2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION TAIWAN

Nailing Down the Debug Complexities of Complex Device Inter-Connect's Signal Pattern

Krunal Kapadiya, Sr. Principal Software Engineer

Shyam Sharma, Sr. Architect

Rahil Jha, Principal Software Engineer

cadence®

accellera SYSTEMS INITIATIVE

General Overview

- Operating speed also evolves by many folds with every new generation of devices. The table below shows an example of DDR evolution over generations

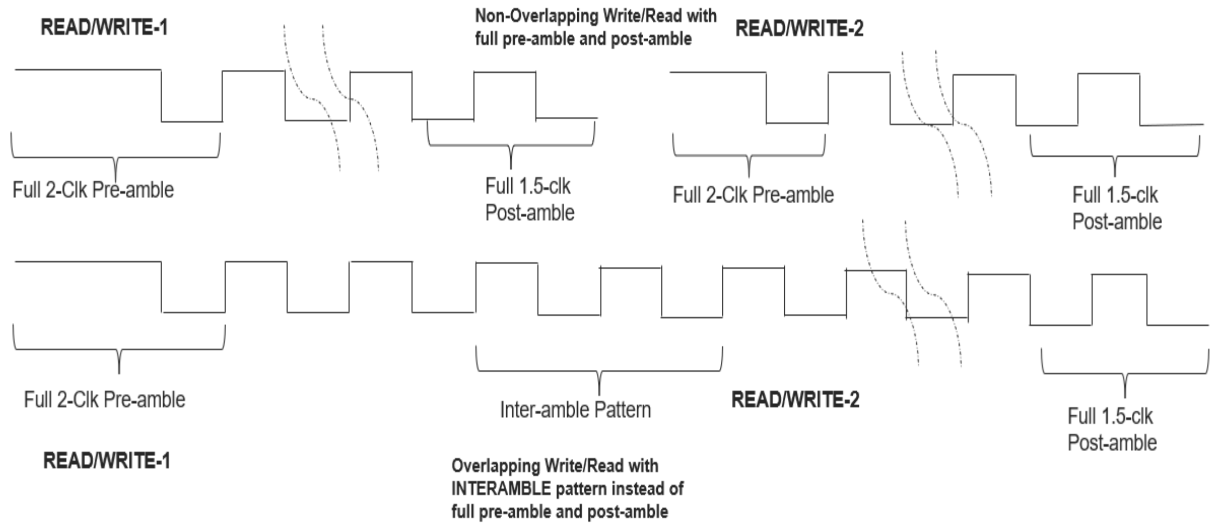
Generation of DDR	Operating Speed
DDR3	**800-2133 MT/s**
DDR4	**1600-3200 MT/s**
DDR5	**3200-8800 MT/s (up to 9200 MT/s in future)**

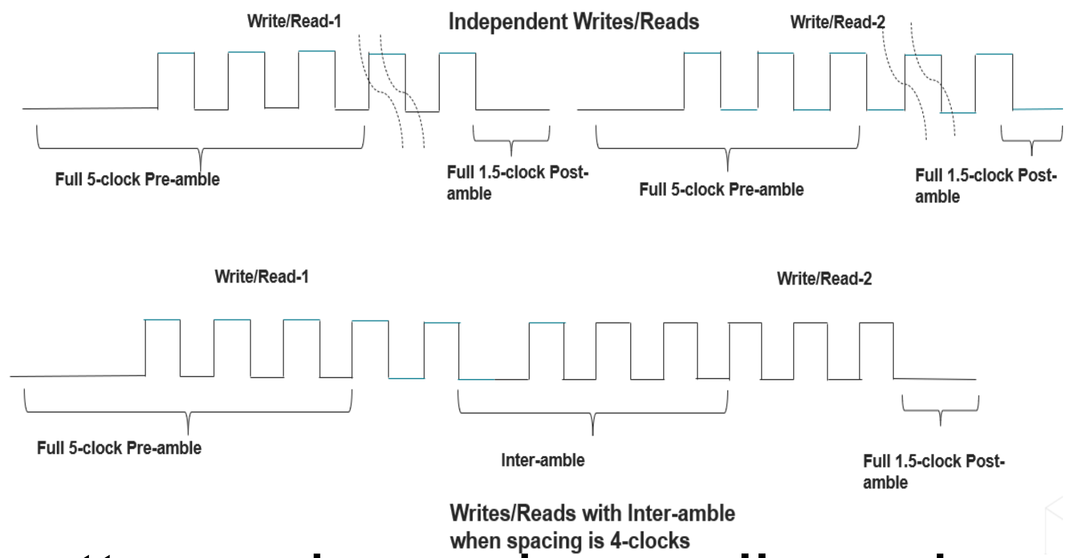
- An increase in operating speed comes at the cost of data integrity issues. To tackle this, especially for memory protocols, predefined signal patterns are defined at the input and output ports to avoid transmission and link errors
- With every new generation of memory, these signal patterns also evolve to the next level of complexities

Signal Pattern Types and Deep Dive Into DDR5 Specific Challenge

- Almost every memory device uses one or other signal patterns. The most widely used patterns are pre-amble and post-amble. For bus protocols, handshaking signals form either deterministic or non-deterministic patterns.
- For DDR5, widely recognized signal patterns are pre-amble, post-amble, and inter-amble. They are deterministic in nature
- Inter-amble: This signal pattern is formed between nearby write or read commands whose full pre-amble and post-amble cannot be accommodated due to spacing constraints. Hence, partial or full pre-amble and post-amble get truncated to form a new pattern called inter-amble
- Thumb rule: Toggles take precedence over static

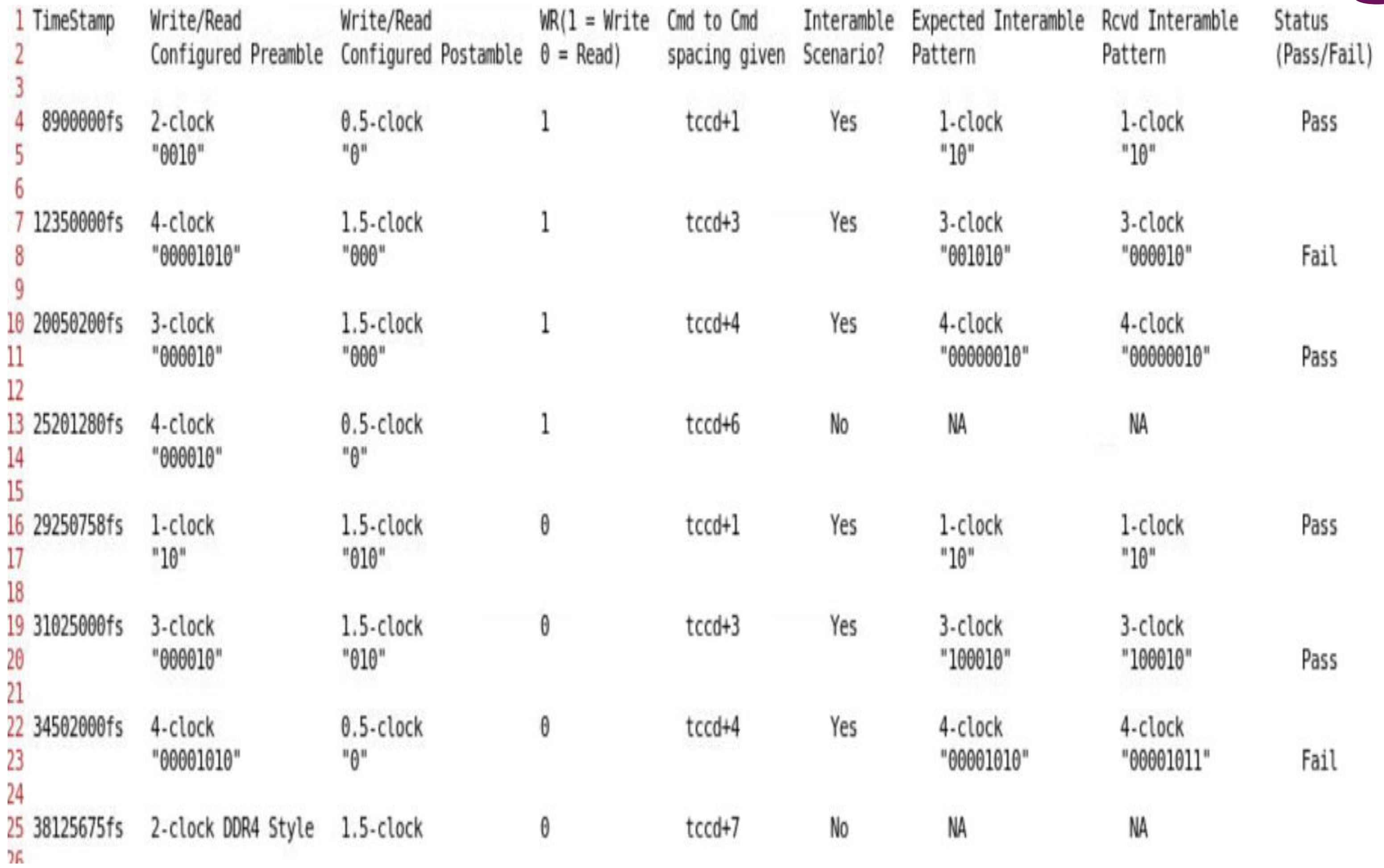
DDR5 Pre-Amble, Post-Amble and Inter-Amble Examples





- The number of possible inter-amble patterns depends on allowed pre-amble and post-amble patterns and allowed spacing between two commands
- Considering these factors, for DDR5, for writes, total possible inter-amble patterns are 30 and total possible read inter-amble patterns are 50

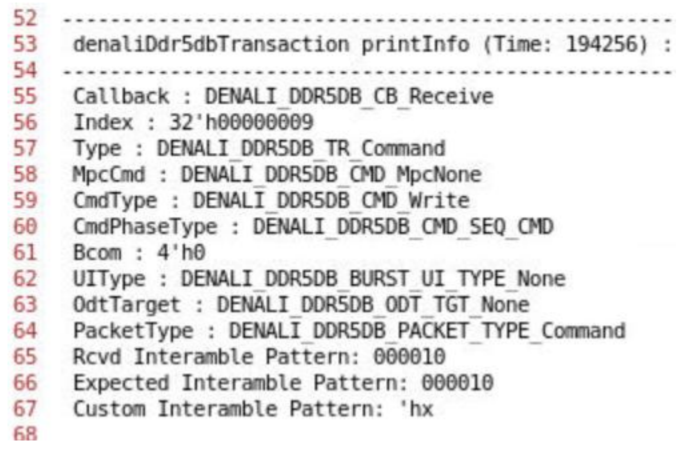
Enabling Smart Log and Sneak Peek into Fields of Smart Log



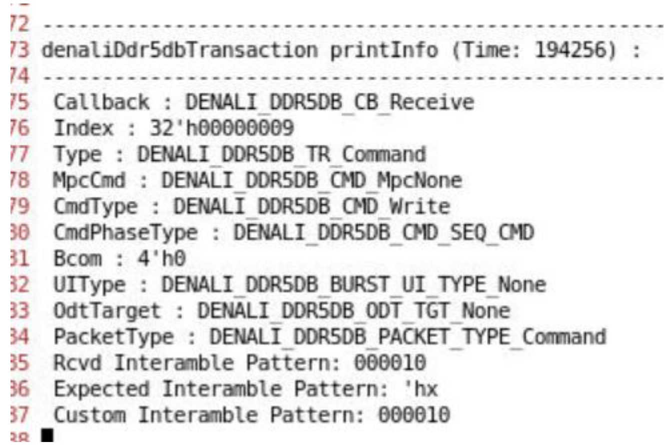
TimeStamp	Write/Read Configured Preamble	Write/Read Configured Postamble	WR(1 = Write 0 = Read)	Cmd to Cmd spacing given	Interamble Scenario?	Expected Interamble Pattern	Rcvd Interamble Pattern	Status (Pass/Fail)
8900000fs	2-clock "0010"	0.5-clock "0"	1	tccd+1	Yes	1-clock "10"	1-clock "10"	Pass
12350000fs	4-clock "00001010"	1.5-clock "000"	1	tccd+3	Yes	3-clock "001010"	3-clock "000010"	Fail
20050200fs	3-clock "000010"	1.5-clock "000"	1	tccd+4	Yes	4-clock "00000010"	4-clock "00000010"	Pass
25201280fs	4-clock "000010"	0.5-clock "0"	1	tccd+6	No	NA	NA	
29250758fs	1-clock "10"	1.5-clock "010"	0	tccd+1	Yes	1-clock "10"	1-clock "10"	Pass
31025000fs	3-clock "000010"	1.5-clock "010"	0	tccd+3	Yes	3-clock "100010"	3-clock "100010"	Pass
34502000fs	4-clock "00001010"	0.5-clock "0"	0	tccd+4	Yes	4-clock "00001010"	4-clock "00001011"	Fail
38125675fs	2-clock DDR4 Style	1.5-clock	0	tccd+7	No	NA	NA	

- Various fields populated in Smart Log are the Timestamp of the inter-amble start, configured pre-amble and post-amble, command type, command-to-command spacing, expected and received inter-amble patterns, and the status of the comparison.
- RTC Parameter name – “EnableSmartLog”

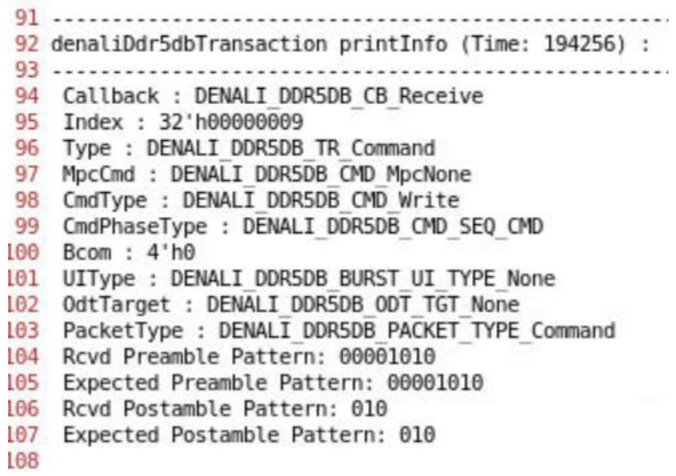
Callbacks Populating Expected, Received, and Custom Pattern Values



```
------------------------------------------------------
 denaliDdr5dbTransaction printInfo (Time: 194256) :
------------------------------------------------------
 Callback : DENALI_DDR5DB_CB_Receive
 Index : 32'h00000009
 Type : DENALI_DDR5DB_TR_Command
 MpcCmd : DENALI_DDR5DB_CMD_MpcNone
 CmdType : DENALI_DDR5DB_CMD_Write
 CmdPhaseType : DENALI_DDR5DB_CMD_SEQ_CMD
 Bcom : 4'h0
 UIType : DENALI_DDR5DB_BURST_UI_TYPE_None
 OdtTarget : DENALI_DDR5DB_ODT_TGT_None
 PacketType : DENALI_DDR5DB_PACKET_TYPE_Command
 Rcvd Interamble Pattern: 000010
 Expected Interamble Pattern: 000010
 Custom Interamble Pattern: 'hx
```



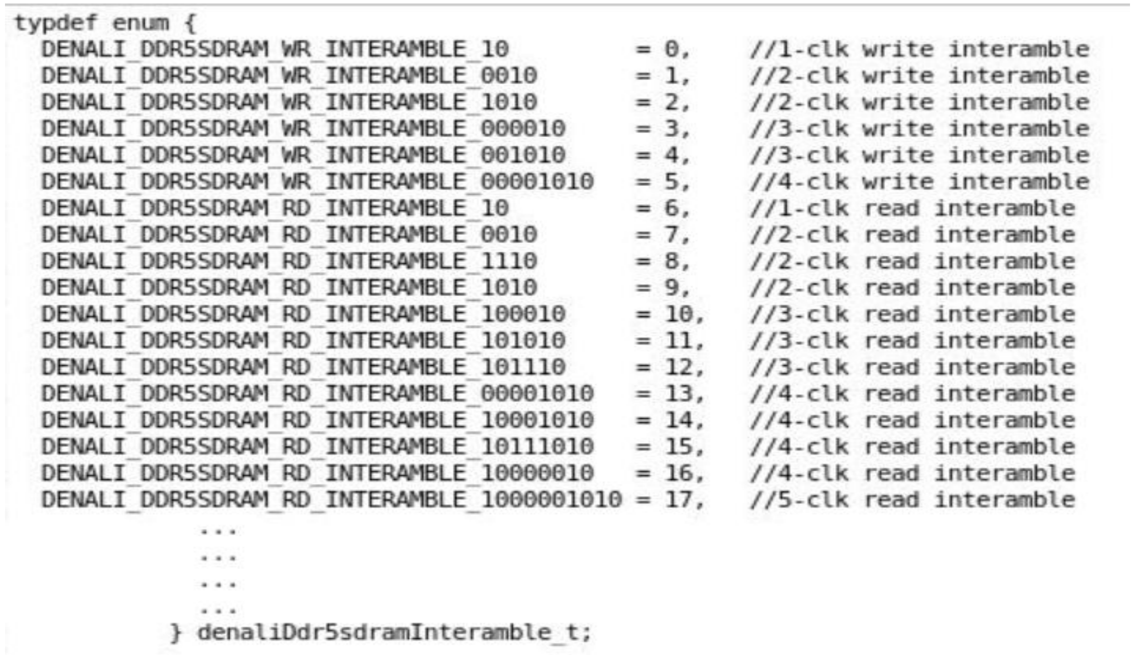
```
------------------------------------------------------
denaliDdr5dbTransaction printInfo (Time: 194256) :
------------------------------------------------------
 Callback : DENALI_DDR5DB_CB_Receive
 Index : 32'h00000009
 Type : DENALI_DDR5DB_TR_Command
 MpcCmd : DENALI_DDR5DB_CMD_MpcNone
 CmdType : DENALI_DDR5DB_CMD_Write
 CmdPhaseType : DENALI_DDR5DB_CMD_SEQ_CMD
 Bcom : 4'h0
 UIType : DENALI_DDR5DB_BURST_UI_TYPE_None
 OdtTarget : DENALI_DDR5DB_ODT_TGT_None
 PacketType : DENALI_DDR5DB_PACKET_TYPE_Command
 Rcvd Interamble Pattern: 000010
 Expected Interamble Pattern: 'hx
 Custom Interamble Pattern: 000010
```



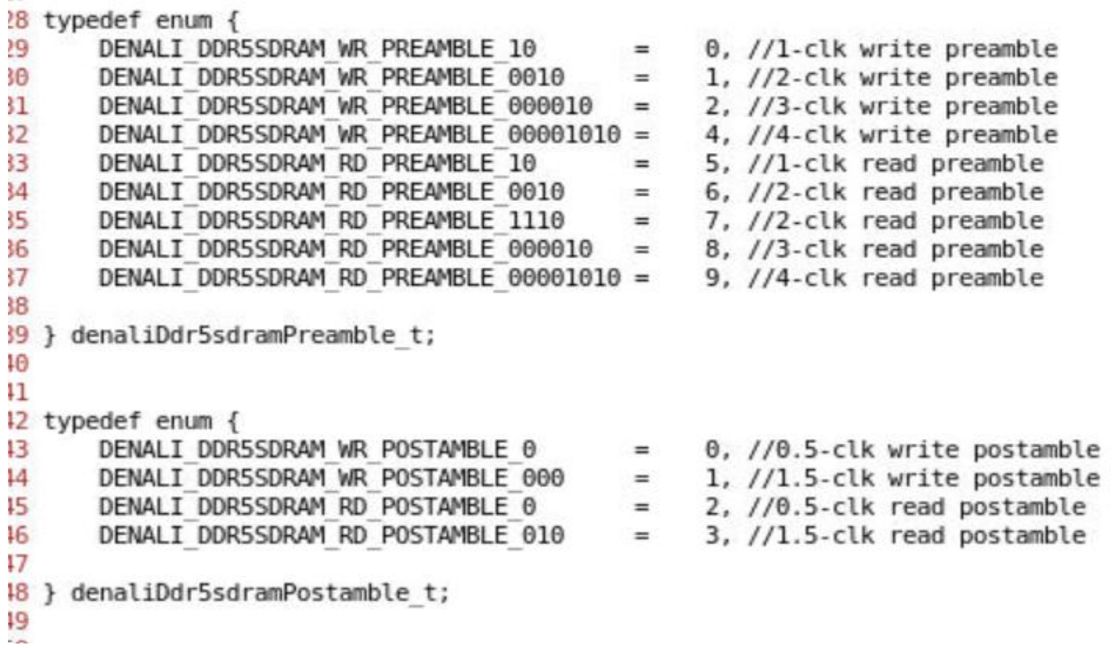
```
------------------------------------------------------
denaliDdr5dbTransaction printInfo (Time: 194256) :
------------------------------------------------------
 Callback : DENALI_DDR5DB_CB_Receive
 Index : 32'h00000009
 Type : DENALI_DDR5DB_TR_Command
 MpcCmd : DENALI_DDR5DB_CMD_MpcNone
 CmdType : DENALI_DDR5DB_CMD_Write
 CmdPhaseType : DENALI_DDR5DB_CMD_SEQ_CMD
 Bcom : 4'h0
 UIType : DENALI_DDR5DB_BURST_UI_TYPE_None
 OdtTarget : DENALI_DDR5DB_ODT_TGT_None
 PacketType : DENALI_DDR5DB_PACKET_TYPE_Command
 Rcvd Preamble Pattern: 00001010
 Expected Preamble Pattern: 00001010
 Rcvd Postamble Pattern: 010
 Expected Postamble Pattern: 010
```

- Callback snippets depicting how “Rcvd Interamble Pattern,” “Expected Interamble Pattern,” “Customer Interamble Pattern,” or “Rcvd Preamble Pattern,” “Expected Preamble Pattern,” “Rcvd Postamble Pattern,” “Expected Postamble Pattern” get printed when “EnableSmartLog” option is set to 1. The value populated in a field depends on the inter-amble or full pre-amble and post-amble being generated

ENUMs to Store Predefined Pattern Values



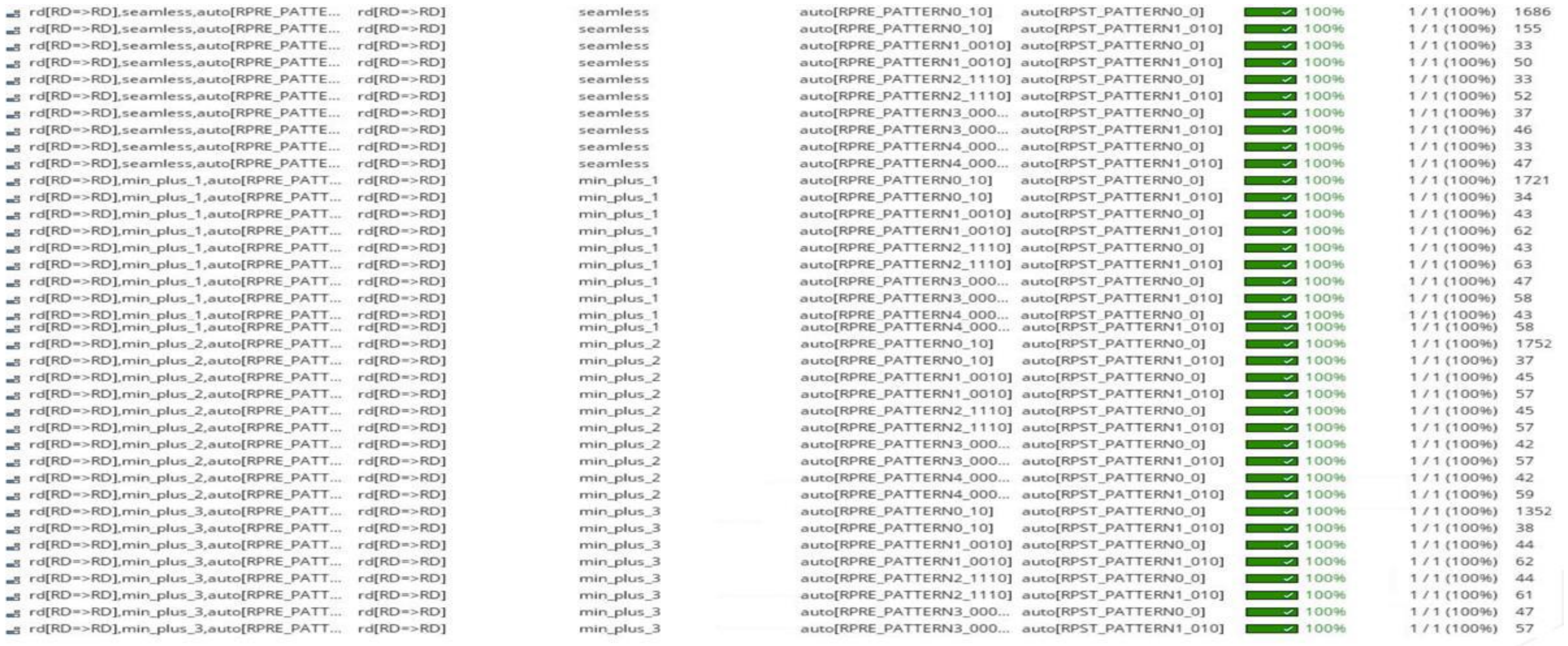
```
typdef enum {
  DENALI_DDR5SDRAM_WR_INTERAMBLE_10          = 0,    //1-clk write interamble
  DENALI_DDR5SDRAM_WR_INTERAMBLE_0010        = 1,    //2-clk write interamble
  DENALI_DDR5SDRAM_WR_INTERAMBLE_1010        = 2,    //2-clk write interamble
  DENALI_DDR5SDRAM_WR_INTERAMBLE_000010      = 3,    //3-clk write interamble
  DENALI_DDR5SDRAM_WR_INTERAMBLE_001010      = 4,    //3-clk write interamble
  DENALI_DDR5SDRAM_WR_INTERAMBLE_00001010    = 5,    //4-clk write interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_10          = 6,    //1-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_0010        = 7,    //2-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_1110        = 8,    //2-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_1010        = 9,    //2-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_100010      = 10,   //3-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_101010      = 11,   //3-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_101110      = 12,   //3-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_00001010    = 13,   //4-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_10001010    = 14,   //4-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_10111010    = 15,   //4-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_10000010    = 16,   //4-clk read interamble
  DENALI_DDR5SDRAM_RD_INTERAMBLE_1000001010  = 17,   //5-clk read interamble
              ...
              ...
              ...
              ...
        } denaliDdr5sdramInteramble_t;
```



```
typedef enum {
    DENALI_DDR5SDRAM_WR_PREAMBLE_10          =    0, //1-clk write preamble
    DENALI_DDR5SDRAM_WR_PREAMBLE_0010        =    1, //2-clk write preamble
    DENALI_DDR5SDRAM_WR_PREAMBLE_000010      =    2, //3-clk write preamble
    DENALI_DDR5SDRAM_WR_PREAMBLE_00001010 =       4, //4-clk write preamble
    DENALI_DDR5SDRAM_RD_PREAMBLE_10          =    5, //1-clk read preamble
    DENALI_DDR5SDRAM_RD_PREAMBLE_0010        =    6, //2-clk read preamble
    DENALI_DDR5SDRAM_RD_PREAMBLE_1110        =    7, //2-clk read preamble
    DENALI_DDR5SDRAM_RD_PREAMBLE_000010      =    8, //3-clk read preamble
    DENALI_DDR5SDRAM_RD_PREAMBLE_00001010 =       9, //4-clk read preamble

} denaliDdr5sdramPreamble_t;


typedef enum {
    DENALI_DDR5SDRAM_WR_POSTAMBLE_0          =    0, //0.5-clk write postamble
    DENALI_DDR5SDRAM_WR_POSTAMBLE_000        =    1, //1.5-clk write postamble
    DENALI_DDR5SDRAM_RD_POSTAMBLE_0          =    2, //0.5-clk read postamble
    DENALI_DDR5SDRAM_RD_POSTAMBLE_010        =    3, //1.5-clk read postamble

} denaliDdr5sdramPostamble_t;
```

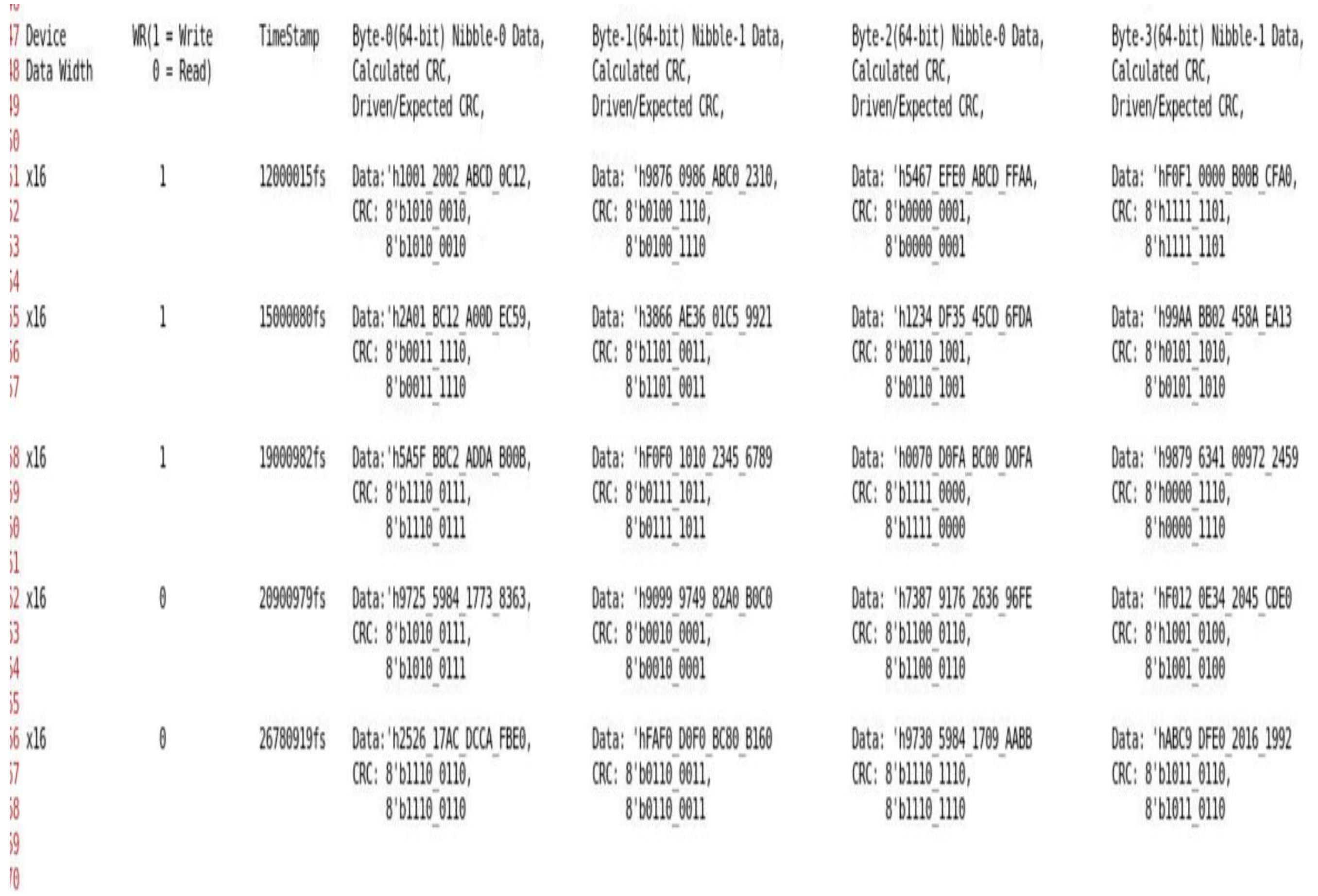
- The Smart Log generation and callbacks that populate the value of the expected and received inter-amble, pre-amble, and post-amble pattern, then fetch those pattern values based on the ENUMs index from these ENUMs
- Different ENUMs to store inter-amble, pre-amble, and post-amble values for writes and reads.

Pre-amble, Post-amble, Inter-amble Coverage (Internal VIP Coverage)



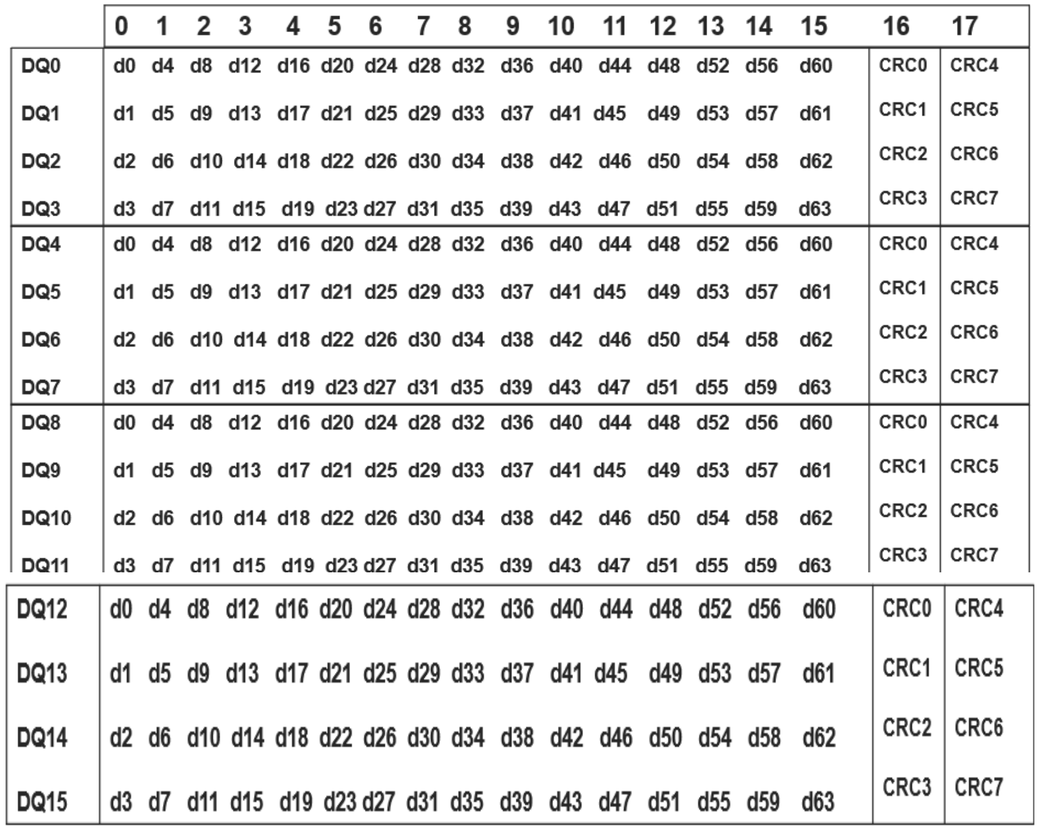
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN0_10]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	1686
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN0_10]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	155
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN1_0010]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	33
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN1_0010]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	50
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN2_1110]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	33
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN2_1110]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	52
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN3_000...	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	37
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN3_000...	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	46
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN4_000...	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	33
rd[RD=>RD],seamless,auto[RPRE_PATTE...	rd[RD=>RD]	seamless	auto[RPRE_PATTERN4_000...	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	47
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN0_10]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	1721
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN0_10]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	34
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN1_0010]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	43
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN1_0010]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	62
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN2_1110]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	43
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN2_1110]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	63
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN3_000...	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	47
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN3_000...	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	58
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN4_000...	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	43
rd[RD=>RD],min_plus_1,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_1	auto[RPRE_PATTERN4_000...	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	58
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN0_10]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	1752
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN0_10]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	37
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN1_0010]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	45
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN1_0010]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	57
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN2_1110]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	45
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN2_1110]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	57
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN3_000...	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	42
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN3_000...	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	57
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN4_000...	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	42
rd[RD=>RD],min_plus_2,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_2	auto[RPRE_PATTERN4_000...	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	59
rd[RD=>RD],min_plus_3,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_3	auto[RPRE_PATTERN0_10]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	1352
rd[RD=>RD],min_plus_3,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_3	auto[RPRE_PATTERN0_10]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	38
rd[RD=>RD],min_plus_3,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_3	auto[RPRE_PATTERN1_0010]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	44
rd[RD=>RD],min_plus_3,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_3	auto[RPRE_PATTERN1_0010]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	62
rd[RD=>RD],min_plus_3,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_3	auto[RPRE_PATTERN2_1110]	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	44
rd[RD=>RD],min_plus_3,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_3	auto[RPRE_PATTERN2_1110]	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	61
rd[RD=>RD],min_plus_3,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_3	auto[RPRE_PATTERN3_000...	auto[RPST_PATTERN0_0]	🟩✓ 100%	1 / 1 (100%)	47
rd[RD=>RD],min_plus_3,auto[RPRE_PATT...	rd[RD=>RD]	min_plus_3	auto[RPRE_PATTERN3_000...	auto[RPST_PATTERN1_010]	🟩✓ 100%	1 / 1 (100%)	57

Use Case II: For DDR5 DQ CRC Pattern Matching



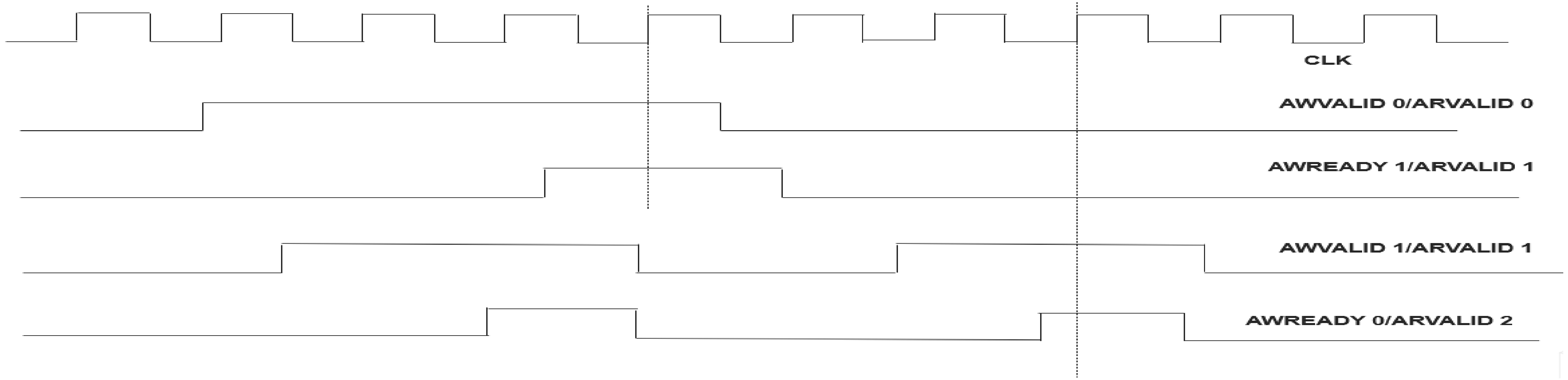
Device Data Width	WR(1 = Write 0 = Read)	TimeStamp	Byte-0(64-bit) Nibble-0 Data, Calculated CRC, Driven/Expected CRC,	Byte-1(64-bit) Nibble-1 Data, Calculated CRC, Driven/Expected CRC,	Byte-2(64-bit) Nibble-0 Data, Calculated CRC, Driven/Expected CRC,	Byte-3(64-bit) Nibble-1 Data, Calculated CRC, Driven/Expected CRC,
x16	1	12000015fs	Data:'h1001_2002_ABCD_0C12, CRC: 8'b1010_0010, 8'b1010_0010	Data: 'h9876_0986_ABC0_2310, CRC: 8'b0100_1110, 8'b0100_1110	Data: 'h5467_EFE0_ABCD_FFAA, CRC: 8'b0000_0001, 8'b0000_0001	Data: 'hF0F1_0000_B00B_CFA0, CRC: 8'h1111_1101, 8'h1111_1101
x16	1	15000080fs	Data:'h2A01_BC12_A00D_EC59, CRC: 8'b0011_1110, 8'b0011_1110	Data: 'h3866_AE36_01C5_9921 CRC: 8'b1101_0011, 8'b1101_0011	Data: 'h1234_DF35_45CD_6FDA CRC: 8'b0110_1001, 8'b0110_1001	Data: 'h99AA_BB02_458A_EA13 CRC: 8'h0101_1010, 8'b0101_1010
x16	1	19000982fs	Data:'h5A5F_BBC2_ADDA_B00B, CRC: 8'b1110_0111, 8'b1110_0111	Data: 'hF0F0_1010_2345_6789 CRC: 8'b0111_1011, 8'b0111_1011	Data: 'h0070_D0FA_BC00_D0FA CRC: 8'b1111_0000, 8'b1111_0000	Data: 'h9879_6341_00972_2459 CRC: 8'h0000_1110, 8'h0000_1110
x16	0	20900979fs	Data:'h9725_5984_1773_8363, CRC: 8'b1010_0111, 8'b1010_0111	Data: 'h9099_9749_82A0_B0C0 CRC: 8'b0010_0001, 8'b0010_0001	Data: 'h7387_9176_2636_96FE CRC: 8'b1100_0110, 8'b1100_0110	Data: 'hF012_0E34_2045_CDE0 CRC: 8'h1001_0100, 8'b1001_0100
x16	0	26780919fs	Data:'h2526_17AC_DCCA_FBE0, CRC: 8'b1110_0110, 8'b1110_0110	Data: 'hFAF0_D0F0_BC80_B160 CRC: 8'b0110_0011, 8'b0110_0011	Data: 'h9730_5984_1709_AABB CRC: 8'b1110_1110, 8'b1110_1110	Data: 'hABC9_DFE0_2016_1992 CRC: 8'b1011_0110, 8'b1011_0110

CRC Bit Mapping for X16 Devices



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
DQ0	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ1	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ2	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ3	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7
DQ4	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ5	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ6	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ7	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7
DQ8	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ9	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ10	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ11	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7
DQ12	d0	d4	d8	d12	d16	d20	d24	d28	d32	d36	d40	d44	d48	d52	d56	d60	CRC0	CRC4
DQ13	d1	d5	d9	d13	d17	d21	d25	d29	d33	d37	d41	d45	d49	d53	d57	d61	CRC1	CRC5
DQ14	d2	d6	d10	d14	d18	d22	d26	d30	d34	d38	d42	d46	d50	d54	d58	d62	CRC2	CRC6
DQ15	d3	d7	d11	d15	d19	d23	d27	d31	d35	d39	d43	d47	d51	d55	d59	d63	CRC3	CRC7

Use Case III: Outside DDR5

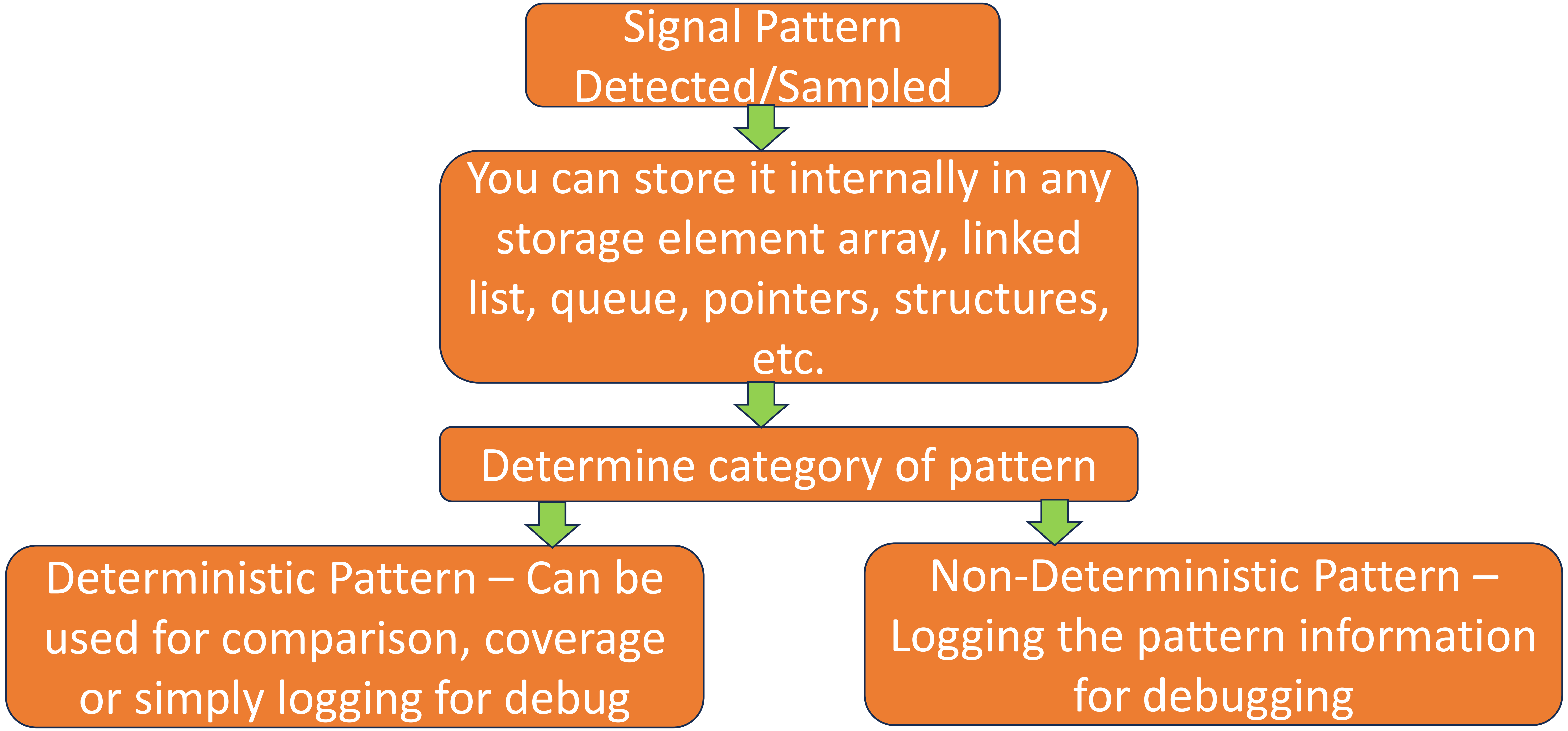


- Assuming AXI Multi-Master, Multi-Slave Sub System
- Each master can communicate with any slave using VALID, READY handshaking. Separate VALID, READY handshake signals for write and read channels.
- Logging information related to each VALID, READY, such as its assertion time, de-assertion time, overall pulse duration, and more, can help significantly in debugging issues pertaining to these signals

Actual Customer Use Case

- One of the internal customer was working on a full chip memory controller verification
- Use case – A long burst of write and read with varying spacing and configured pre-amble and post-amble being changed after every set of 8-10 commands
- Challenge – Many inter-amble patterns are possible in this case
- Issue—The customer encountered many errors related to the wrong inter-amble pattern or the number of toggles in the inter-amble pattern
- Solution – Using the Smart Log solution helped them in faster debugging of errors and fixing the stimulus
- Impact - Efforts reduced significantly by ~25%.

General Sense of Solution


Signal Pattern Detected/Sampled

You can store it internally in any storage element array, linked list, queue, pointers, structures, etc.

Determine category of pattern

Deterministic Pattern – Can be used for comparison, coverage or simply logging for debug

Non-Deterministic Pattern – Logging the pattern information for debugging

Questions ???