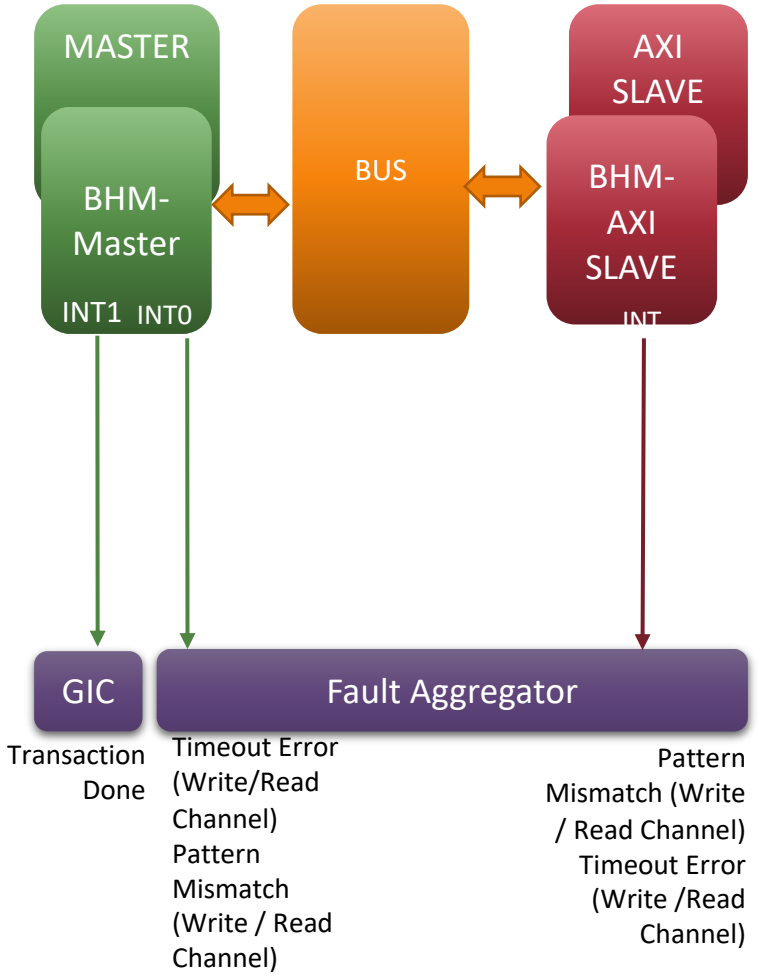


BHM Verification Component(BHMVC)

- BHMVC Plugged-in during initial design integration phase
- Support provided to check output interrupt integrity from all over the blocks to Fault Aggregator and GIC
- It helped in making SOC environment up quickly for BHM verification
- User friendly sequences which user can customize as per vector requirements
- Target Master and slave combinations selection
- Choice of start and end addresses for memory slaves
- Target types selection for 1 to many and many to 1 scenarios
- Based on user selection configuration of master and slave will be done by BHMVC
- Pre-defined data patterns will be transmitted from master to slave ensuring bus health
- Continuous Parallel monitoring of faults/Interrupts using extensive checkers
- In case any mismatch or timeout error occurs fault will be reported to fault aggregator
- In case no fault/error occurs transaction done will be routed to GIC once asserted

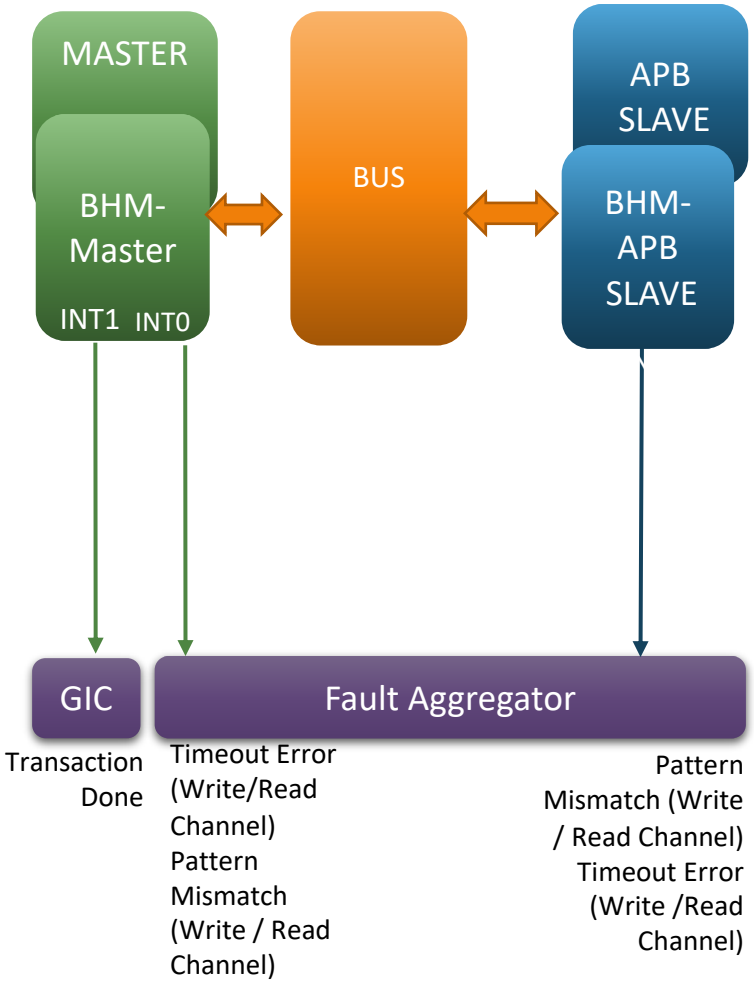
BHM Master to AXI Slave

- After PoR first BHM AXI slave is configured
- AXI sideband signal and data register is programmed what master is going to send
- Slave is initialized by configuring control register which will keep slave ready to receive data from Master
- Then BHM master register is configured with AXI slave address
- Along with timer register and slave type (AXI) are also programmed
- Once Master is initialized it will send predefined data pattern to slave and do write and read operation
- During operation, timeout and data mismatch error registers are also checked at both master and slave end which will assert INT0/INT
- INT0 and INT will be processed to FA for any BUS failure from both master and slave respectively
- INT1 will be sent to GIC once all data operation is completed



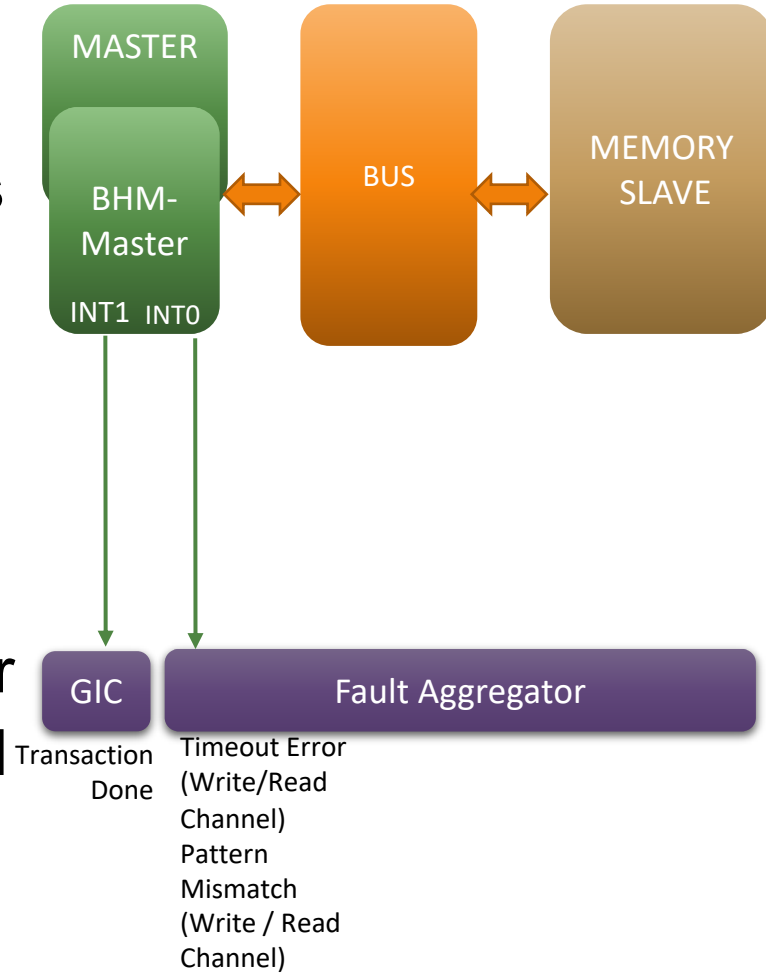
BHM Master to APB Slave

- After PoR first BHM APB slave is configured
- Slave is initialized by configuring control register which will keep slave ready to receive data from Master
- Then BHM master register is configured with APB slave address
- Along with timer register and slave type (APB) are also programmed
- Once Master is initialized it will send predefined data pattern to slave and do write and read operation
- During operation, timeout and data mismatch error registers are also checked at both master and slave end which will assert INT0/INT
- INT0 and INT will be processed to FA for any BUS failure from both master and slave respectively
- INT1 will be sent to GIC once all data operation is completed



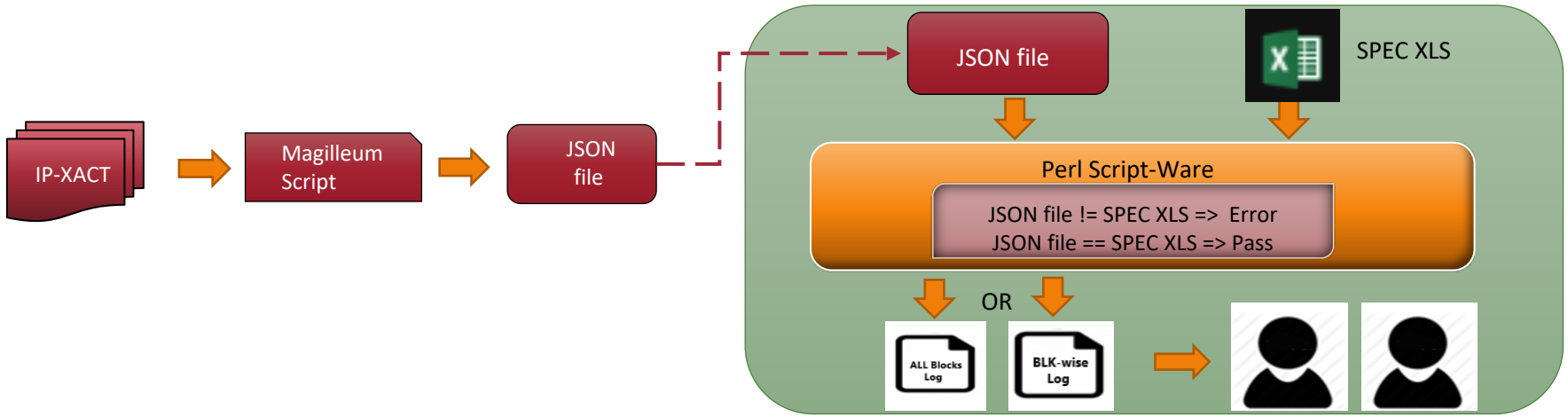
BHM Master to Memory Slave

- After PoR first BHM AXI Master is configured
- Then BHM master register is configured with DRAM address
- Along with timer register and slave type (DRAM) are also programmed
- Once Master is initialized it will send predefined data pattern to DRAM and do write and read operation
- During operation timeout and data mismatch error registers are also checked at master end which will assert INT0
- INT0 will be processed to FA for any BUS failure from master
- INT1 will be sent to GIC once all data operation is completed



ParaHunter

- BHM is highly configurable IP. Which has multiple Parameters (Generics) and supports Register based Configurations
- Depending upon the parameters sub-blocks/registers within BHM will be instantiated
- Though the correct functionality is verified by BHMVC, any wrong parameter value especially higher parameter values (e.g. NO_OF_SLAVES_SUPPORTED is 5 instead of 4) causing additional logic instantiation within BHM won't be detected by BHMVC
- Hence a Script-Ware named "ParaHunter" has been developed to verify the BHM IPs parameters against the Specification



Conclusion

- Major outcomes from BHMVC and ParaHunter :
  - More than **300** tests added to verify all BHM combinations using this plug and play solution, with TAT of **2-3 weeks** instead of 4 man-months
  - Parameters **(~800**) verification reduced from 2 man-weeks to **5-10 minutes**
  - **~500 µm²** area saved due to wrong sub-module instances
- Design Architecture Impact and Critical bug findings :
  - **Architectural Impact**:-
    - **Bus latency**: Timeout counter width increased for masters to cover worst case
    - **Power** :Clock Gating support added to request clock for all masters
    - **Higher Throughput:** Interconnects outstanding transaction handling capacity increased
  - Slave address alignment bug fixed for all masters **(>10 Blocks**)
  - Default value for periodic transaction interval updated for all masters **(>10 Blocks**)