ARM Cortex Core microcontrollers

7th DMA *(Direkt Memory Access)*

Scherer Balázs

Méréstechnika és
Információs Rendszerek
Tanszék

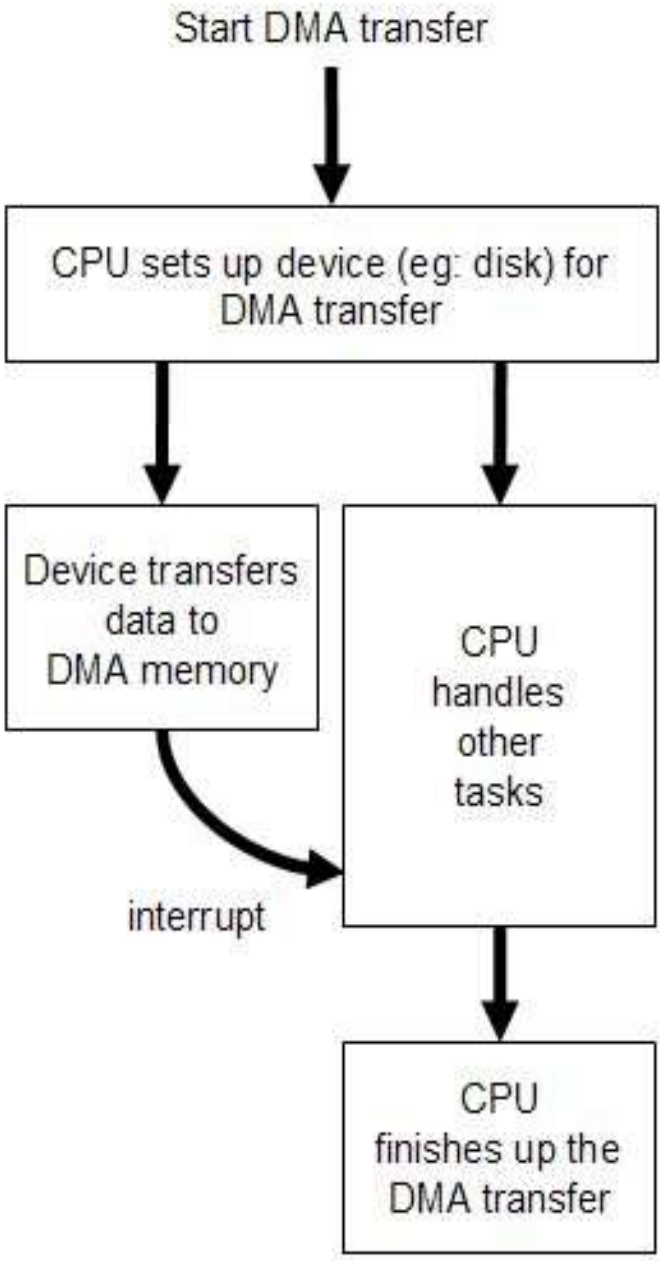
DMA overview I.

Peripheral and memory transfers without interacting with the MCU

- Peripheral – Peripheral
- Memory – Peripheral
- Memory – Memory

- The usage of DMA can reduce the number of interrupts, and also can simplify the hardware by eliminating the need for individual peripheral buffers

DMA overview II.



DMA overview III.

- Every DMA cycle is consist of two phases
  - In case of peripheral to memory transfer
    - Peripheral read
    - Memory write
- The DMA controller do not process the data

DMA controller programing

- Similar to other peripherals. Information to program:
  - Source base address
  - Destination base address
  - Source and Destination address increment
  - Block size
  - Number of cycles
  - IT used to signal transfer end, or half transfer
  - Burst or single phase access
    - Burst is more efficient, but can delay other masters

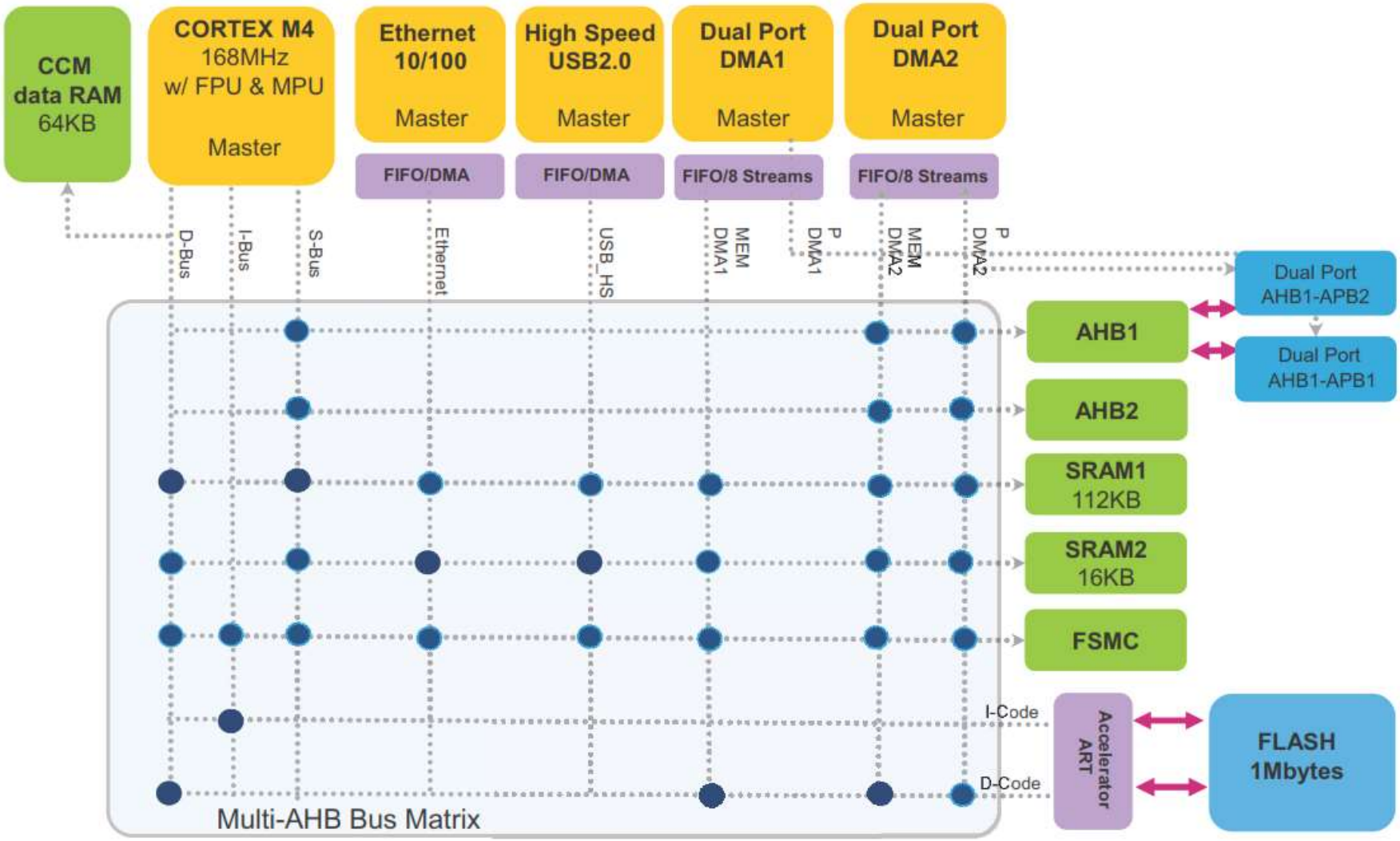
Peripheral – Memory transfer

- AHB transfer reguires 2 cycles at AHB clock rate
- APB transfer also reguires 2 cycles at APB clock frequency, and additional two at AHB clock rate
- Example SPI to memory cycle:

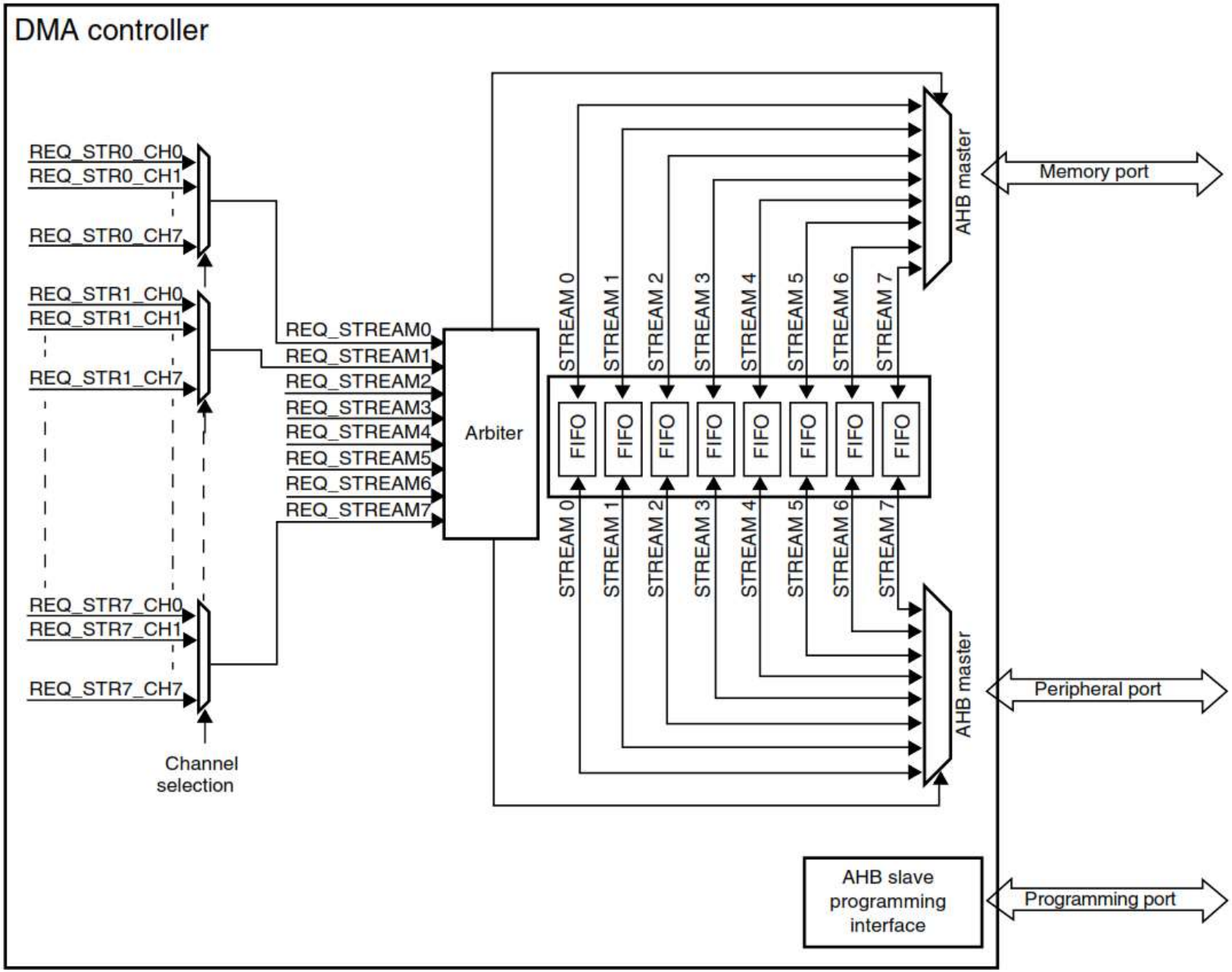
SPI to SRAM DMA transfer = SPI transfer (APB) + SRAM transfer (AHB) + free cycle(AHB)
= (2 APB cycles + 2 AHB cycles) + 2 AHB cycles + 1 AHB cycle
= 2APB Cycles + 5 AHB cycles

DMA in the STM32F2, STM32F4 lines

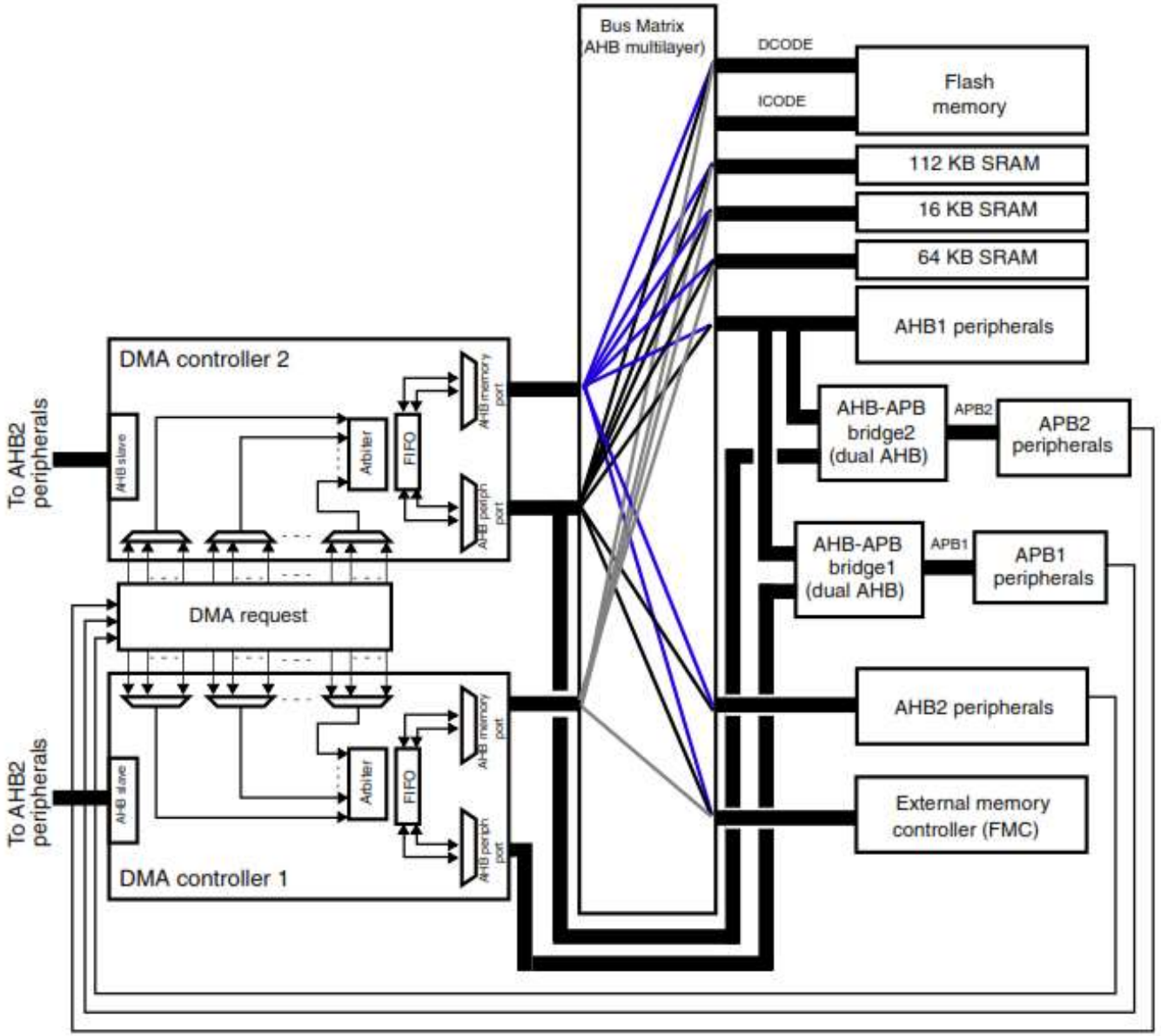
- 2 DMA controller, and many other masters



Internal architecture of the DMA controller



Internal architecture of the DMA controller



DMA channel – peripheral connection predetermined

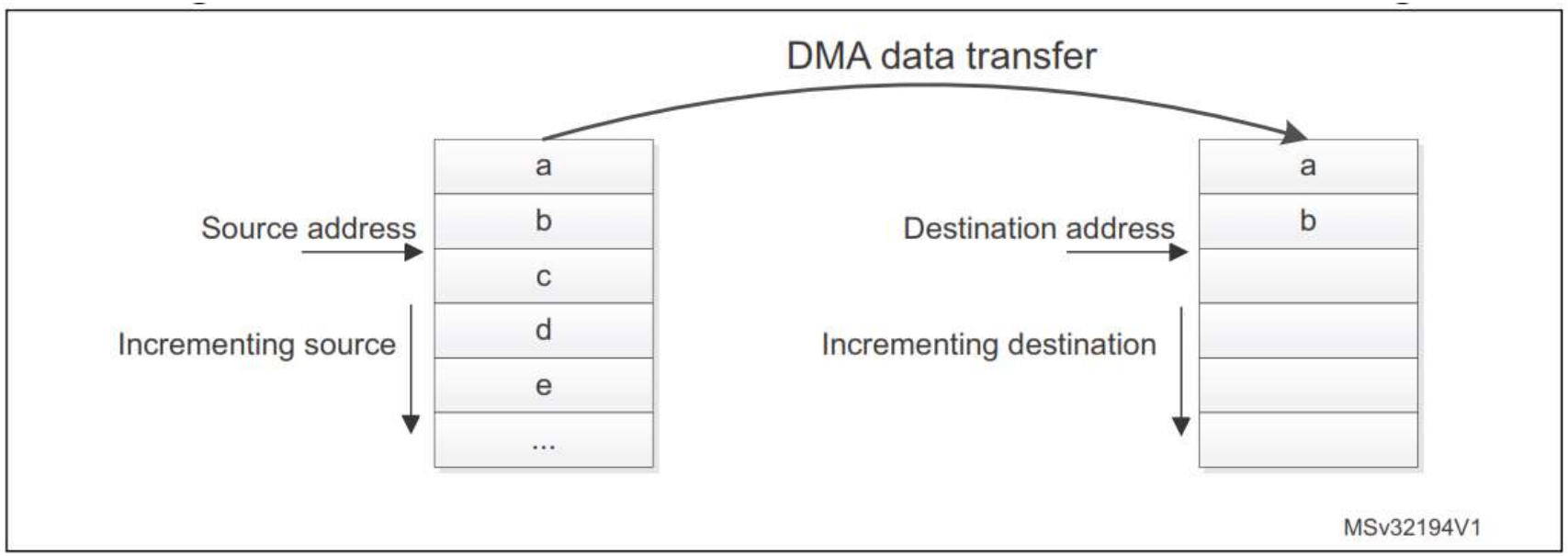
- Care should be taken to select possible paring

Table 2. DMA1 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	SPI3_RX	-	SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX	-	SPI3_TX
Channel 1	I2C1_RX	-	TIM7_UP		TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
Channel 2	TIM4_CH1	-	I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Channel 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Channel 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
Channel 5	UART8_TX(1)	UART7_TX(1)	TIM3_CH4 TIM3_UP	UART7_RX(1)	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX(1)	TIM3_CH3
Channel 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2	-	TIM5_UP	-
Channel 7	-	TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

DMA properties at STM32F4 line

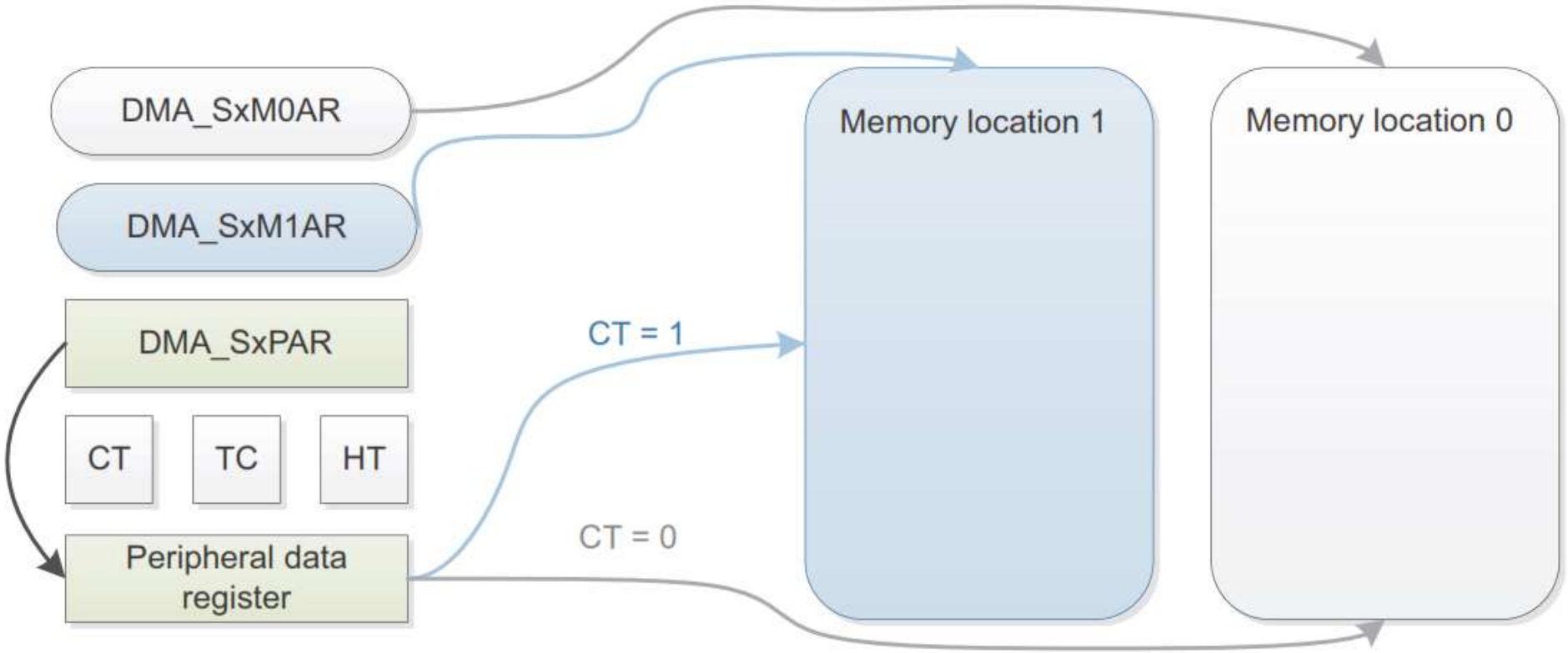
- Address incrementing



- Flexible data width
  - Byte (8-bit)
  - Half-Word (16-bit)
  - Word (32-bit)

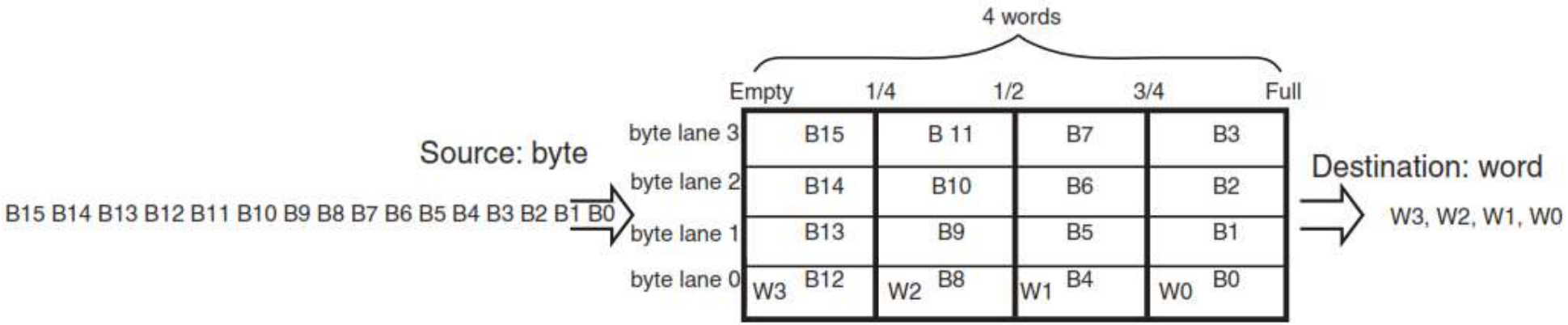
Duoble buffering

- Typical real-time processing feature



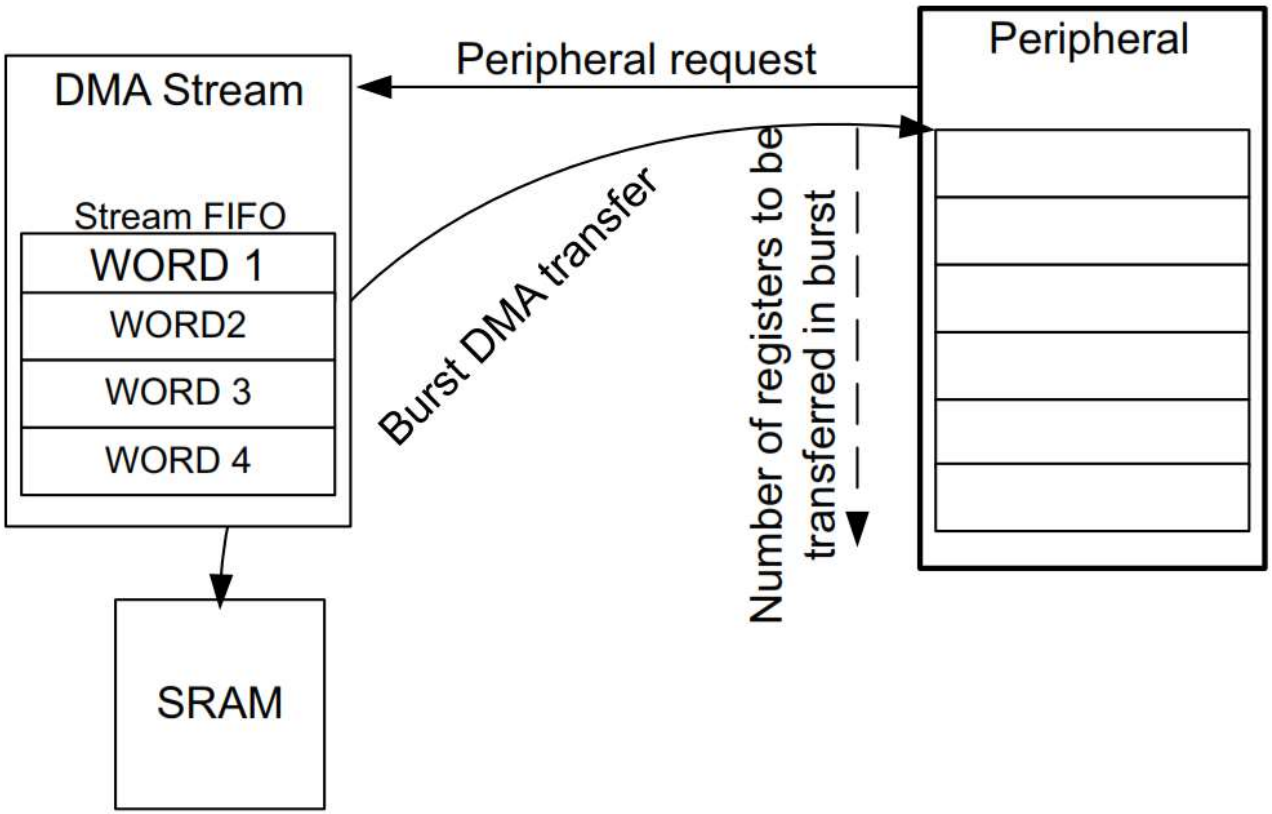
DMA FIFO

- 4 pieces of 32-bit FIFOs
  - Treshold configurable to ¼, ½, ¾ level
  - Packing – Unpacking feature
  - Burst mode transfer



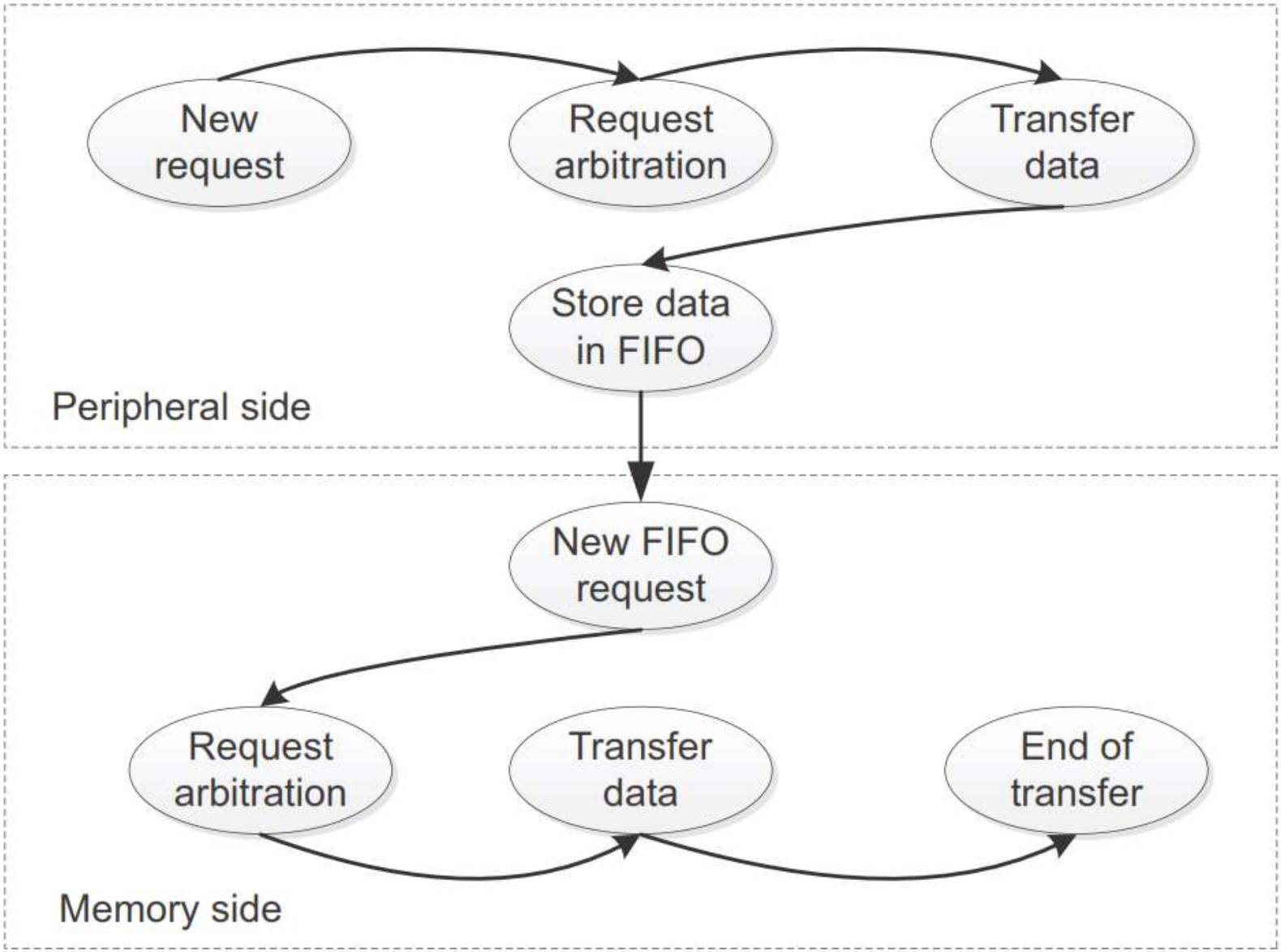
DMA burst

- Configurable maximum to the size of the 4 pieces of 32-bit DMA FIFO
  - 16 byte, 8 half-word, 4 word max.



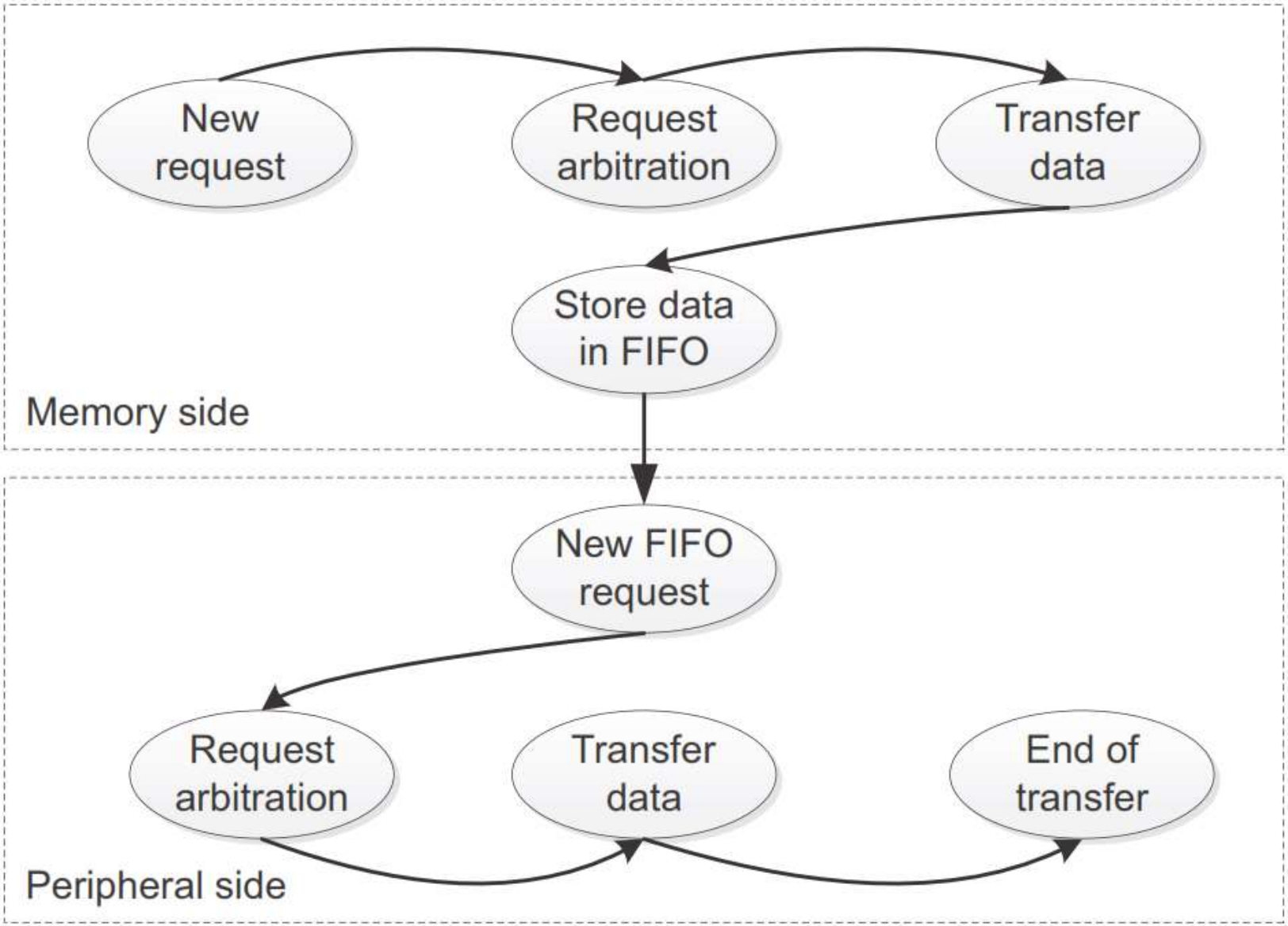
DMA transfer states

- Peripheral – Memory: 2 bus accesses



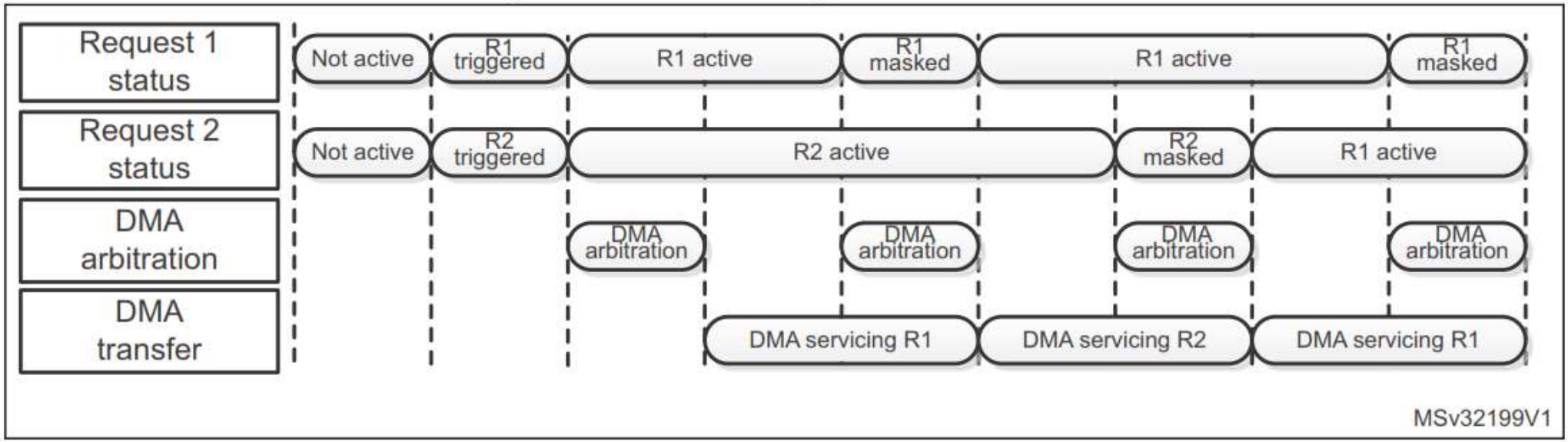
DMA transfer states

- Memory – Peripheral: 2 bus access



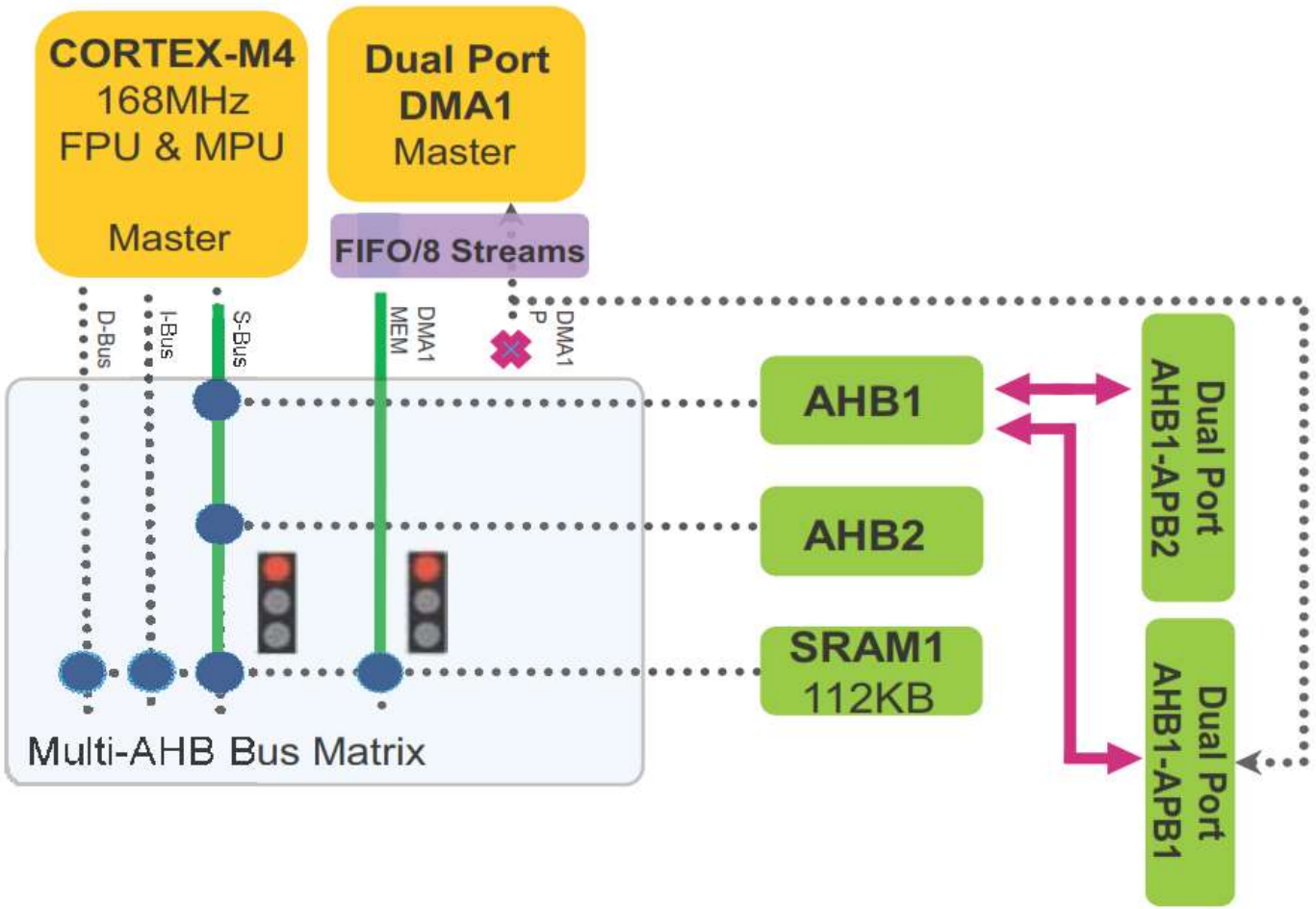
DMA arbitration

- Peripheral – Memory transfer
  - Request 1 is the higher priority one



Bus matrix access

- In case of collision round-robin access method is used
  - SRAM access example



DMA timing

- Access time

$T_S = T_{SP}$ (peripheral access/transfer time) + T_{SM} (memory access/transfer time)

- Peripheral access in detail

$$T_{SP} = t_{PA} + t_{PAC} + t_{BMA} + t_{EDT} + t_{BS}$$

Description	Through bus matrix		DMA's direct paths
	To AHB peripherals	To APB peripherals	
t_{PA}: DMA peripheral port arbitration	1 AHB cycle	1 AHB cycle	1 AHB cycle
t_{PAC}: peripheral address computation	1 AHB cycle	1 AHB cycle	1 AHB cycle
t_{BMA}: bus matrix arbitration (when no concurrency)(1)	1 AHB cycle	1 AHB cycle	N/A
t_{EDT}: effective data transfer	1 AHB cycle(2) (3)	2 APB cycles	2 APB cycle
t_{BS}: bus synchronization	N/A	1 AHB cycle	1 AHB cycle

DMA timing

- Access time

  $T_S = T_{SP}$ (peripheral access/transfer time) + T_{SM} (memory access/transfer time)

- Memory access in detail

  $T_{SM} = t_{MA} + t_{MAC} + t_{BMA} + t_{SRAM}$

Description	Latency
t_{MA}: DMA memory port arbitration	1 AHB cycle
t_{MAC}: memory address computation	1 AHB cycle
t_{BMA}: bus matrix arbitration (when no concurrency)(1)	1 AHB cycle(2)
t_{SRAM}: SRAM read or write access	1 AHB cycle

DMA timing example

- ADC – SRAM transfer
  - Peripheral port delay

AHB/APB2 frequency Transfer time	F_{AHB} = 72 MHz/ F_{APB2} = 72 MHz AHB/APB ratio = 1	FAHB = 144 MHz/ FAPB2 = 72 MHz AHB/APB ratio = 2
t_{PA}: DMA peripheral port arbitration	1 AHB cycle	1 AHB cycle
t_{PAC}: peripheral address computation	1 AHB cycle	1 AHB cycle
t_{BMA}: bus matrix arbitration	N/A(1)	N/A(1)
t_{EDT}: effective data transfer	2 AHB cycles	4 AHB cycles
t_{BS}: bus synchronization	1 AHB cycle	1 AHB cycle
T_{SP}: total DMA transfer time for peripheral port	5 AHB cycles	7 AHB cycles

  - Memory port delay

CPU/APB2 frequency Transfer time	FAHB = 72MHz/ FAPB2=72MHz AHB/APB ratio = 1	FAHB = 144MHz/ FAPB2=72MHz AHB/APB ratio = 2
t_{MA}: DMA memory port arbitration	1 AHB cycle	1 AHB cycle
t_{MAC}: memory address computation	1 AHB cycle	1 AHB cycle
t_{BMA}: bus matrix arbitration	1 AHB cycle(1)	1 AHB cycle(1)
t_{SRAM}: SRAM write access	1 AHB cycle	1 AHB cycle
T_{SM}: total DMA transfer time for memory port	4 AHB cycles	4 AHB cycles

Low-level DMA programming:

UART DMA

Implementation

- USART1 already configured
- USART1 reconfiguring to use DMA

- Identifying properDMA stream, DMA channel
- DMA clock enable
- DMA configuration
- Starting DMA transfer

USART1 reconfiguration for DMA

- USART1 has to signal the DMA that it is ready to send data

- Firmware libry (USART module):
  - LL_USART_EnableDMAReq_TX(USART1);

Identifying DMA Channel, and Stream

- A DMA2-re van szükségünk (AHB1 busz), Channel 4, Stream 7

Table 43. DMA2 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	ADC1	SAI1_A(1)	TIM8_CH1 TIM8_CH2 TIM8_CH3	SAI1_A(1)	ADC1	SAI1_B(1)	TIM1_CH1 TIM1_CH2 TIM1_CH3	-
Channel 1	-	DCMI	ADC2	ADC2	SAI1_B(1)	SPI6_TX(1)	SPI6_RX(1)	DCMI
Channel 2	ADC3	ADC3	-	SPI5_RX(1)	SPI5_TX(1)	CRYP_OUT	CRYP_IN	HASH_IN
Channel 3	SPI1_RX	-	SPI1_RX	SPI1_TX	-	SPI1_TX	-	-
Channel 4	SPI4_RX(1)	SPI4_TX(1)	USART1_RX	SDIO	-	USART1_RX	SDIO	USART1_TX
Channel 5	-	USART6_RX	USART6_RX	SPI4_RX(1)	SPI4_TX(1)	-	USART6_TX	USART6_TX
Channel 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
Channel 7	-	TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX(1)	SPI5_TX(1)	TIM8_CH4 TIM8_TRIG TIM8_COM

- LL_AHB1_GRP1_EnableClock (LL_AHB1_GRP1_PERIPH_DMA2);

DMA Configuration

- DMA2, Channel 4, Stream 7, data with, memory addresses and increment, the target is UART1->DR

```
LL_DMA_InitTypeDef  DMA_InitStruct;

DMA_InitStruct.Channel = LL_DMA_CHANNEL_4;
DMA_InitStruct.Direction = LL_DMA_DIRECTION_MEMORY_TO_PERIPH;
DMA_InitStruct.FIFOMode = LL_DMA_FIFOMODE_DISABLE;
DMA_InitStruct.MemBurst = LL_DMA_MBURST_SINGLE;
DMA_InitStruct.MemoryOrM2MDstAddress = (uint32_t)buffer;
DMA_InitStruct.MemoryOrM2MDstDataSize = LL_DMA_MDATAALIGN_BYTE;
DMA_InitStruct.MemoryOrM2MDstIncMode = LL_DMA_MEMORY_INCREMENT;
DMA_InitStruct.Mode = LL_DMA_MODE_NORMAL;
DMA_InitStruct.NbData = strlen(buffer);
DMA_InitStruct.PeriphBurst = LL_DMA_PBURST_SINGLE;
DMA_InitStruct.PeriphOrM2MSrcAddress = (uint32_t)&(USART1->DR);
DMA_InitStruct.PeriphOrM2MSrcDataSize = LL_DMA_PDATAALIGN_BYTE;
DMA_InitStruct.PeriphOrM2MSrcIncMode = LL_DMA_PERIPH_NOINCREMENT;
DMA_InitStruct.Priority = LL_DMA_PRIORITY_LOW;

LL_DMA_Init  (DMA2,LL_DMA_STREAM_7, &DMA_InitStruct);
```

Starting the DMA

- After configuration only a start is needed

```
LL_DMA_EnableStream  (DMA2, LL_DMA_STREAM_7);
```