Direct Memory Access

Last updated 4/29/20

DMA

- Consider
  - Need to load a page from disk to main memory
  - Need to transfer an MP3 file from memory to an audio block
  - Need to copy a file from your computer to a flash drive
  - All of these transactions need to be managed in some way

DMA

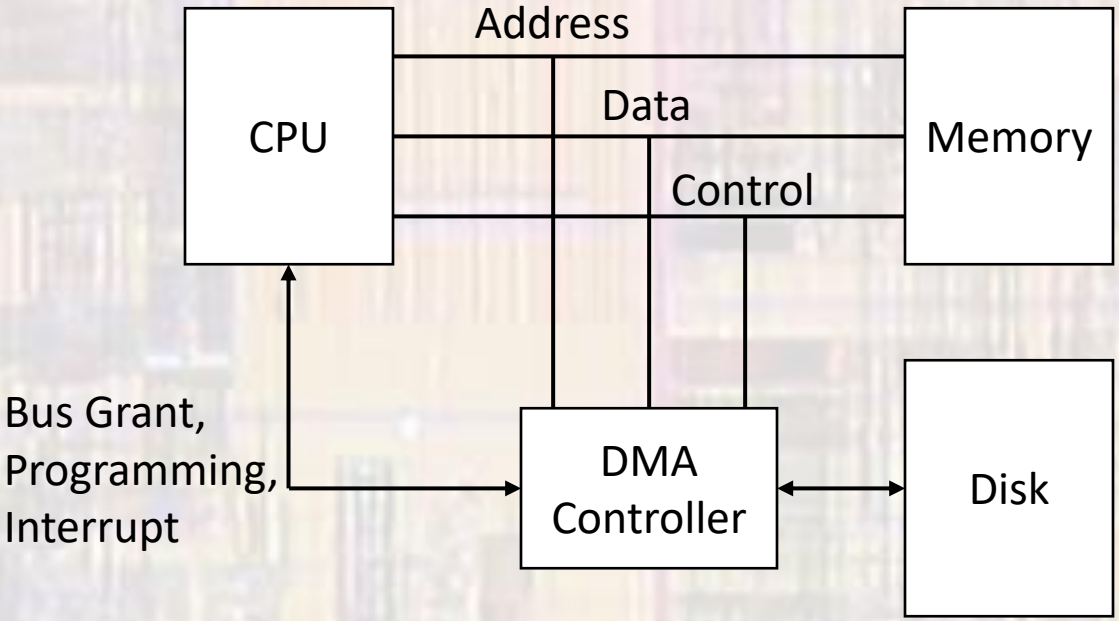
- Data transfer
  - Use the CPU to transfer the data
  - CPU reads a word from the hard drive
  - CPU writes a word to main memory
  - Repeat
  - If the transfers are fast – the CPU cannot accomplish anything else
  - If the transfers are slow, the CPU will constantly be changing it's context in response to interrupts

Request word
DO something else
Interrupt – word ready
Get word and submit write

Do something else
Interrupt – write complete
Submit request for next word
…

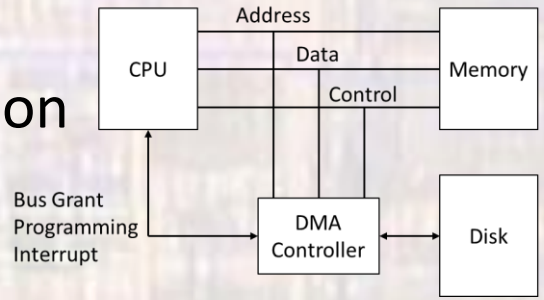
DMA

- DMA
  - Remove the CPU from the ongoing operation



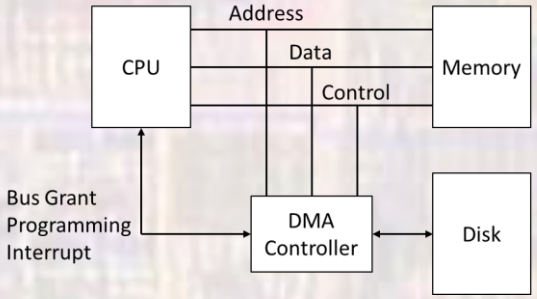
DMA

- DMA
  - Remove the CPU from the ongoing operation
    - CPU tells the DMAC where to get the data
    - CPU tells the DMAC how much data to get
    - CPU tells the DMAC where to put the data
    - CPU grants the DMAC control of the bus
      - Meanwhile the CPU operates on registers, cache data, …



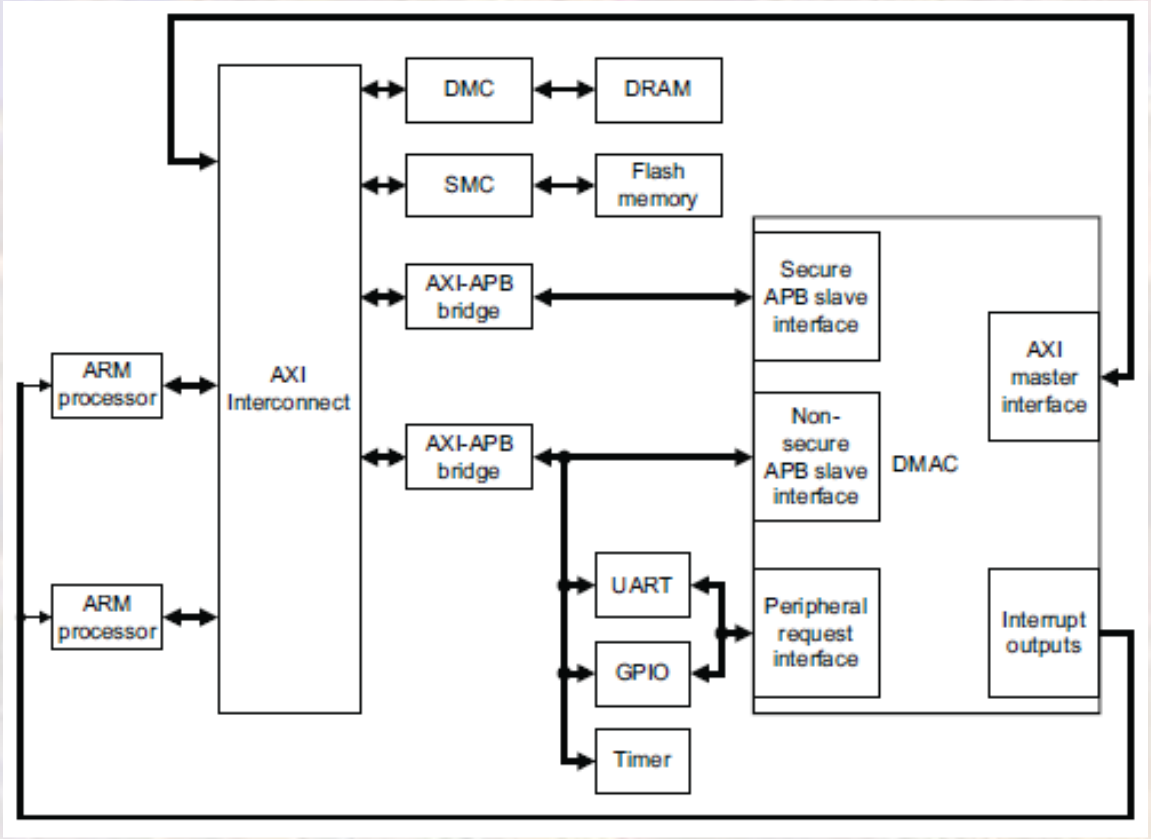
DMA

- DMA
  - Transfer modes
    - Burst mode
      - Once the DMA has control of the bus – it keeps it until transfer is complete
    - Cycle stealing mode
      - DMA interleaves requests with CPU 1-1 1-2 1-4, …
    - Transparent mode
      - CPU reclaims the bus any time it needs it
      - DMA remembers where it was and continues as soon as the CPU releases the bus



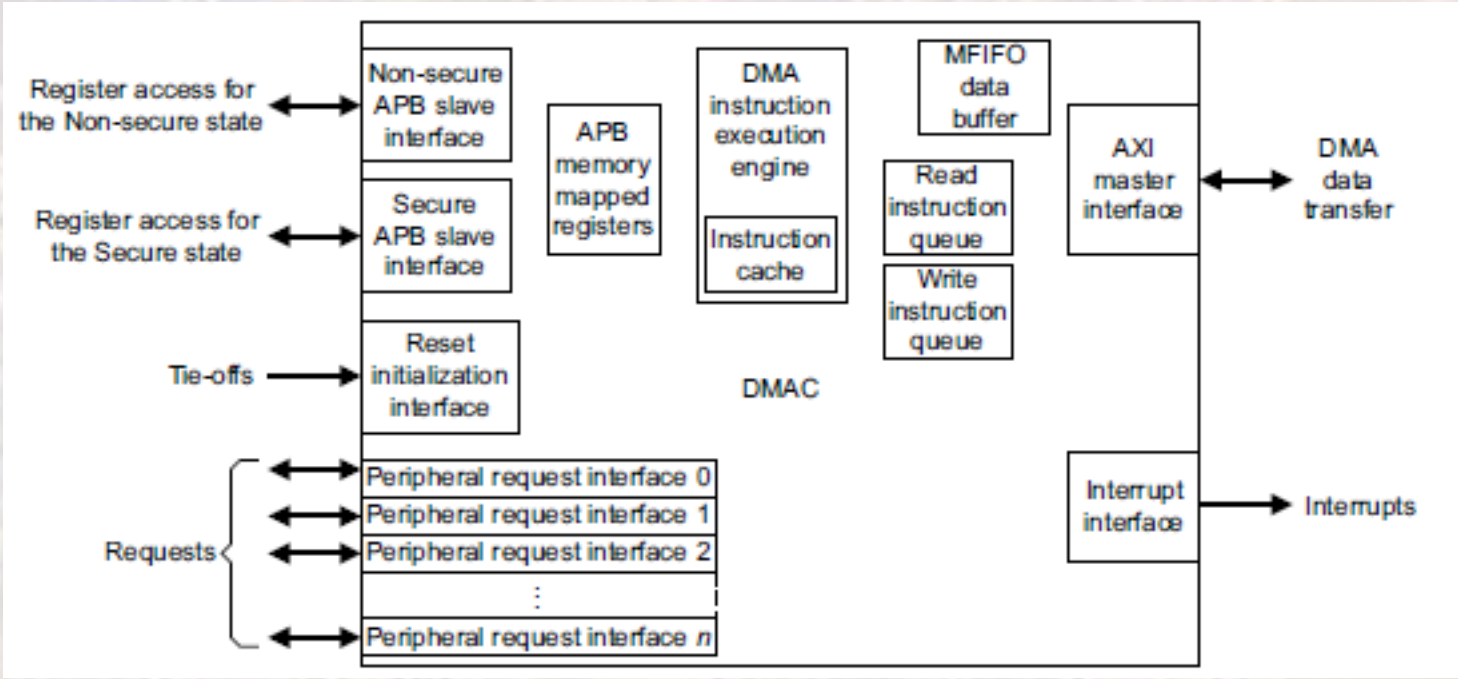
DMA

- DMA – Modern example



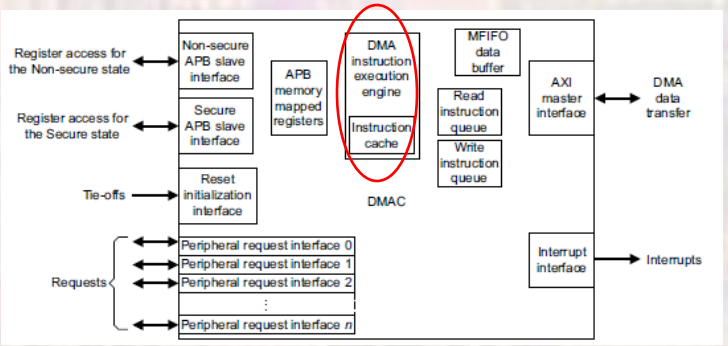
DMA

- DMA – Modern example



DMA

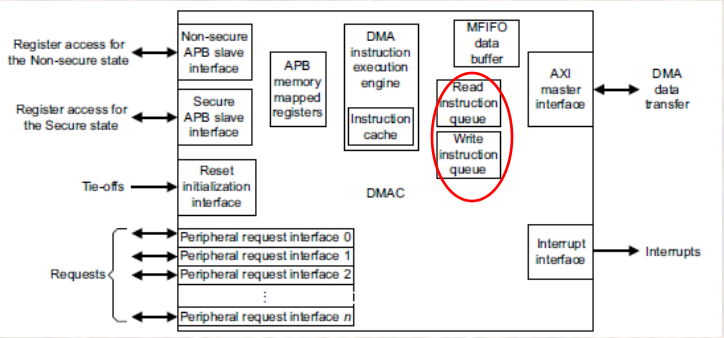
- DMA – Modern example



  - Instruction Execution Engine
    - Reads instructions from its cache
    - Supports a separate PC for each DMA channel
      - Up to 8 DMA channels → 8 concurrent threads
    - 1 thread for management
    - Executes 1 management instruction then 1 channel instruction
      - Changes channel threads each cycle (round robin)

DMA

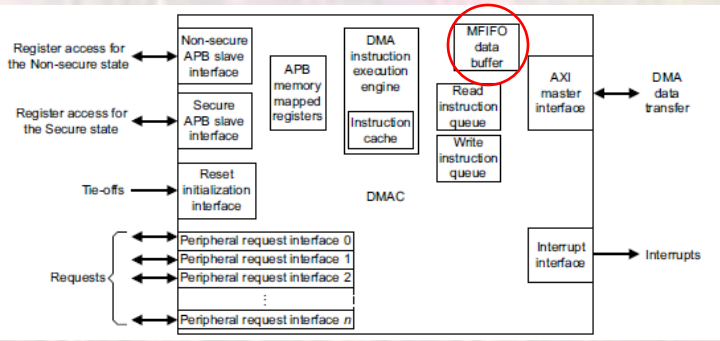
- DMA – Modern example



  - Read/Write instruction queues
    - Load instructions are written to the read instruction queue
    - Store instructions are written to the write instruction queue
    - Instructions then execute out of the queue via the AXI protocol

DMA

- DMA – Modern example



  - Multi-FIFO
    - Buffers data in/out for each channel
    - Single FIFO with width matched to bus width
      - Can pack and unpack narrower data
      - E.g. 32 bit width can hold 2 16 bit data values
    - Individual channels can be variable depth
    - Sum of all channels must not exceed physical size

DMA

- DMA – Instruction Set

Mnemonic	Instruction	Thread usage: • M = DMA manager • C = DMA channel	
DMAADDH	Add Halfword	-	C
DMAADNH	Add Negative Halfword	-	C
DMAEND	End	M	C
DMAFLUSHP	Flush and Notify Peripheral	-	C
DMAGO	Go	M	-
DMAKILL	Kill	M	C
DMALD	Load	-	C
DMALDP	Load and Notify Peripheral	-	C
DMALP	Loop	-	C
DMALPEND	Loop End	-	C
DMALPFE	Loop Forever	-	C
DMAMOV	Move	-	C
DMANOP	No operation	M	C
DMARMB	Read Memory Barrier	-	C
DMASEV	Send Event	M	C
DMAST	Store	-	C
DMASTP	Store and Notify Peripheral	-	C
DMASTZ	Store Zero	-	C
DMAWFE	Wait For Event	M	C
DMAWFP	Wait For Peripheral	-	C
DMAWMB	Write Memory Barrier	-	C

DMA

- DMA – Instruction Set
  - DMALP lcx #
  - DMALPEND
    - Loop instructions
    - lcx – loop counter x, 0-8
    - # - number of iterations, 1-256

DMA

- DMA – Instruction Set

  - DMALD
  - DMAST

    - LD – loads values from the peripheral into the MFIFO using the AXI
      tags the MFIFO entry with the correct channel #

    - ST – Stores values into the peripheral from the MFIFO using the AXI
      tags the MFIFO entry with the correct channel #

DMA

- DMA – Instruction Set
  - DMAMOV REG #
    - Load immediate value # into register REG
  - DMAEND
    - Transfer is complete

DMA

- DMA – Key Registers
  - Source address registers
    - Used for loads
    - 32 bit addressing

0x400	SAR0	Source address for DMA channel 0
0x420	SAR1	Source address for DMA channel 1
0x440	SAR2	Source address for DMA channel 2
0x460	SAR3	Source address for DMA channel 3
0x480	SAR4	Source address for DMA channel 4
0x4A0	SAR5	Source address for DMA channel 5
0x4C0	SAR6	Source address for DMA channel 6
0x4E0	SAR7	Source address for DMA channel 7

DMA

- DMA – Key Registers
  - Destination address registers
    - Used for stores
    - 32 bit addressing

0x404	DAR0	Destination address for DMA channel 0
0x424	DAR1	Destination address for DMA channel 1
0x444	DAR2	Destination address for DMA channel 2
0x464	DAR3	Destination address for DMA channel 3
0x484	DAR4	Destination address for DMA channel 4
0x4A4	DAR5	Destination address for DMA channel 5
0x4C4	DAR6	Destination address for DMA channel 6
0x4E4	DAR7	Destination address for DMA channel 7

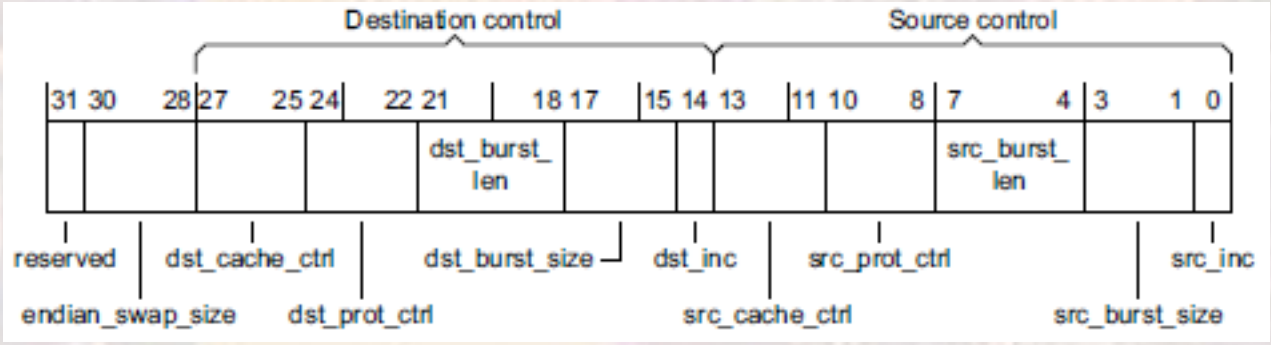
DMA

- DMA – Key Registers
  - Channel control registers
    - Used for setting up transactions

0x408	CCR0	Channel control for DMA channel 0
0x428	CCR1	Channel control for DMA channel 1
0x448	CCR2	Channel control for DMA channel 2
0x468	CCR3	Channel control for DMA channel 3
0x488	CCR4	Channel control for DMA channel 4
0x4A8	CCR5	Channel control for DMA channel 5
0x4C8	CCR6	Channel control for DMA channel 6
0x4E8	CCR7	Channel control for DMA channel 7

DMA

- DMA – Key Registers
  - Channel control registers



DMA

- DMA – Key Registers
  - Channel control registers – assembler directives

Syntax	Description	Options	Default
SA	Source address increment. Sets the value of **ARBURST[0]**.	I = Increment F = Fixed	I
SS	Source burst size in bits. Sets the value of **ARSIZE[2:0]**.	8, 16, 32, 64, or 128	8
SB	Source burst length. Sets the value of **ARLEN[3:0]**.	1 to 16	1
SP	Source protection.	0 to 7[a]	0
SC	Source cache.	0 to 15[ab]	0
DA	Destination address increment. Sets the value of **AWBURST[0]**.	I = Increment F = Fixed	I
DS	Destination burst size in bits. Sets the value of **AWSIZE[2:0]**.	8, 16, 32, 64, or 128	8
DB	Destination burst length. Sets the value of **AWLEN[3:0]**.	1 to 16	1
DP	Destination protection.	0 to 7[a]	0
DC	Destination cache.	0 to 15[ac]	0
ES	Endian swap size, in bits.	8, 16, 32, 64, or 128	8

Limited to 16 byte beats

Limited to 16 beat bursts

DMA

- Example 1

  - Write a code snippit to transfer 8, 32 bit words from memory mapped location 0x1000 to location 0x2000. (assume a 32bit AXI data bus)

```
DMAMOV CCR SB8 SS32 SA1     // set up 32 bit beat, 8 beat burst at source, incrementing
DMAMOV CCR DB8 DS32 DA1     // set up 32 bit beat, 8 beat burst at destination, incrementing
DMAMOV SAR 0x1000           // set source address
DMAMOV DAR 0x2000           // set destination address

DMALD                       // load from 0x1000 to MFIFO
DMAST                       // store from MFIFO to 0x2000
```

DMA

- Example 2

  - Write a code snippit to transfer 128, 32 bit words from memory mapped location 0x4000 to location 0x5000. (assume a 32bit AXI data bus)

Note – max burst length is 16 beats

```
DMAMOV CCR SB8 SS32 SA1      // set up 32 bit beat, 8 beat burst at source, incrementing
DMAMOV CCR DB8 DS32 DA1      // set up 32 bit beat, 8 beat burst at destination, incrementing
DMAMOV SAR 0x4000            // set source address
DMAMOV DAR 0x5000            // set destination address

DMALP lc2 16                 // set up to do 16 transfers
  DMALD                      // load from 0x4000 to MFIFO
  DMAST                      // store from MFIFO to 0x5000
DMALPEND
```

DMA

- Examples 1 and 2
  - How much of the MFIFO was used in example 1
    - 8 words
  - How much of the MFIFO was used in example 2
    - 8 words