AURIX 2G Direct Memory Access(DMA)

IFCN ATV SMD GC SAE MC

Aurix2G- TC39XA-step DMA

This presentation briefly explains:

- System View: DMA
- The Basic Transaction Control Set (TCS) of the DMA controller in AURIX-TC39xA.
- Features of Aurix DMA.
- HW architecture of the DMA controller.
- Channel Request and Arbitration
- **NOTE: The description in this version applies to TC39xA, and does not cover the changes in TC38xA or TC39xB.**
- **There are no functional changes between AurixTC2XX and TC39xA-step DMA.**
- **There will be some functional changes on the TC38xA/TC39xB DMA.**

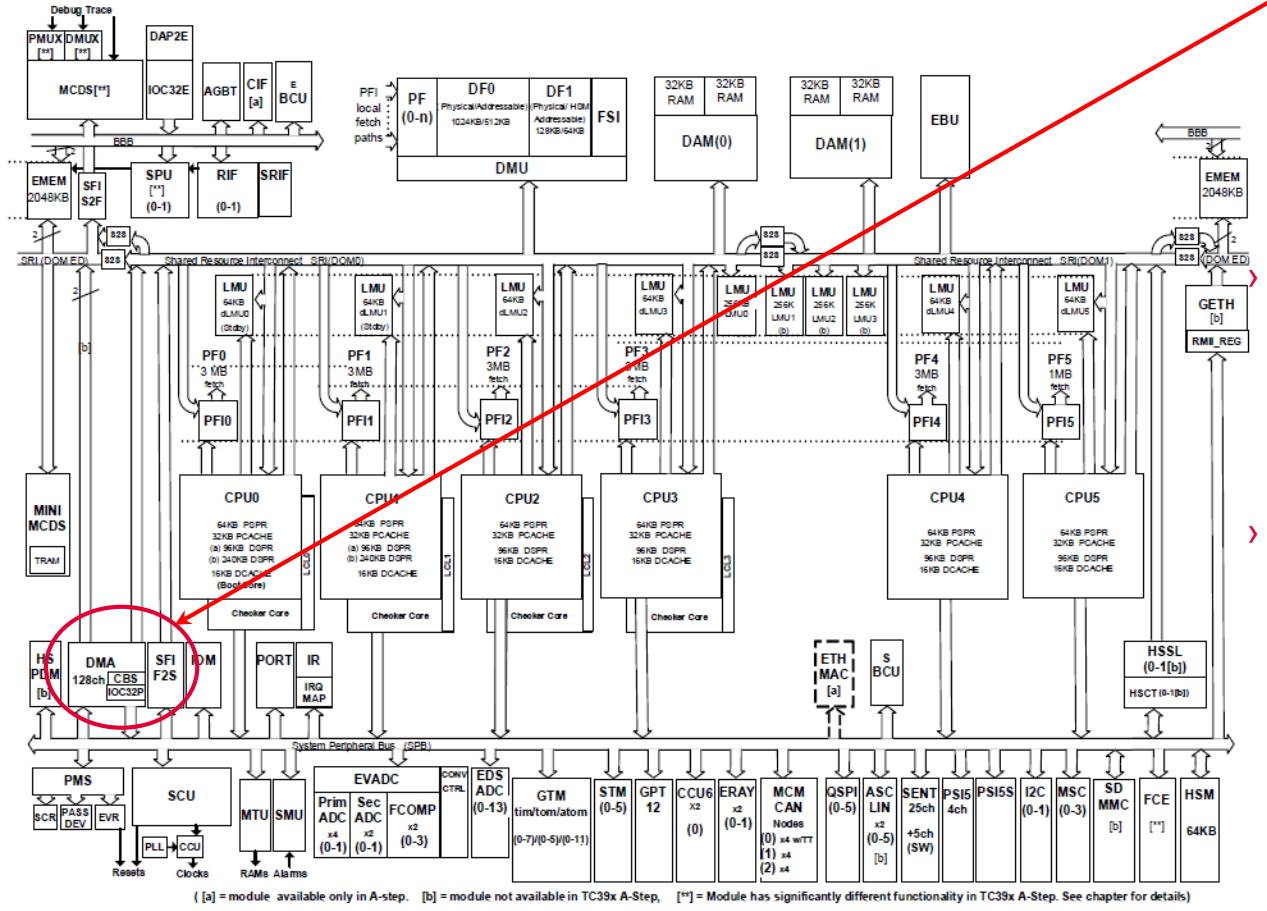
Agenda

1. System View
2. DMA Basics
3. Features
4. H/W Architecture
5. Channel Request & Arbitration
6. Summary

Agenda

1. System View
2. DMA Basics
3. Features
4. H/W Architecture
5. Channel Request & Arbitration
6. Summary

DMA in Aurix2G-TC39xA



([a] = module available only in A-step. [b] = module not available in TC39x A-Step, [**] = Module has significantly different functionality in TC39x A-Step. See chapter for details)

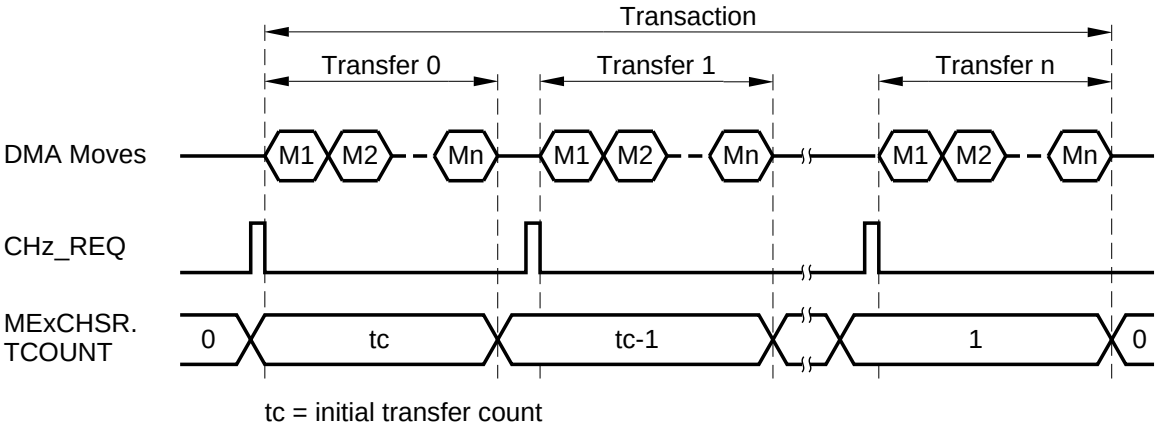
- DMA:
  - One SRI Master to DOM0
  - One SPB Master
  - One SPB slave (Access to DMA resources)
- Multiple DMA Channels
  - 2 ME, 128 Channels in TC39x
  - CH 127 is Highest Prio.
- Interrupt Connections:
  - All Interrupts can be routed to DMA.
  - Act as Hardware Triggers.

Agenda

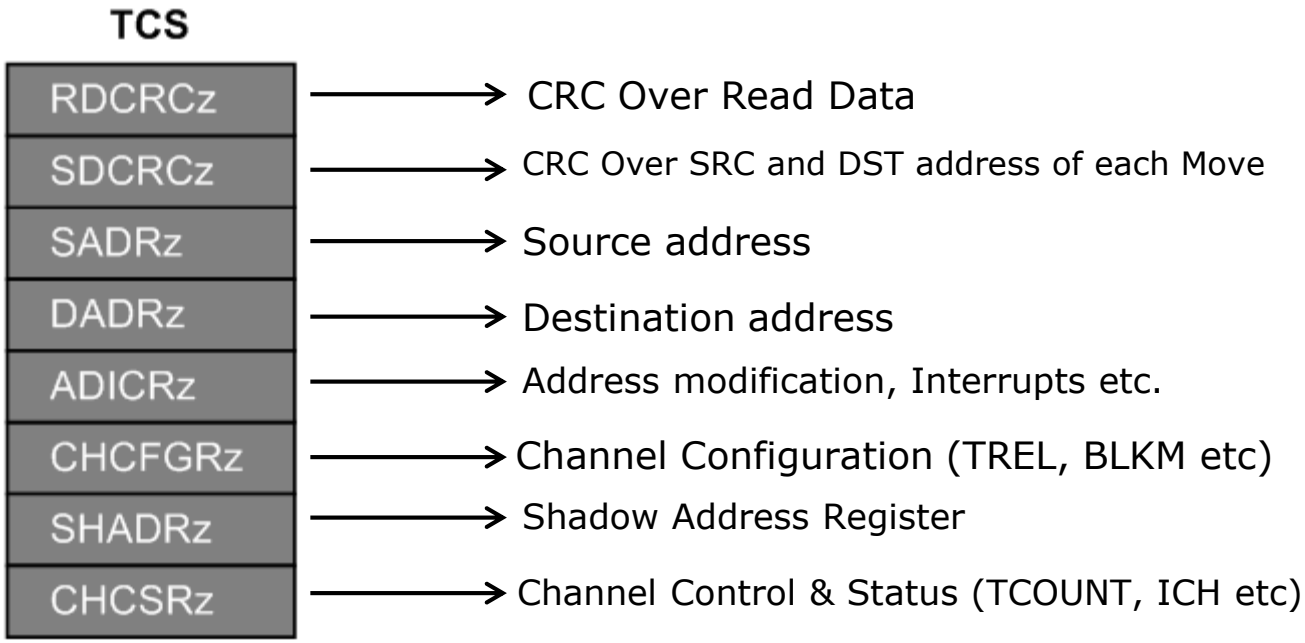
1. System View
2. DMA Basics
3. Features
4. H/W Architecture
5. Channel Request & Arbitration
6. Summary

DMA Transaction & Transfers

- 1 DMA Move = 1 Bus Read + 1 Bus Write
- Transfer: Un-interruptable DMA operation.
  - Possible to trigger each transfer, not each move.
  - ➔ Longer transfers can block a higher priority channel.



Transaction Control Set – Programmer View



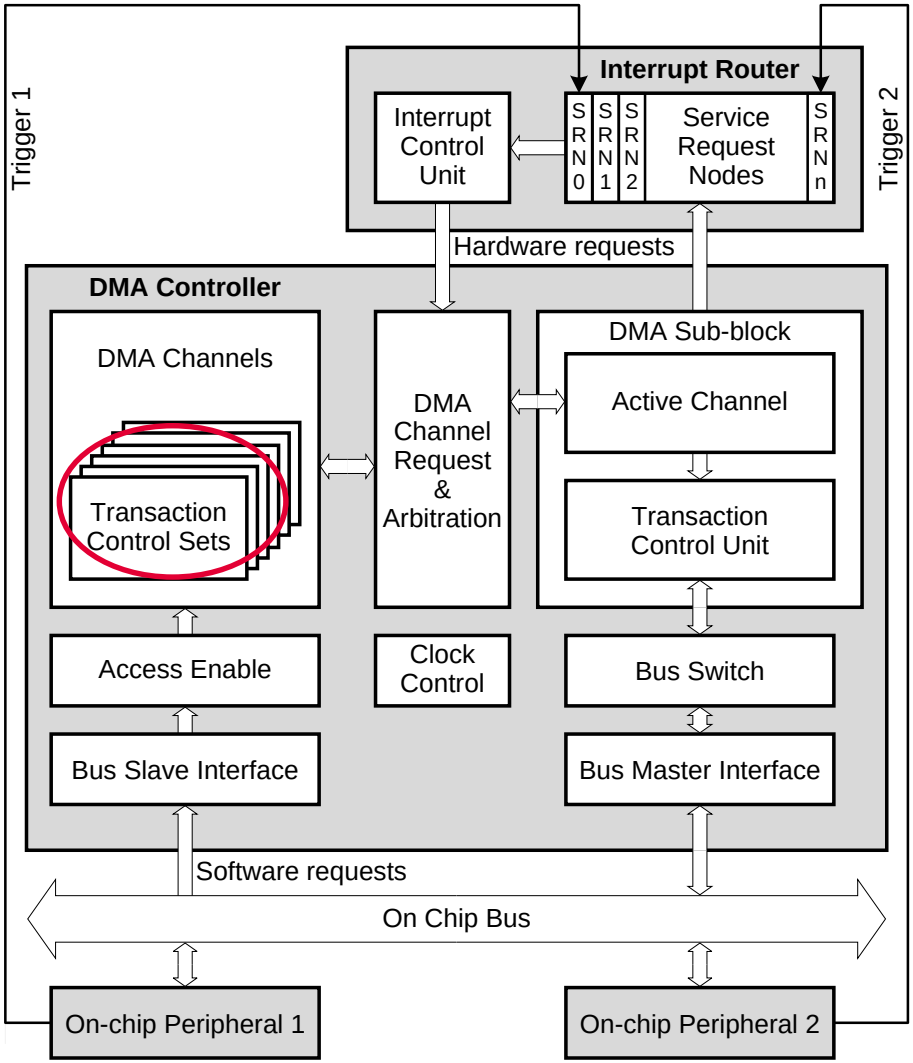
DMA Transaction Control Parameters

Parameter	Description
CHCFGRz.CHDW	channel data width – one DMA move (one read and one write access)
CHCFGRz.BLKM	block mode value - number of DMA moves per DMA transfer
CHCFGRz.TREL	transfer reload value – number of DMA transfers per transaction (per request)
CHCFGRz.RROAT	reset request only after transaction (RROAT = 1)
ADICRz.SCBE / ADICRz.DCBE	src/dest circular buffer enable/disable
ADICRz.SMF / ADICRz.DMF	src/dest address modification factor
ADICRz.INCS / ADICRz.INCD	increment of src/dest address
ADICRz.CBLS / ADICRz.CBLD	circular buffer length src/dest

Double Buffering Modes
Shadow Control Table ADICRz.SHCT

SHCT	Description
0000_B	Mission Mode (Shadow address register not used and set to 0x00)
0001_B	Shadow Address used for Source Address Buffering (Read Only)
0010_B	Shadow Address used for Destination Address Buffering (Read Only)
0011_B	Reserved
0100_B	Reserved
0101_B	Shadow Address used for Source Address Buffering (Direct Write)
0110_B	Shadow Address used for Destination Address Buffering (Direct Write)
0111_B	Reserved
1000_B	Double Source Buffering (software switch only)
1001_B	Double Source Buffering (automatic hardware & software switch)
1010_B	Double Destination Buffering (software switch only)
1011_B	Double Destination Buffering (automatic hardware & software switch)
1100_B	DMA Linked List
1101_B	Accumulated Linked List
1110_B	Safe Linked Link
1111_B	Conditional Linked List

Transaction Control Set – HW View



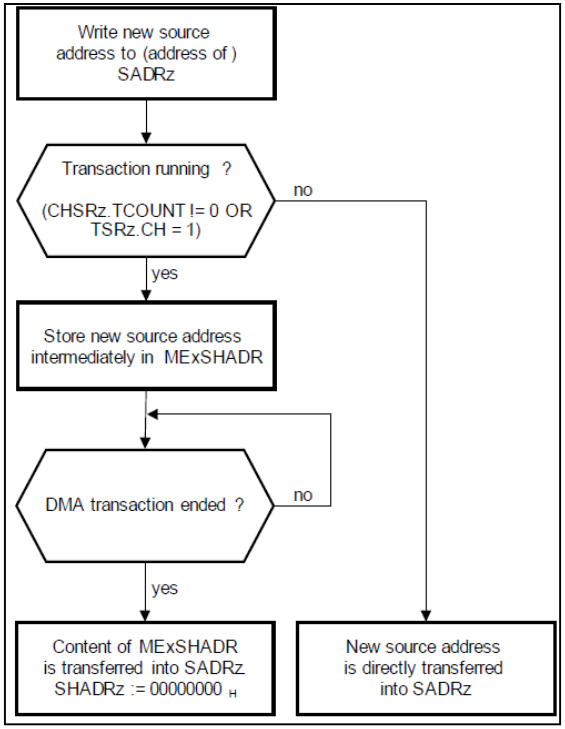
- TCS stored in the DMARAM.
- Copied into Active sub-block (Move Engine) when a channel becomes active

TIP: Updating the TCS (& Shadow Address)

The reprogramming of channel specific values (except for the selected shadow address register) must be avoided while a DMA channel is active as the data transfer maybe corrupted.

SHCT	Description
0000_B	Source and destination registers written directly. *Note: Shadow address register is not used and set to 00000000_H.*
0001_B	**Source Address Buffering (Read Only)** Shadow address used for source buffering. When writing to SADRz, the address is buffered in SHADRz and transferred to SADRz with the start of the next DMA transaction. *Note: Shadow address register is read only and is automatically set to 00000000_H when the shadow transfer takes place (equal to AudoNG).*
0010_B	**Destination Address Buffering (Read Only)** Shadow address used for destination buffering. When writing to DADRz, the address is buffered in SHADRz and transferred to DADRz with the start of the next DMA transaction. *Note: Shadow address register is read only and is automatically set to 00000000_H when the shadow transfer takes place (equal to AudoNG).*
0101_B	**Source Address Buffering (Direct Write)** Shadow address used for source buffering. Shadow address register can be read and can be directly written. The value stored in SHADRz is not automatically modified when the shadow transfer takes place.
0110_B	**Destination Address Buffering (Direct Write)** Shadow address used for destination buffering. Shadow address register can be read and can be directly written. The value stored in SHADRz is not automatically modified when the shadow transfer takes place.

Shadow Register Control (ADICRz.SHCT)



Triggering the DMA: HW & SW

- DMA Transfers can be triggered via
  - Hardware Triggers
  - Software Triggers.

- HW Triggers:
  - Interrupts are routed to DMA.
  - Any peripheral which can trigger an interrupt can trigger a DMA transfer.
  - DMA Channel selected corresponds to Interrupt Priority Number
    - (SRPN in the Interrupt Router SRC register).

- SW Triggers:
  - Software writes to the SCH bit.

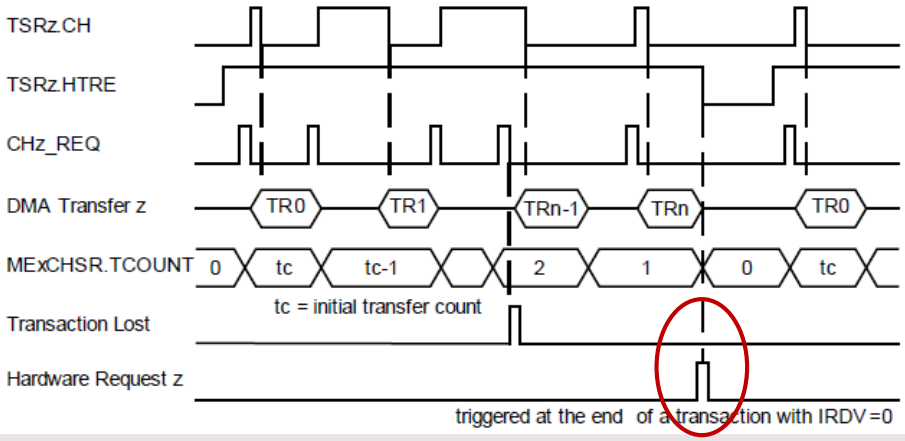
Agenda

1. System View
2. DMA Basics
3. Features
4. H/W Architecture
5. Channel Request & Arbitration
6. Summary

DMA Channel Interrupt

- Interrupts can be triggered –
  - When TCOUNT matches a specific value (CHCSRz.IRDV).
  - Whenever TCOUNT is decremented.

- In general, S/W handling of DMA transactions may look like:
  - S/W Configures DMA.
  - S/W Enables the S/W or H/W trigger.
  - Trigger interrupt at the end of the transaction.
  - Configure new transaction.



DMA CRC

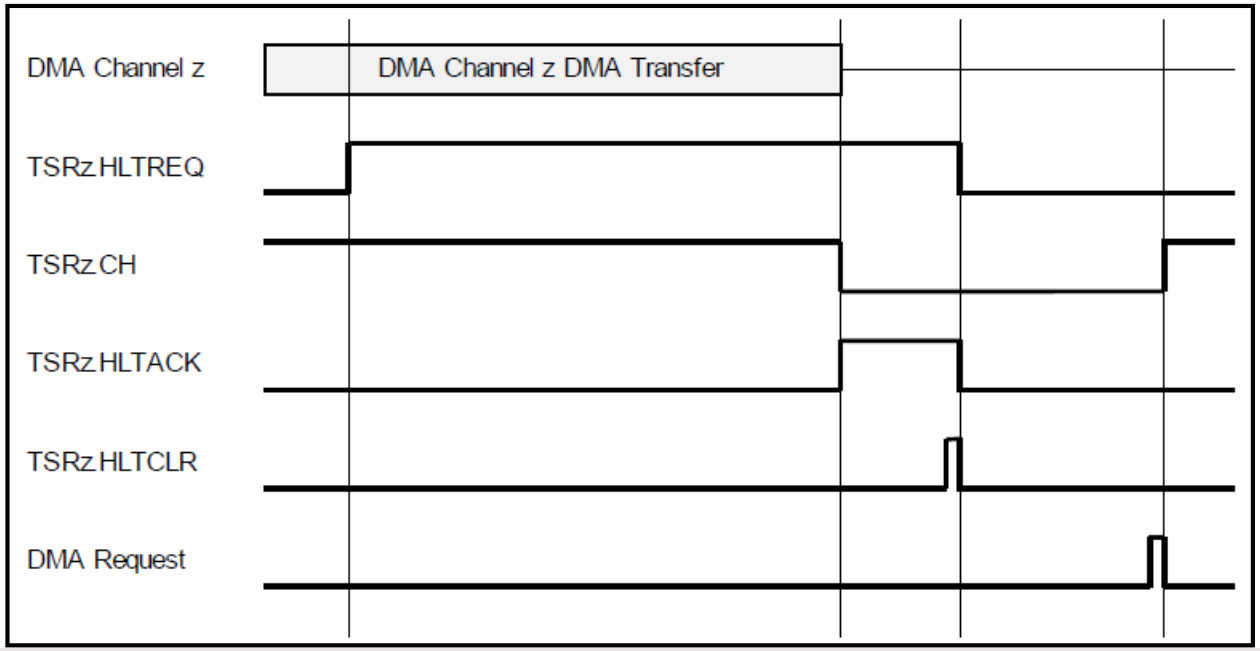
- **RDCRC**: CRC32 Over Read Data
- **SDCRC**: CRC32 Over SRC and DST addresses.

```
001:  <Mip>_ReturnType <Mip>_Dma_checkCRC (void)
002:  {
003:      <Mip>_ReturnType res = <MIP>_E_DMA_CRC_OK;
004:      uint32 sx[4] = { 0xD, 0xE, 0xA, 0xD };
005:      uint32 sy[4] = { 0xC, 0xA, 0xF, 0xE };
006:      uint32 dx[4] = { 0, 0, 0, 0 };
007:      uint32 dy[4] = { 0, 0, 0, 0 };
008:      uint32 *psx = sx, *psy = sy, *pdx = dx, *pdy = dy;
009:      uint32 sdcrc0 = 0, sdcrc1 = 0, rdcrc0 = 0, rdcrc1 = 0;
010:      const uint32 x = 10, y = 20;
011:
012:      // 1. Configure two channels x and y for software controlled mode.
013:      //    Source and destination are on the local stack
014:      MODULE_DMA.CH[x].CHCFGR.U = 0x004A0001; // CHDW=2, RROAT=1, BLKM=2, TREL=1
015:      MODULE_DMA.CH[y].CHCFGR.U = 0x004A0001;
016:      MODULE_DMA.CH[x].ADICR.U = 0x00000088;  // INCD=1, INCS=1
017:      MODULE_DMA.CH[y].ADICR.U = 0x00000088;
018:      MODULE_DMA.CH[x].SADR.U = (uint32) sx;
019:      MODULE_DMA.CH[y].SADR.U = (uint32) sy;
020:      MODULE_DMA.CH[x].DADR.U = (uint32) dx;
021:      MODULE_DMA.CH[y].DADR.U = (uint32) dy;
```

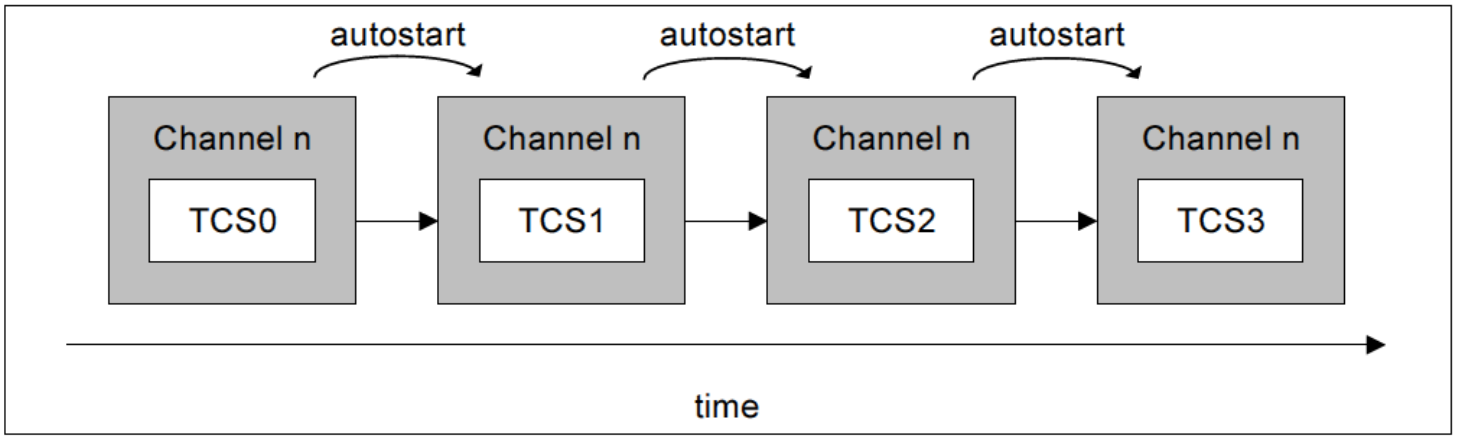
```
037:      // 3a. Calculate expected channel read data CRC and channel source
038:      //     and destination address CRC for both move engines.
039:      for (uint32 i = 0; i < 4; i++, psx++, pdx++, psy++, pdy++)
040:      {
041:          sdcrc1 = __crc32 (__crc32 (sdcrc1, (uint32) psx), (uint32) pdx);
042:          rdcrc1 = __crc32 (rdcrc1, *pdx);
043:          sdcrc0 = __crc32 (__crc32 (sdcrc0, (uint32) psy), (uint32) pdy);
044:          rdcrc0 = __crc32 (rdcrc0, *pdy);
045:      }
046:      // 3b ME0SDCRC, ME1SDCRC, ME0RDCRC, ME1SDCRC as expected?
047:      if (sdcrc0 != DMA_ME0SDCRC.U || sdcrc1 != DMA_ME1SDCRC.U ||
048:          rdcrc0 != DMA_ME0RDCRC.U || rdcrc1 != DMA_ME1RDCRC.U)
049:      {
050:          res = <MIP>_E_DMA_CRC_NOT_OK;
051:          goto error;
052:      }
053:
```

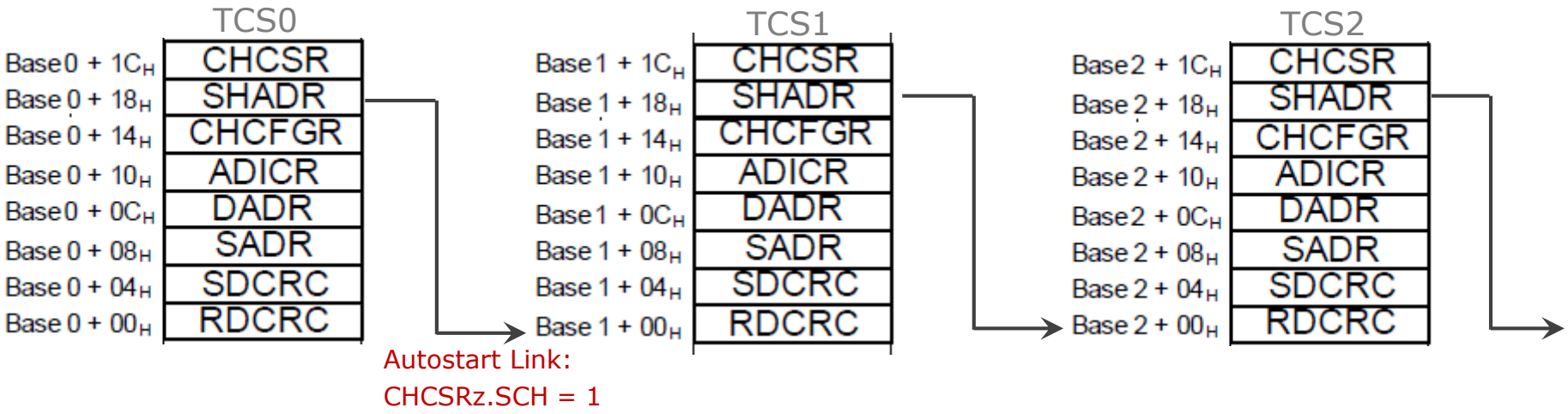
DMA Channel Halt Operation

- If DMA channel is in halt state (TSRz.HLTACK = 1) and hardware transaction request enable bit TSRz.HTRE = 1 then the DMA channel responds to DMA hardware requests as follows
  - Idle channel (TSRz.CH = 0): the access pending bit will be set and the DMA hardware request will be serviced when DMA channel z exits halt mode
  - Active channel (TSRz.CH = 1): the TSRz.TRL is set and the transaction is lost
    - If the enable transaction request lost interrupt bit ADICRz.ETRL is set then an interrupt will be generated for a transaction request lost event



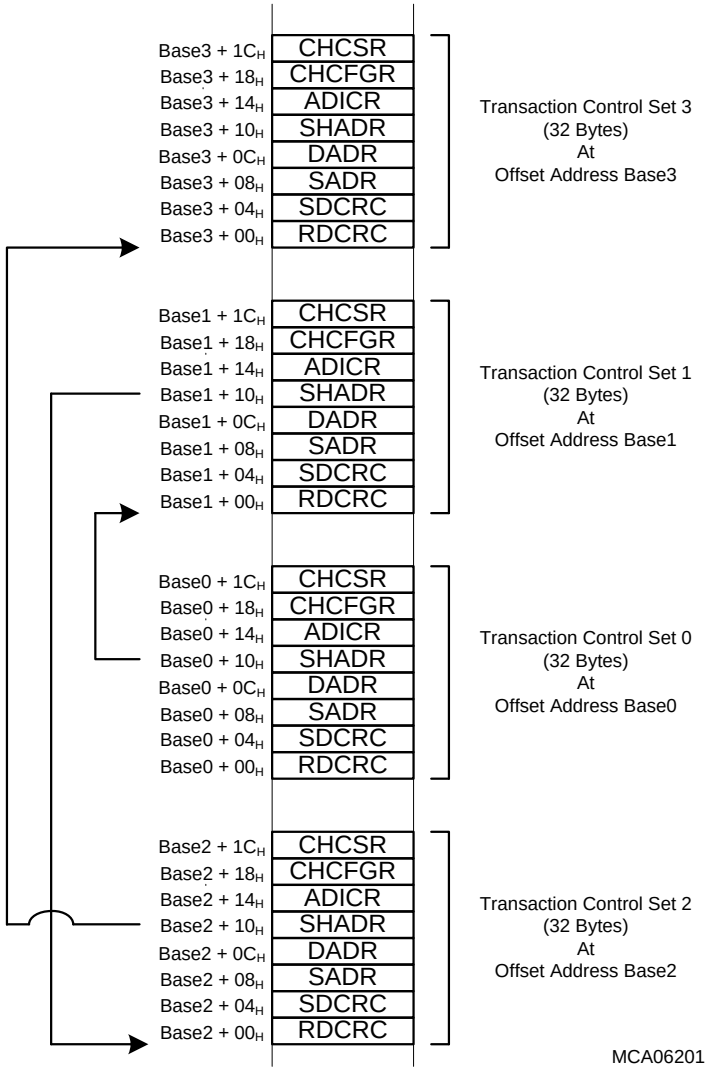
DMA Linked List (SHCT = 0xC)





- Base x: 32-byte Aligned (Eg. in SPR)

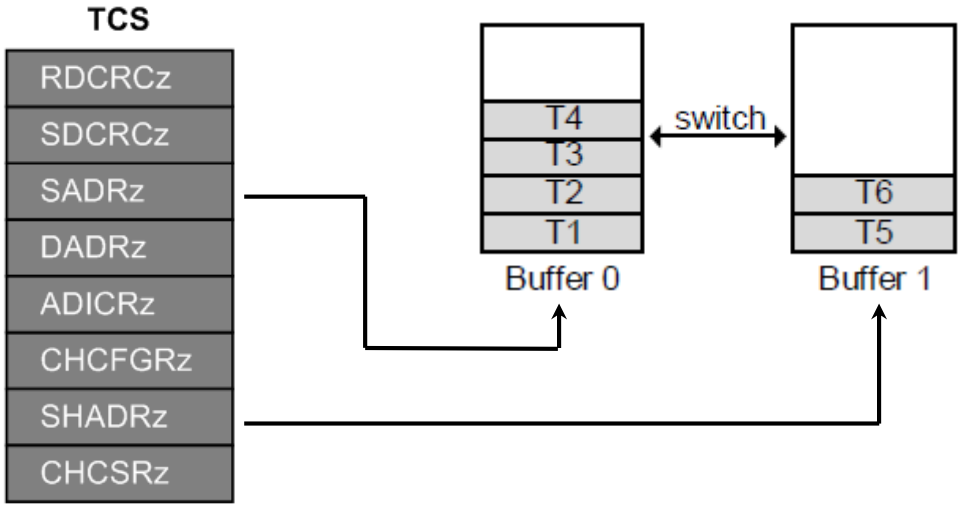
DMA & Accumulated Linked Lists



- Linked Lists
  - Multiple TCS/DMA channel
  - SHADR is address pointer
  - Auto-start if SCH = 1_B
  - Traffic management IRQ
- DMA Linked List
  - SDCRC: unique/transaction
  - RDCRC: unique/transaction
- Accumulated Linked List
  - SDCRC: accumulated
  - RDCRC: accumulated

Double Buffering

SHCT	Description
1000_B	**DMA Double Source Buffering** **Software Switch Only** Switching from the current source buffer to the other source buffer is controlled by the software switch CHCSRz.SWB
1001_B	**DMA Double Source Buffering** **Automatic Hardware and Software Switch** DMA controller automatically switches source buffers when current source buffer is emptied. Software source buffer switching controlled by CHCSRz.SWB
1010_B	**DMA Double Destination Buffering** **Software Switch Only** Switching from the current destination buffer to the other destination buffer is controlled by the software switch CHCSRz.SWB
1011_B	**DMA Double Destination Buffering** **Automatic Hardware and Software Switch** DMA controller automatically switches destination buffers when current destination buffer is full. Software destination buffer switching controlled by CHCSRz.SWB

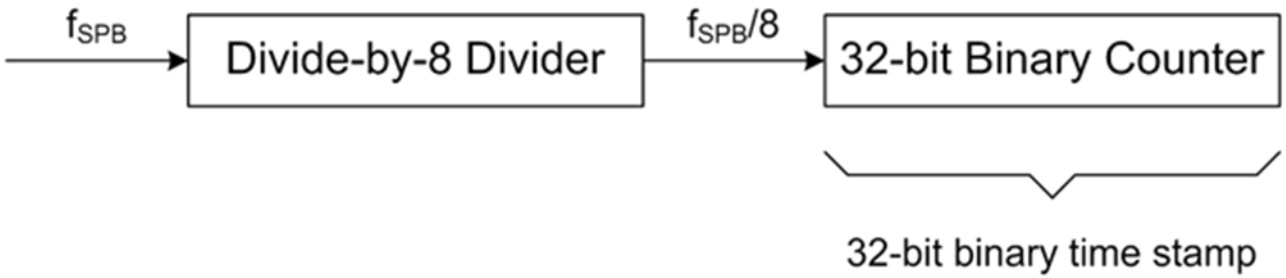


DMA Circular Buffering

- Separate Source and Destination Circular Buffers
- Addresses Wraps around at buffer limits.
- Buffer size defined by programmable bits CBLS (source) and CBLD (dest)
  - In ADICR register in the TCS.
  - 1,2,4,8,16…64k Byte Circular Buffer Sizes possible.
- Alignment:
  - Buffer should be aligned to its size.

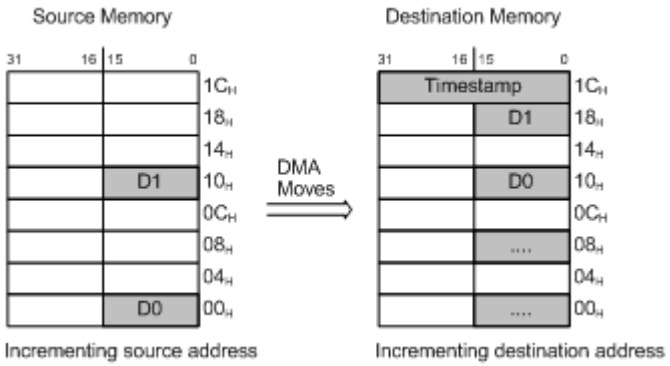
DMA Timestamp

- Digital time stamp to record completion of DMA transaction.
- Timestamp generated from a free-running $f_{SPB}/8$ clock.
- Read only TIME.COUNT is the current value of timestamp.
- Use timestamp to compare completion of DMA events.
- ADICz.STAMP enables/disables appendage of time stamp.
- DMA write move appends timestamp to end of transaction:
  - ADICRz.INCD = 1 then next higher word aligned address
  - ADICRz.INCD = 0 then next lower word aligned address



DMA Timestamp: Appendage of Timestamp

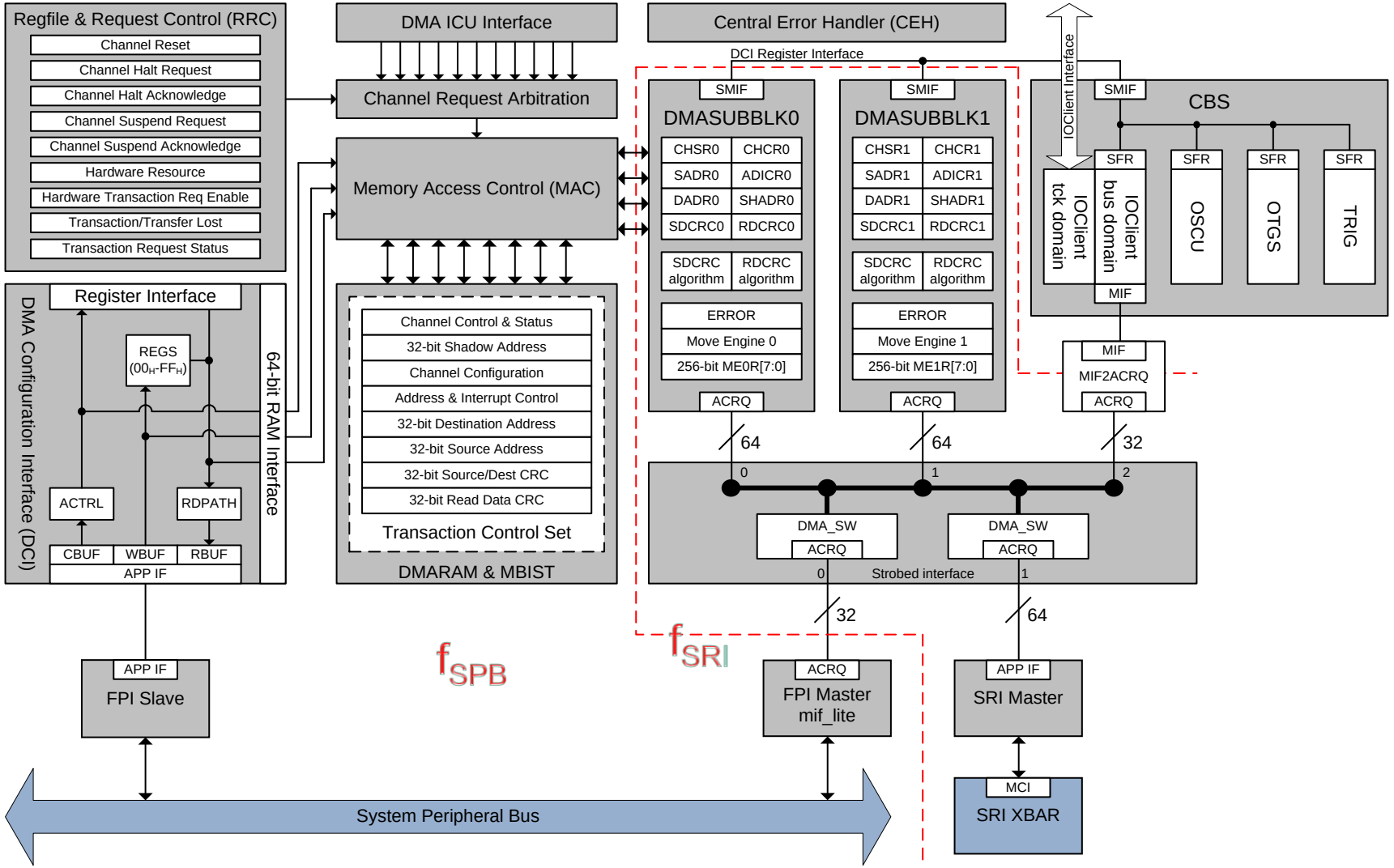
- DMA Channel z Transaction Control Set
  - CHCFGRz.CHDW = 001_B (SDTH)
  - ADICRz.INCD = 1_B (incrementing destination address)
- Timestamp is appended to the next higher word aligned destination address on completion of the DMA transaction.



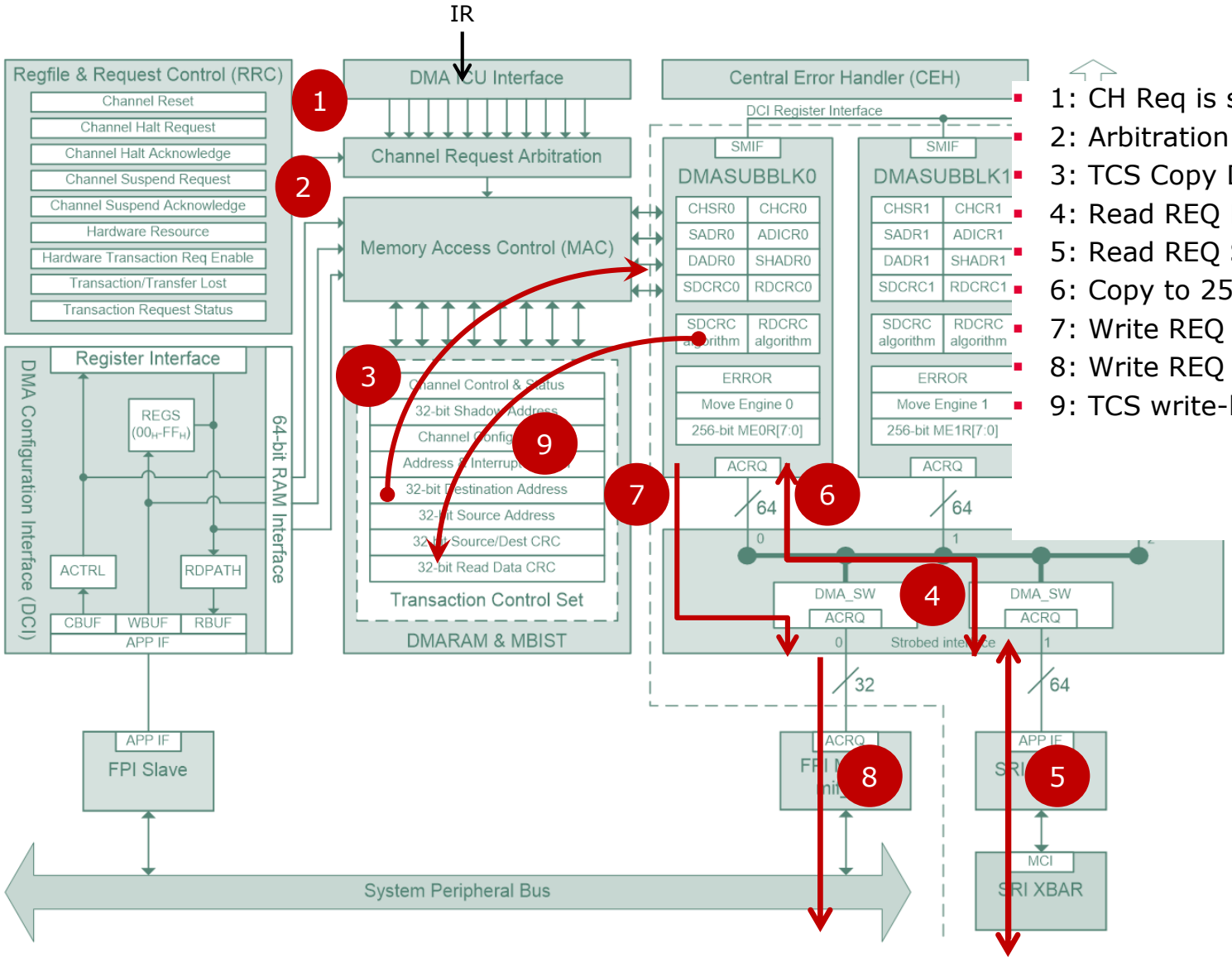
Agenda

1. System View
2. DMA Basics
3. Features
4. H/W Architecture
5. Channel Request & Arbitration
6. Summary

H/W architecture of the DMA



DMA FSM: SRI read -> SPB write (HW trigger)



- 1: CH Req is set.
- 2: Arbitration
- 3: TCS Copy DMARAM -> ME.
- 4: Read REQ ME -> Switch
- 5: Read REQ Switch -> SRI
- 6: Copy to 256-bit ME Buffer.
- 7: Write REQ ME -> Switch
- 8: Write REQ Switch -> FPI.
- 9: TCS write-back to DMARAM.

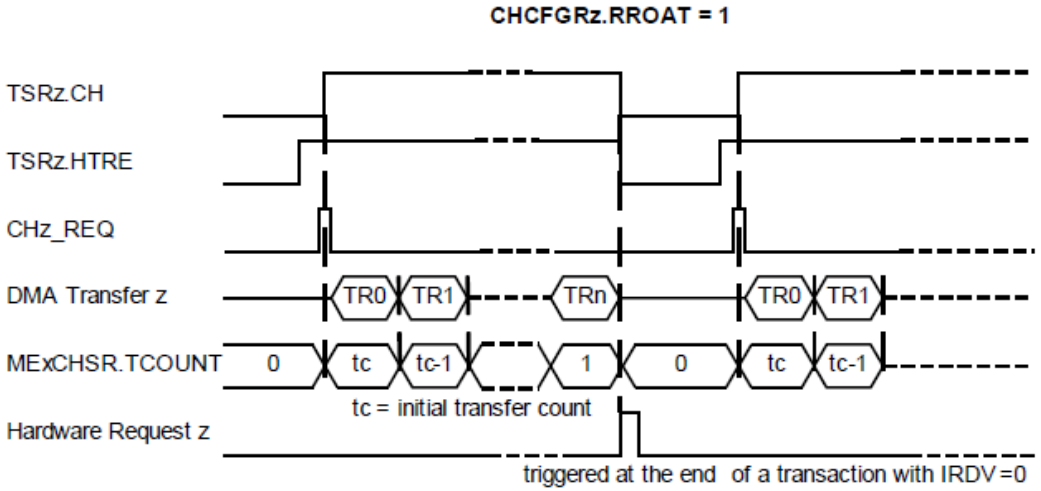
Agenda

1. System View
2. DMA Basics
3. Features
4. H/W Architecture
5. Channel Request & Arbitration
6. Summary

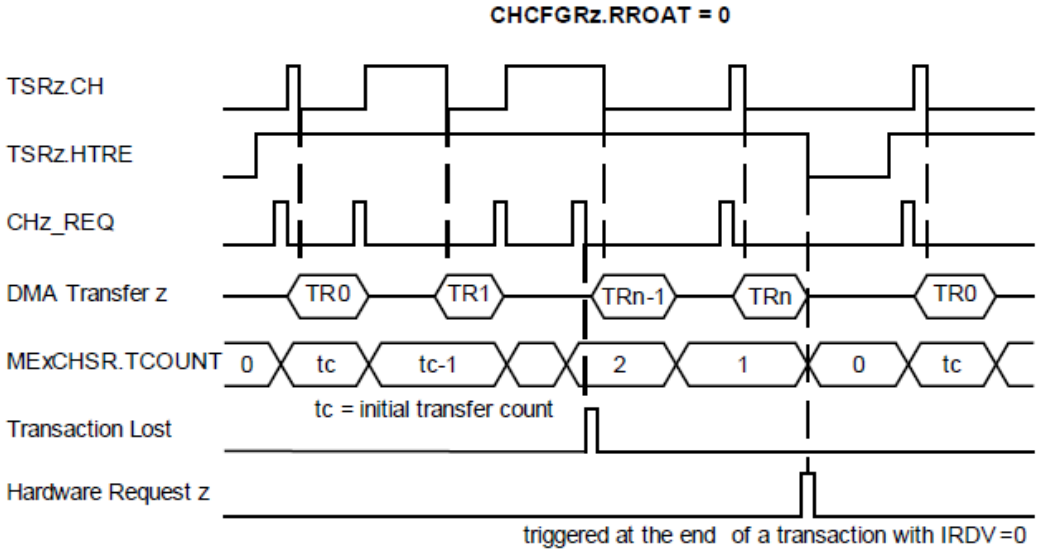
CH request and arbitration

- 1: CH Req is set.
- 2: Arbitration
- 3: TCS Copy DMARAM -> ME.
- 4: Read REQ ME -> Switch
- 5: Read REQ Switch -> SRI
- 6: Copy to 256-bit ME Buffer.
- 7: Write REQ ME -> Switch
- 8: Write REQ Switch -> FPI.
- 9: TCS write-back to DMARAM.

Channel Request (CH)



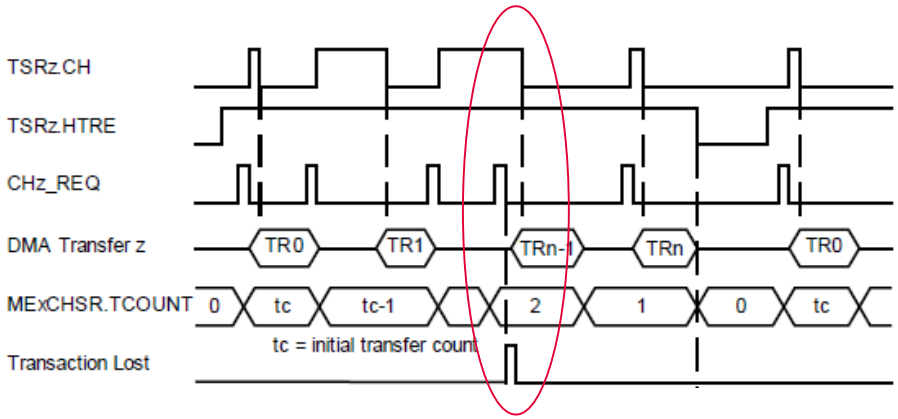
- RROAT = 1
  - Reset Request Only After Transaction.
  - One CH request per whole Transaction

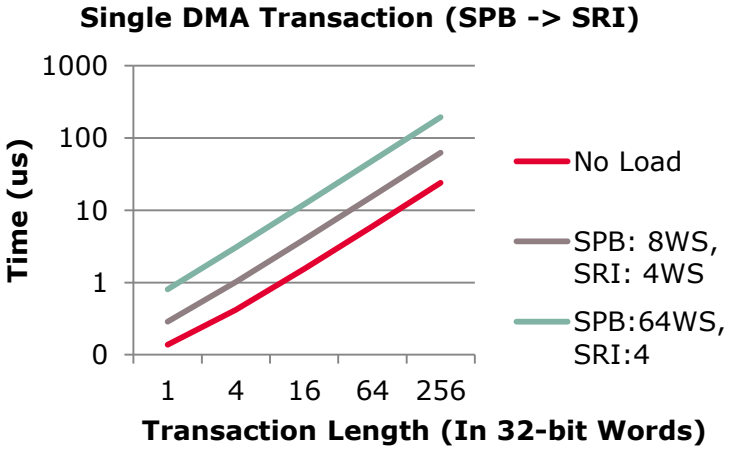


- RROAT = 0
  - Reset Request after each Transfer
  - One CH request per Transfer

TIP: Preventing Request Loss (TRL)

- Consider:
  - Basic DMA triggering rate (Eg. Peripheral interrupt rate)
    - Eg. Usage of FIFO threshold mode in QSPI.
  - System (Bus) Load
  - Priority "Inversion":
    - Internal DMA Arbitration.
      - Longer **Transfers** of a lower prio channel could block a higher prio one.
    - Move Engine Priority on SPB.





Agenda

1. System View
2. DMA Basics
3. Features
4. H/W Architecture
5. Channel Request & Arbitration
6. Summary

Summary and Recap

- DMA Transaction -> Transfer -> Moves.
- Transfer is uninterruptible.
- Use shadow address register to update SADR/DADR during a transaction.
- In general, modifying the TCS during a transaction will result in DMA corruption.
- Need consider: longer **transfers** in lower prio channels. This could block a higher prio channel.
- Take into account the interrupt generation rates and system load.

Part of your life. Part of tomorrow.

Infineon