Gen2 HMC BFM Overview

Aug. 25, 2014

Features

Language - Class based SystemVerilog

Simulators - Big 3

- Synopsys
- Cadence
- Mentor

Environment

- Limiting UVM in the test-bench only. The HMC BFM module itself has only SytemVerilog features.

Interface

- Serial Interface – yes.
- Parallel interface – 128 Bytes
- Parameterized number of links – num_links_c

Debug features

- Default: print out protocol violations and illegal packets detected.
- Additional features: print out every transaction that is received along with responses.

Test bench

- Provide a simple test-bench that shows how to hook it up and get it running independently without specific host controller attached to it.

Encryption

- Not encrypted

BFM Internal Features and Rollout Details

Feature	Status – 10/21/2013
Support link rates: 10/12.5/15Gbps	available
Support single link to the full memory space	available
Support multi-link to the full memory space	available
Supports cube size: 4GB	available
Support full and half-width link modes	available
Support Power on Initialization sequence generation and training sequence check	available
Public register initialization through configuration files.	available
tRST, tRSP1, tRSP2 and Nulls checks	available
Support Tokens based flow control	available
Token return packet - TRET	available
Support scrambling and descrambling	available
Support parallel interface (bypass initialization)	available
Support 16Bytes granularity: 16B to 128B	available
Supports all HMC request and response packets	available
Public register access through WRITE/READ request commands	Available
Register configuration through configuration files	Available
Request packet header fields check	available
Including ADRS to make sure valid vault, bank and DRAM addresses.	available
Request packet tail fields check	available
Support Zero field check for PRET, IRTRY request packets: FRP, RTC and SEQ	available
Support controlled request/response packets (including flow packets) corruption	available
Support Random response corruptions/bit toggles : LEN, CRC, SEQ, DINV, ERRSTAT	available
Support controlled poisoned /error response packet generation	available
Support response open-loop mode	available
Support cube chaining	Available – see readme file for known limitations
Support lane reversal and lane polarity inversion	available
Support automatic link retry sequence to random request packets corruption	Available
Power management limited only to verification purpose	Available
Support warm reset to re-initialize state machines and tokens	Available
Supports retraining and recovery	Availalble
Responses with configurable random delays	Available
Link flow latency	Available

Limitations

No HSS circuit/register

Timing delays and checks

- What is modeled:
  - The link initialization, retraining, power state transition timings are checked
  - Link layer flow control
- What is not modeled:
  - DRAM timings are not modeled.
  - Internal logic delays caused by internal cross bar and vault controllers are not modeled.

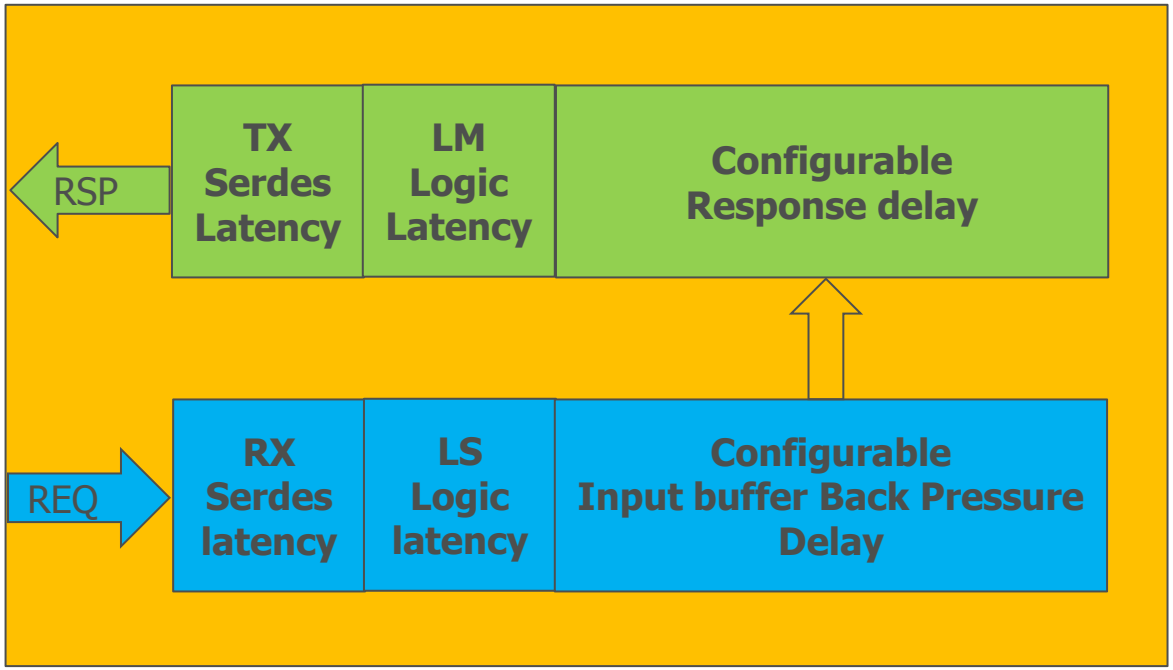
Protocol accurate only.

- Not cycle accurate
- Response packet order is not modeled. Responses can be either in order or with configurable random delays.

No performance considerations

No JTAG/I2C

BFM Latency Modeling



RRP Latencies

Description

- RRP latency covers packets travel and process time in both direction as the “Retry Pointer Loop Time” defined in the spec.
- RRP can be embedded in PRETs if there is no normal transactional response packet. When embedded in normal transactional response packet, the size of packet is determined by the data pattern.
- Input buffer back pressure has NO impact the RRP latencies. The RTC latency configuration is separate from this.
- Response delay, which can be configured as random delays by ***cfg_rsp_mean*** and ***cfg_rsp_std_dev*** in BFM, represents all the backend latencies It changes the order of the response packets carrying the RRPs and creates scenarios that PRET packets need to be generated. This make the RRP returns and their latencies more realistic.

Boundary Conditions

- Lower bound: **SerDes latencies (RX+TX) + Link Layer latencies (LS +LM)** -- Both of them are referring to the latencies of 16B packets, including 16B packet travel times in both RX and TX directions FRP process time from a 16B packet and RRP insertion to a 16B packet in LS and LM respectively
- Upper bound: **SerDes latencies (RX +TX) + Link Layer latencies (LS+LM)** --Both of them are referring to the latencies of 128B packets, including 128B packet travel times in both RX and TX directions FRP process time from a 128B packet and RRP insertion to a 128B packet in LS and LM respectively.

RTC Latency

Description

- RTC latency covers packet travel and process time in both directions.
- The difference between RTC and RRP latency is, RTC is also affected by the input buffer back pressure delay, which is configurable by ***cfg_rtc_mean*** and ***cfg_rtc_std_dev*** in BFM.
- RTC can be embedded in TRETs if there is no normal transactional response packet. When embedded in normal transactional response packet, the size of packet is determined by the data pattern.
- Response delay, which can be configured as random delays by ***cfg_rsp_mean*** and ***cfg_rsp_std_dev*** in BFM, represents all the backend latencies. It changes the order of the response packets carrying the RTCs and creates scenarios that TRET packets need to be generated. This make the RTC returns and their latencies more realistic.

Boundary Conditions

- Lower bound: **SerDes latencies (RX+TX) + Link Layer latencies (LS+LM).** Both of them are referring to the latencies of 16B packet, including 16B packet travel time in TX direction and RTC insertion to a 16B packet in LM.
- Upper bound: **SerDes latencies (RX+TX) + Link Layer latencies (LS+LM) + Input buffer back pressure latency**. Both of them are referring to the latencies of 128B packet, including 128B packet travel time in TX direction and RTC insertion to a 128B packet in LM.

Known Issues

Warm Reset through Mode Write is not supported.

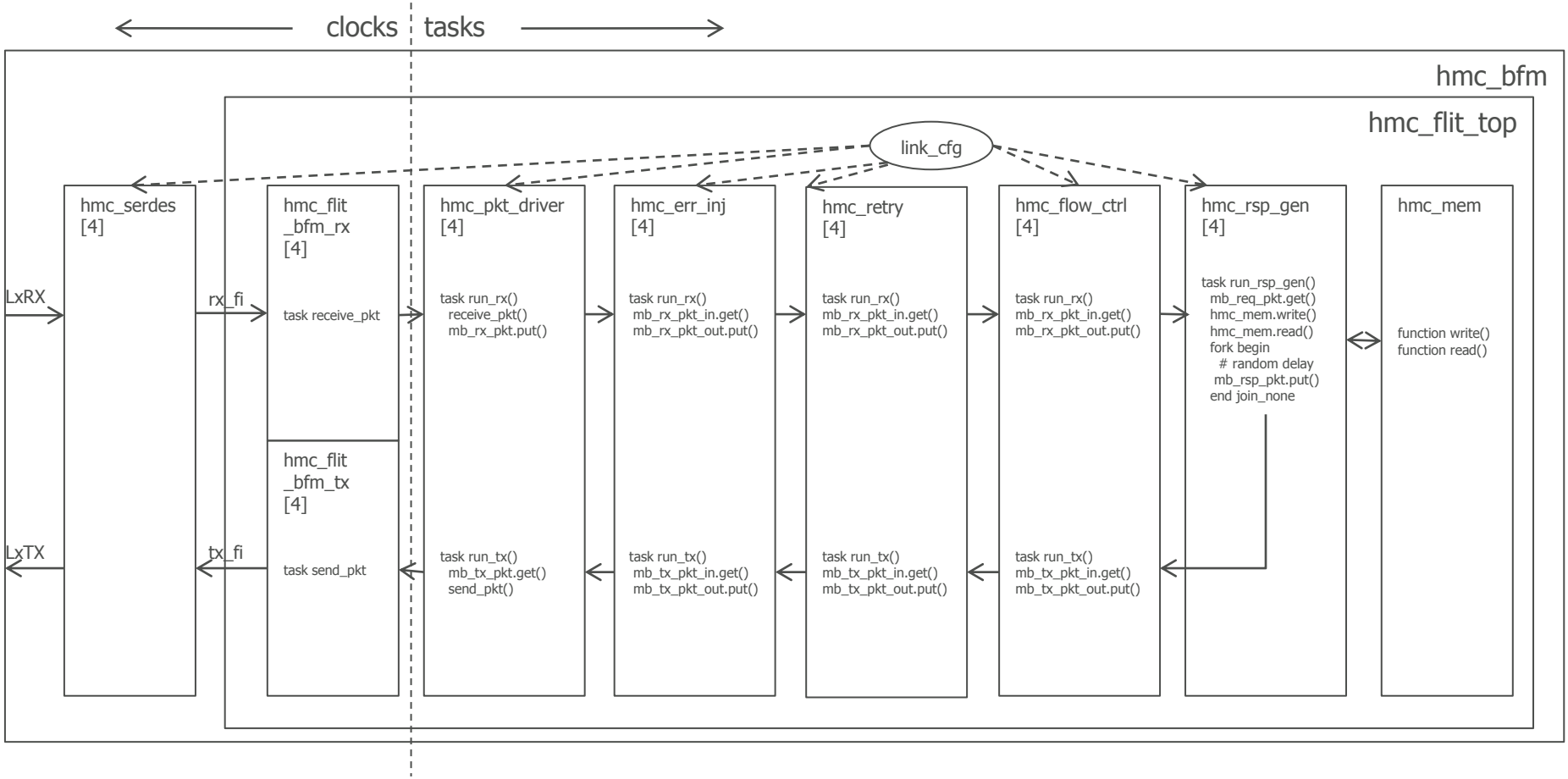
- It is not reliable because when a HMC device needs to be warm reset, it should already have some internal critical errors.
- A note will be added to Micron's datasheet to clarify this.
- Warm Reset is only supported through I2C and JTAG.

The request and response monitors in the BFM are on the flit interface, instead of the serial interface. For simulations with serial interface, there are small latency discrepancies between the time stamps shown in BFM vs. actual time on the Serdes. For example, first few TRET responses may show before the LINKACTIVE state.

Starting from r26472, an update is made to match with RTL behavior (Datasheet update will follow) on "requests from a single external serial link to the same vault/bank address". For the requests from a single external serial link to the same vault/bank address, the executions are in order and the read to read response order is maintained.

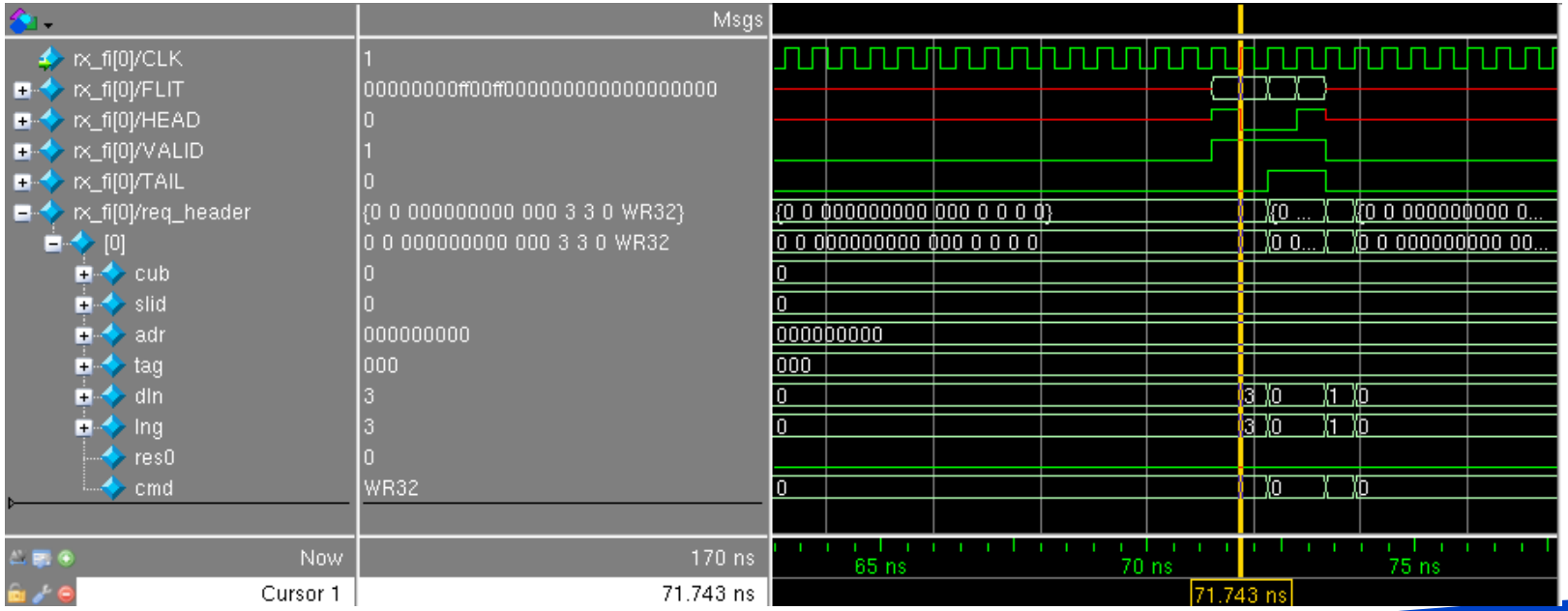
Architecture

Block Diagram



Debugging

- BFM components are static
  - signals can be logged and added to waveform
- FLIT interface shows deserialized, descrambled, uninverted, packets
  - HEAD, VALID, TAIL bits show packet boundaries
- example wave.do file provided for Questa debugging
- Programmable verbosity to print from each component (cfg_info_msg)

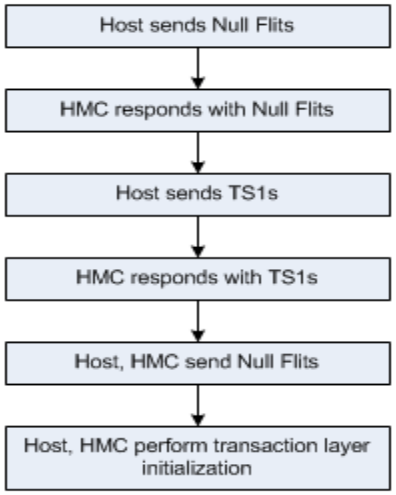


Configuration

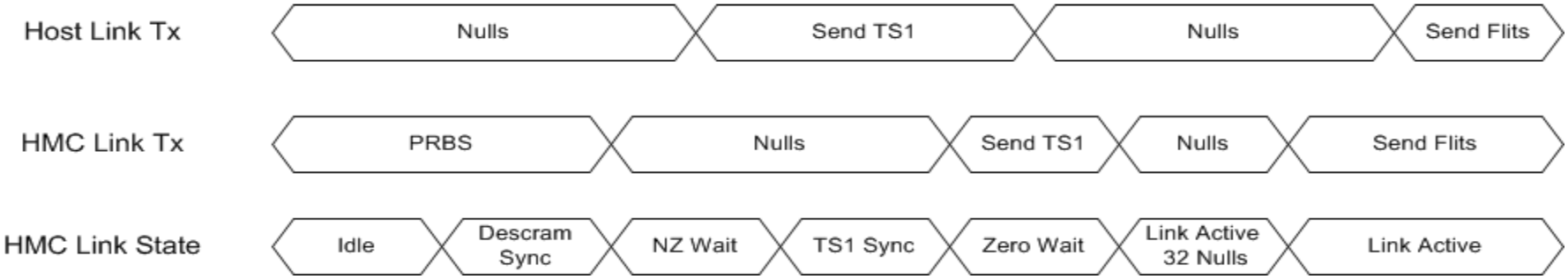
- Parameters
  - num_links_c = number of links on the HMC BFM
  - num_hmc_c = number of HMC devices connected in the fabric
- Configuration Objects
  - can be randomized
  - cls_link_cfg: configuration settings for each link
  - cls_cube_cfg: configuration settings for each HMC device
- Configuration methods
  - `hmc_dut.set_config: passes cls_link_cfg to each interface
  - `hmc_dut.set_cube_cfg: passes cls_cube_cfg to each interface
- Configuration files
  - Example configuration files and methods are provided in the cfg folder

Link Initialization

HMC Link Initialization Flow Chart

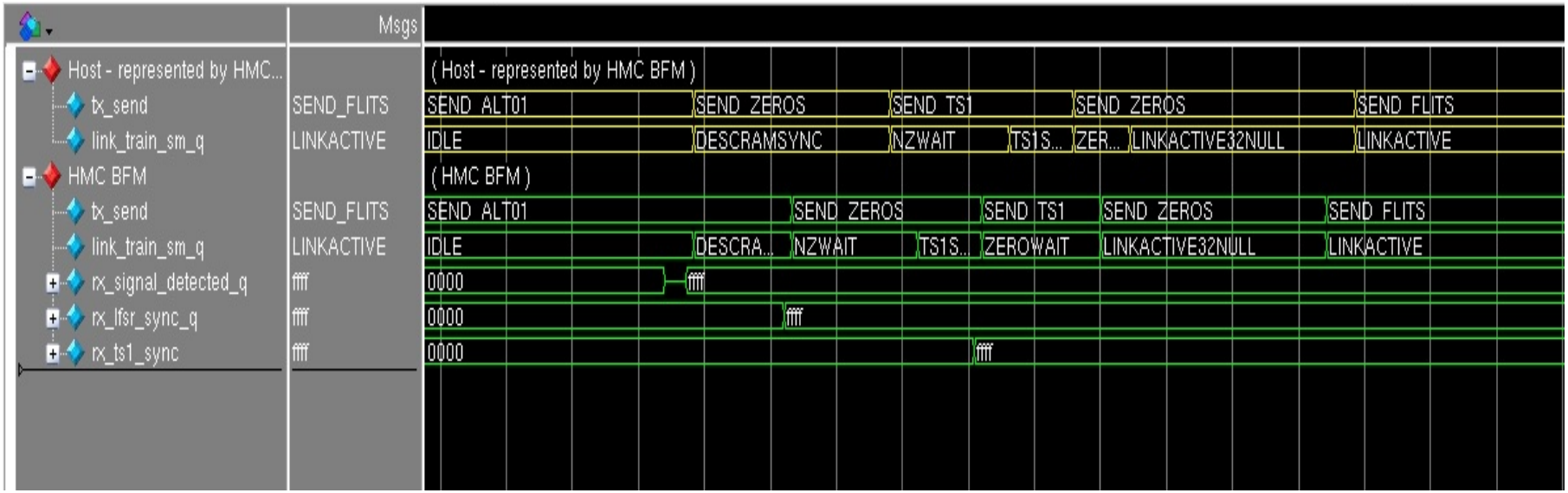


HMC BFM Link States and Tx Data



Link Training

- Host represented by HMC BFM
  - Yellow traces
- HMC BFM
  - Green traces
  - Rx_signal_detected_q trigger to tansistion from Idle to Descrambler Sync
  - Rx_lfsr_sync_q trigger to transition from Descrambler Sync to Non Zero Wait
  - Rx_ts1_sync trigger to transistion ftrom TS1 Sync to Zeroes Wait



HMC BFM Components

Serialization: hmc_serdes

FLIT interface: hmc_flit_bfm

Driver: hmc_pkt_driver

Error Injection: hmc_err_inj

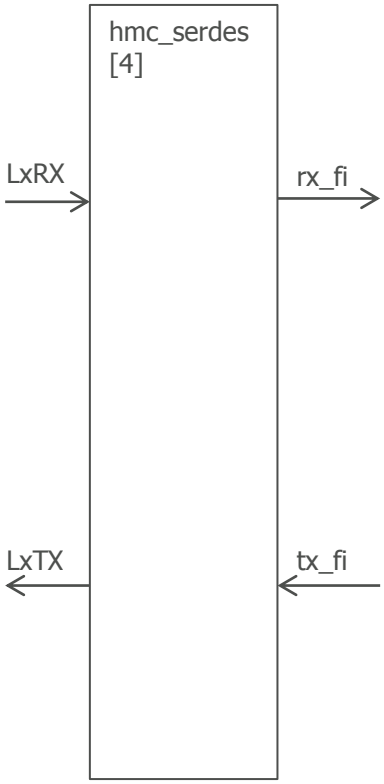
Retry: hmc_retry

Flow Control: hmc_flow_ctrl

Response Generator: hmc_rsp_gen

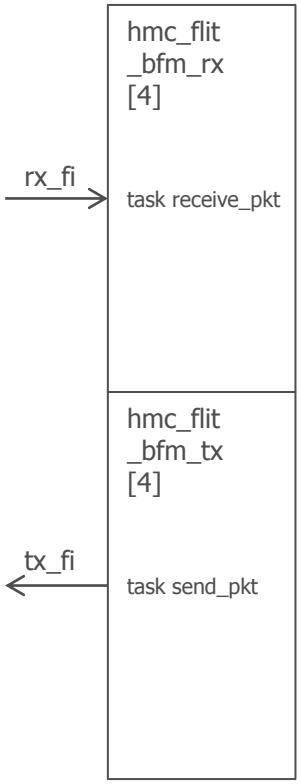
Memory: hmc_mem

Serialization: hmc_serdes



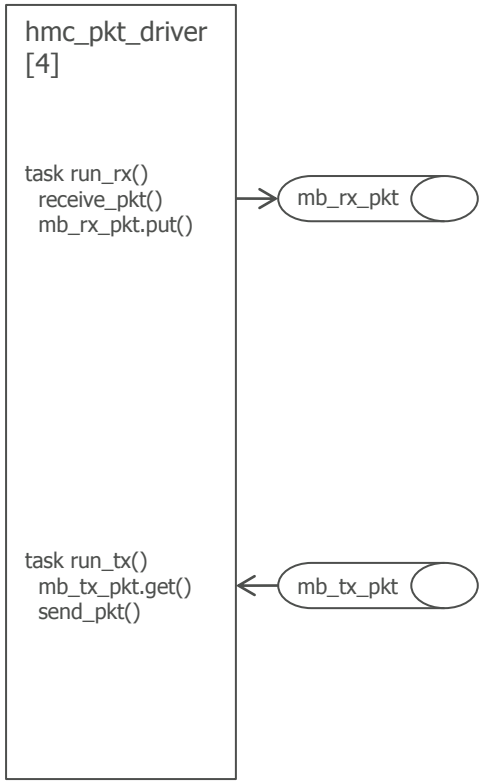
- Generates TX and RX UI and FLIT clocks from REFCLK
  - Independent clock speeds and configuration per RX, TX and link
- Deserializes 16 bit LxRXP onto 128 bit FLIT interface, and drives associated HEAD, VALID and TAIL signals
- Serializes 128 bit FLIT interface onto 16 bit LxTXP
- configuration
  - training
  - scrambling/descrambling
  - half rate
  - lane reversal and polarity inversion

FLIT interface: hmc_flit_bfm



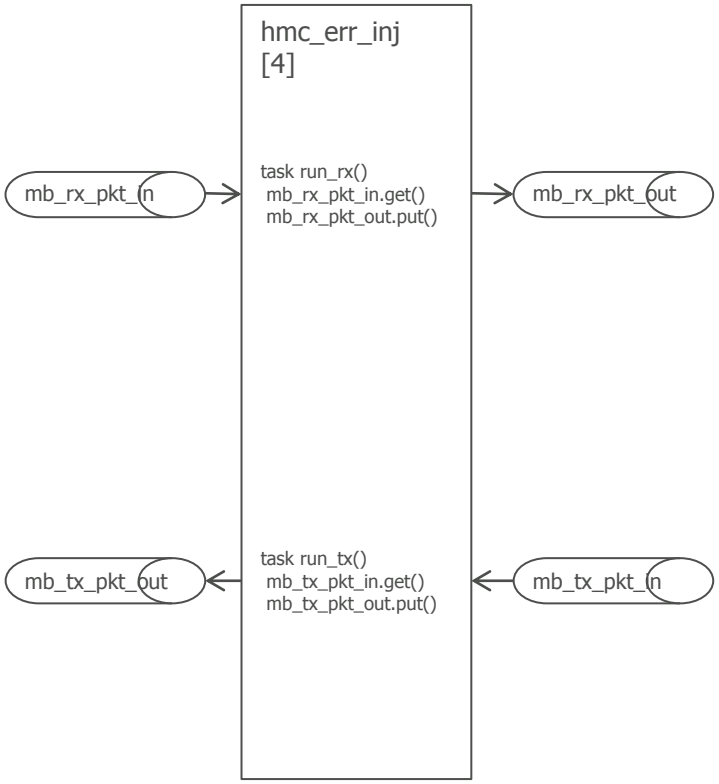
- Defines tasks that drive the FLIT interface
  - send_pkt, receive_pkt, monitor_pkt
- 2 instances in the hmc_bfm, one for TX direction and one for RX direction
- configuration
  - none

Driver: hmc_pkt_driver



- Converts between packets and tasks in hmc_flit_bfm
- run_rx task
  - receives header, data, tail from hmc_flit_bfm
  - creates a new request or response packet
  - puts the packet object into the rx mailbox
- run_tx task
  - gets a packet object from the tx mailbox
  - sends header, data, tail to hmc_flit_bfm
- configuration
  - cfg_host_link: host link or pass-thru link

Error Injection: hmc_err_inj



- Provides manual or randomized error injection
- Can inject errors into the Request or Response streams
  - Request errors can be crc, duplicate length, sequence number, and poison pkt
  - Response errors can be crc, duplicate length, sequence number, data invalid, poison packet, or error packet.
- configuration
  - Queues to store tags for manual error injection
    - rsp_tag_q, req_tag_q

Manual Error Injection

Manual error injection is tag-based. A user can specify which tag to inject an error on and the type of error(s).

- User must create an error packet with the expected tag and push into the request or response queue.
  - Queues are searched for tag matches in the request and response direction
  - Once a tag match is found, the error is injected and removed from the queue
- There are two queues per link, one for request packets, and one for response packets.
  - <path to hmc_err_inj>.req_tag_q.push_back(<req_pkt_err object>);
  - <path to hmc_err_inj>.rsp_tag_q.push_back(<rsp_pkt_err object>);

Randomized error injection is percentage based. A user can specify the likelihood of a request or response error.

- This is the only method for injecting errors into FLOW control packets (PRET, IRTRY, NULL)

See hmc_manual_err test for example usage

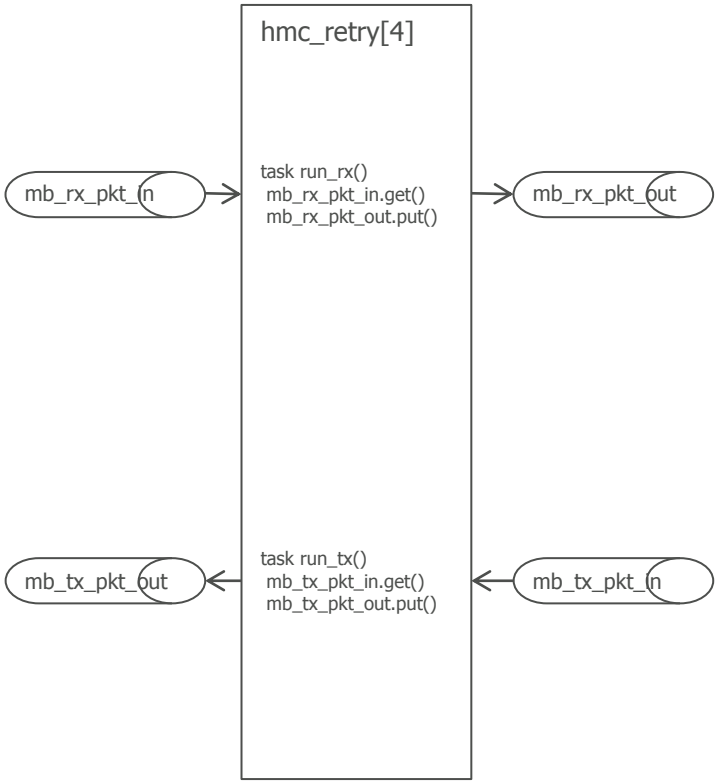
Randomized Error Injection

Randomized error injection is percentage based. A user can specify the likelihood of a request or response error.

- This is the only method for injecting errors into FLOW control packets (PRET, IRTRY, NULL)
  - Error injection rates are defined in the link_cfg object
    - rsp_dln, rsp_crc, rsp_seq, rsp_dinv, rsp_errstat, rsp_poison, rsp_err
    - req_dln, req_crc, req_seq, req_poison

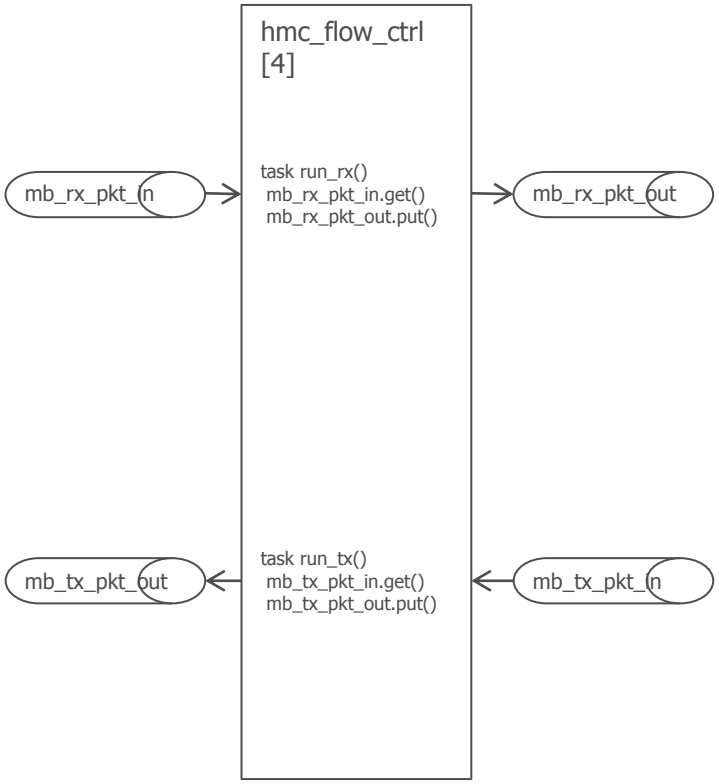
A configuration file (hmc_rand_cfg) randomizes the error injection rates.

Retry: hmc_retry



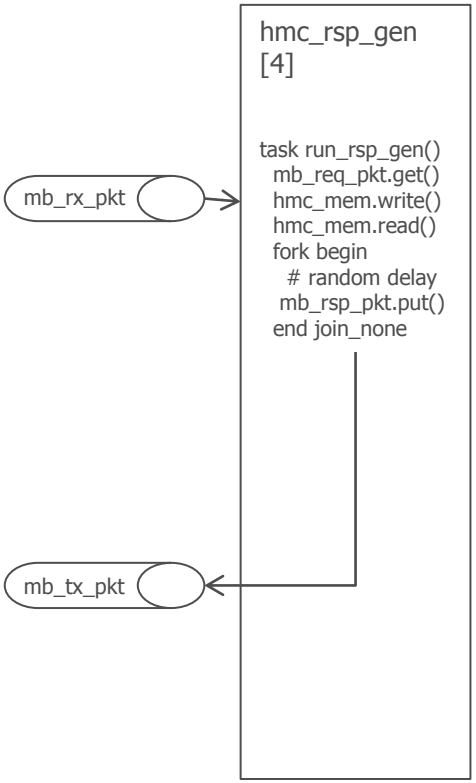
- implements retry state machine
- enters Error Abort Mode on CRC, LEN/DLN mismatch or SEQ error
- stores outgoing packets into retry buffer
- extracts FRP and RRP from incoming packets
- generates FRP and embeds FRP and RRP into outgoing packets
- generates PRET packets when idle
- filters out NULL and PRET packets from downstream devices
- configuration
  - cfg_retry_enb: retry enable
  - cfg_retry_limit: retry limit
  - cfg_retry_timeout: retry timeout
  - cfg_init_retry_txcnt: Number of IRTRY packets to transmit during retry sequence
  - cfg_init_retry_rxcnt: Number of IRTRY packets to receive during retry sequence

Flow Control: hmc_flow_ctrl



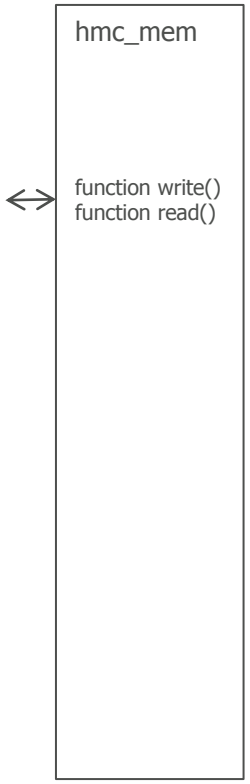
- stores a queue of outgoing commands
- extracts RTC from request packets
- embeds RTC into response packets
- generates TRET packets when queue is empty
- filters out TRET packets from downstream devices
- blocks TX stream when token count = 0
- configuration
  - cfg_tx_pkt_q_size: size of link outgoing command queue
  - cfg_tx_tokens: token count
  - cfg_rsp_open_loop: response open loop mode
  - cfg_tail_rtc_dsbl: disables RTC embedding and TRET generation

Response Generator: hmc_rsp_gen



- Responds to any requests that match the Cube ID
- run_rsp_gen task
  - gets a request packet from req mailbox
  - sends write write and read commands to hmc_mem
  - creates a new response packet
  - waits a random amount of simulation time
  - puts the response into the rsp mailbox
- configuration
  - cfg_cube_id: host link or pass-thru link
  - cfg_lat_avg: average response delay
  - cfg_lat_stddev: standard devation of response delay

Memory: hmc_mem



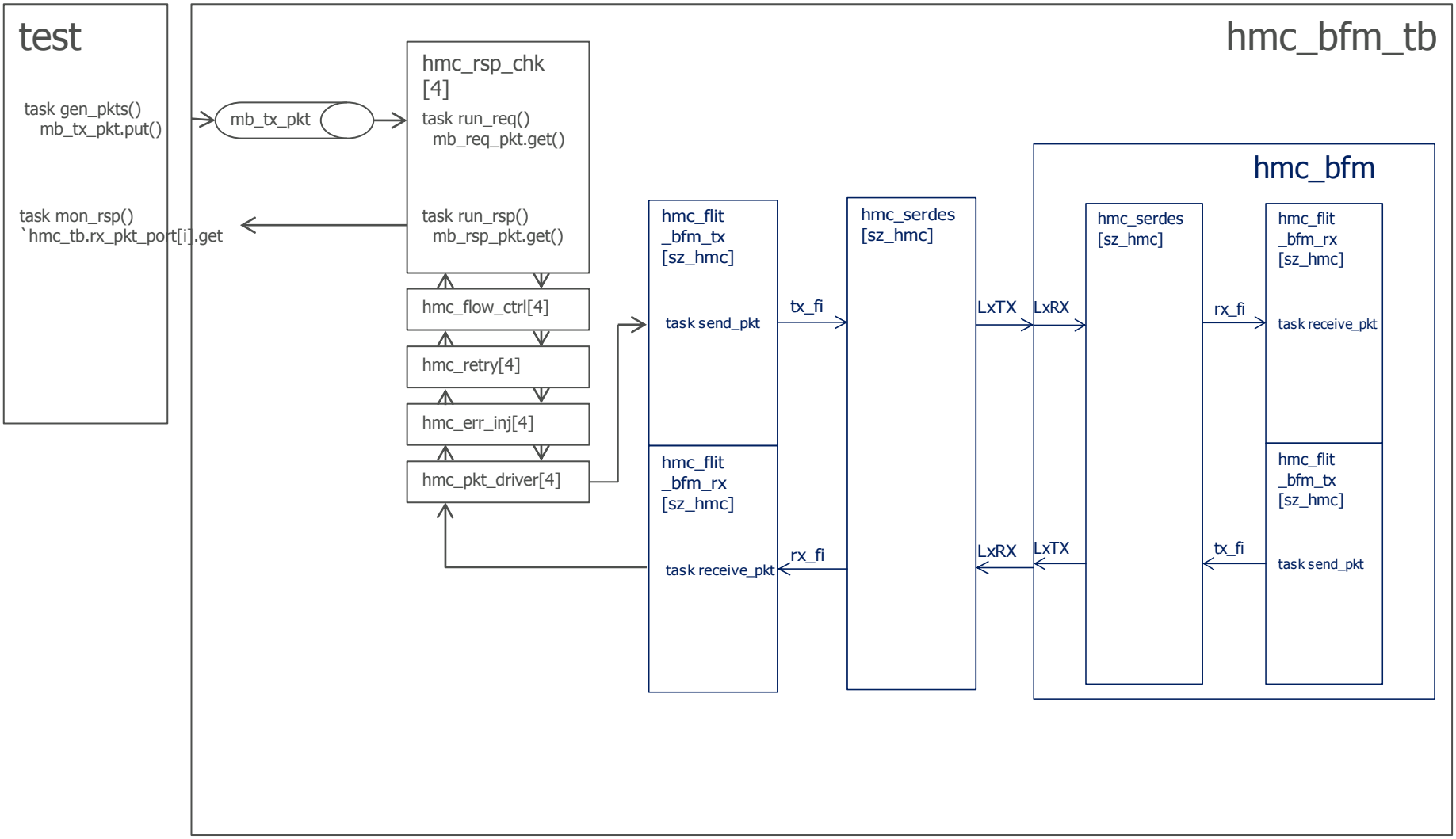
- Defines functions that read and write memory
- One object with shared access from all links
- Configuration register array is a scratchpad for the MD_WR and MD_RD commands
- configuration
  - cfg_block_size: 32, 64 or 128B block sizes
  - cfg_cube_size: 2GB or 4GB cube size

Test Benches

2 example test benches are provided

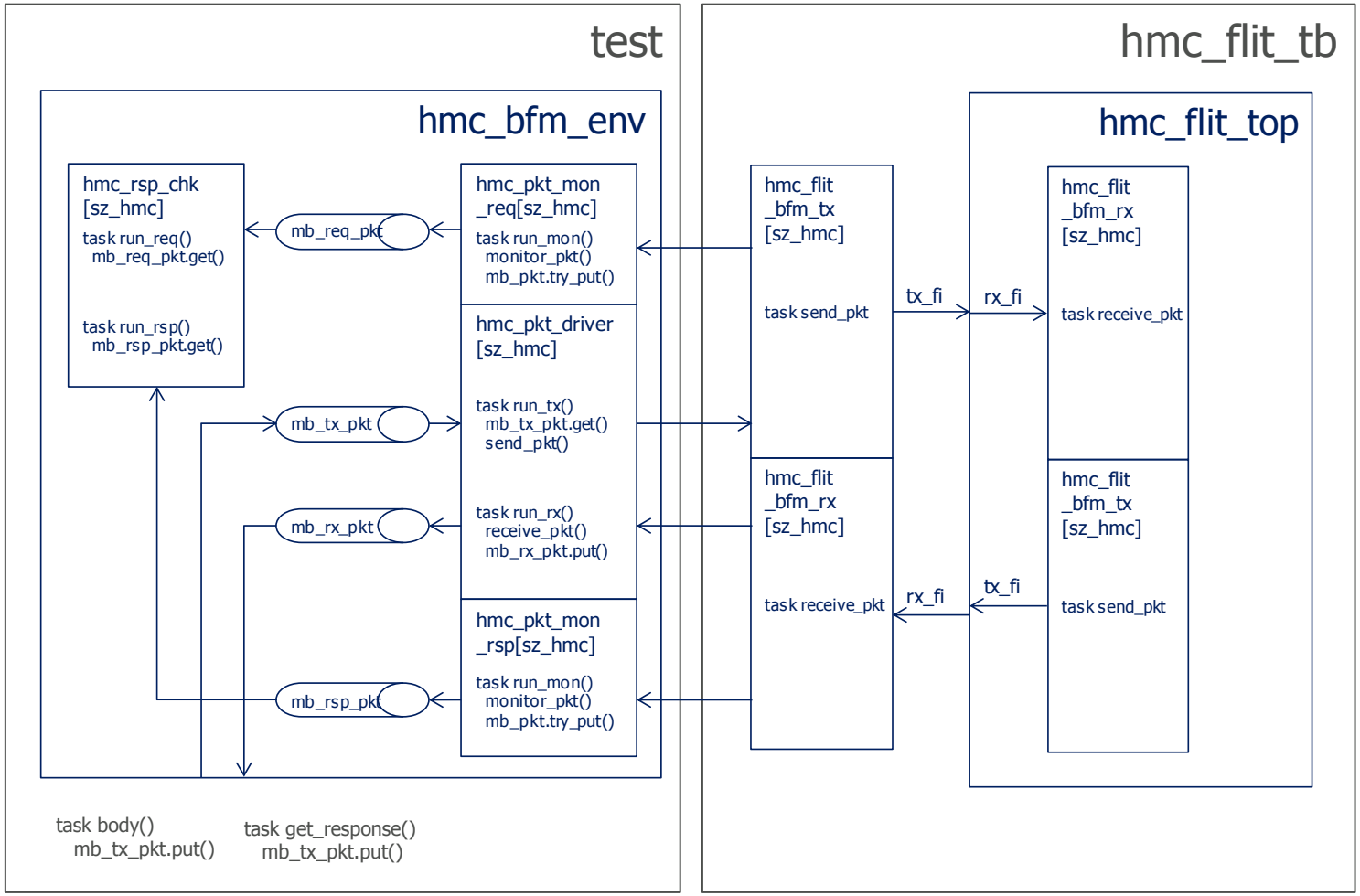
- hmc_bfm_tb: provides link training, serialization, and power state management
  - slower, detailed implementation
- hmc_flit_tb: bypasses link serialization and runs wider FLIT interface.
  - high throughput, easier to debug

Testbench: hmc_bfm_tb



Testbench: hmc_flit_tb

- Same test can run without the serialization/deserialization in hmc_serdes




Micron®
Focused on Memory | Engineered for your Innovation