DSP™
TEXAS INSTRUMENTS TECHNOLOGY

DMA
The Hidden Key to DSP Systems

Steve Krueger

DSP Architecture Group

Dallas

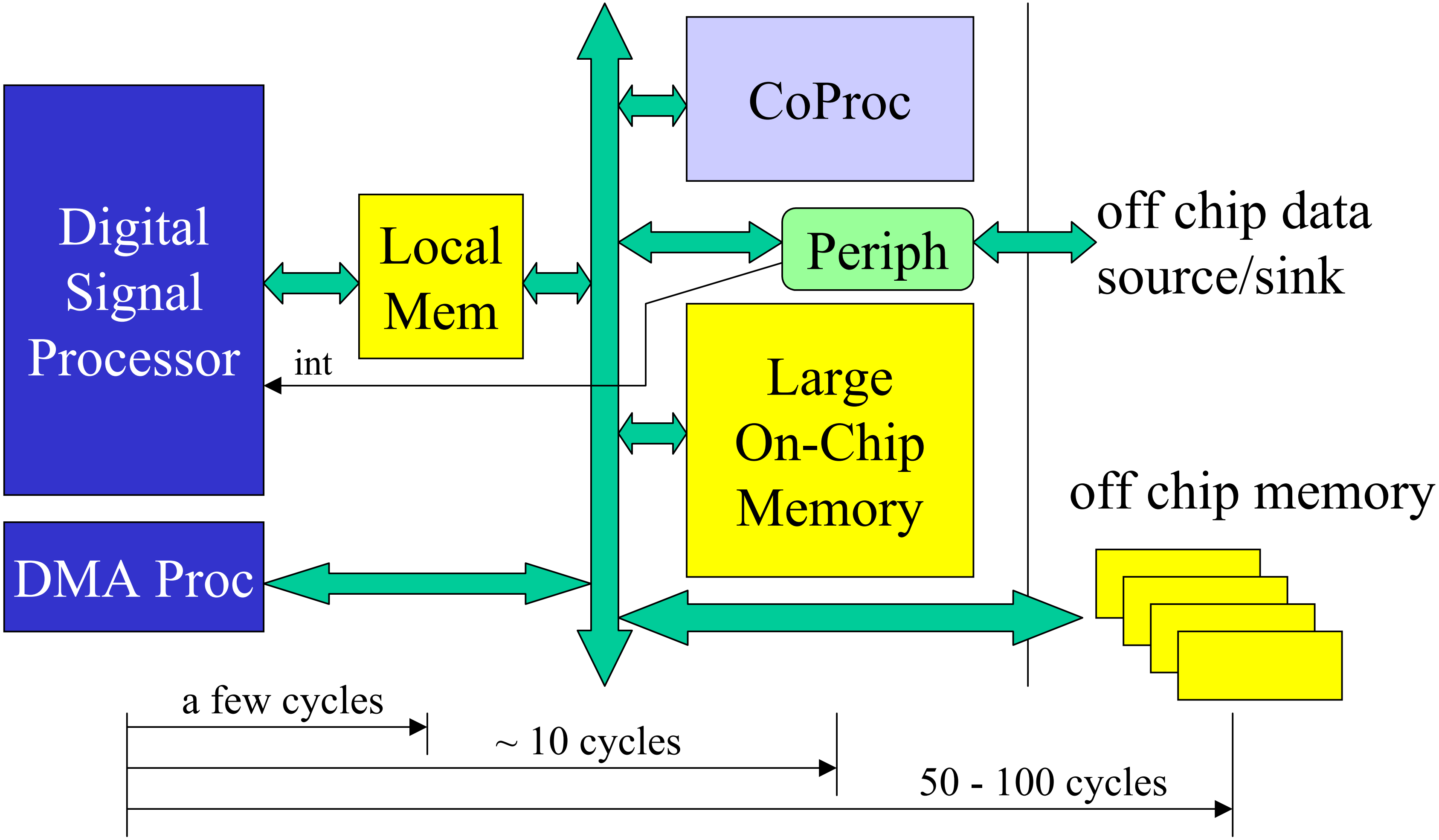
What Do DSP Systems Do?

- Process
  - filter, scale, transform, encode/decode, correlate, etc.
  - these operations are processor and data intensive
- Signals
  - generally continuous or nearly continuous streams of sampled real-world data
  - usually, real-time

Impediments to Processing

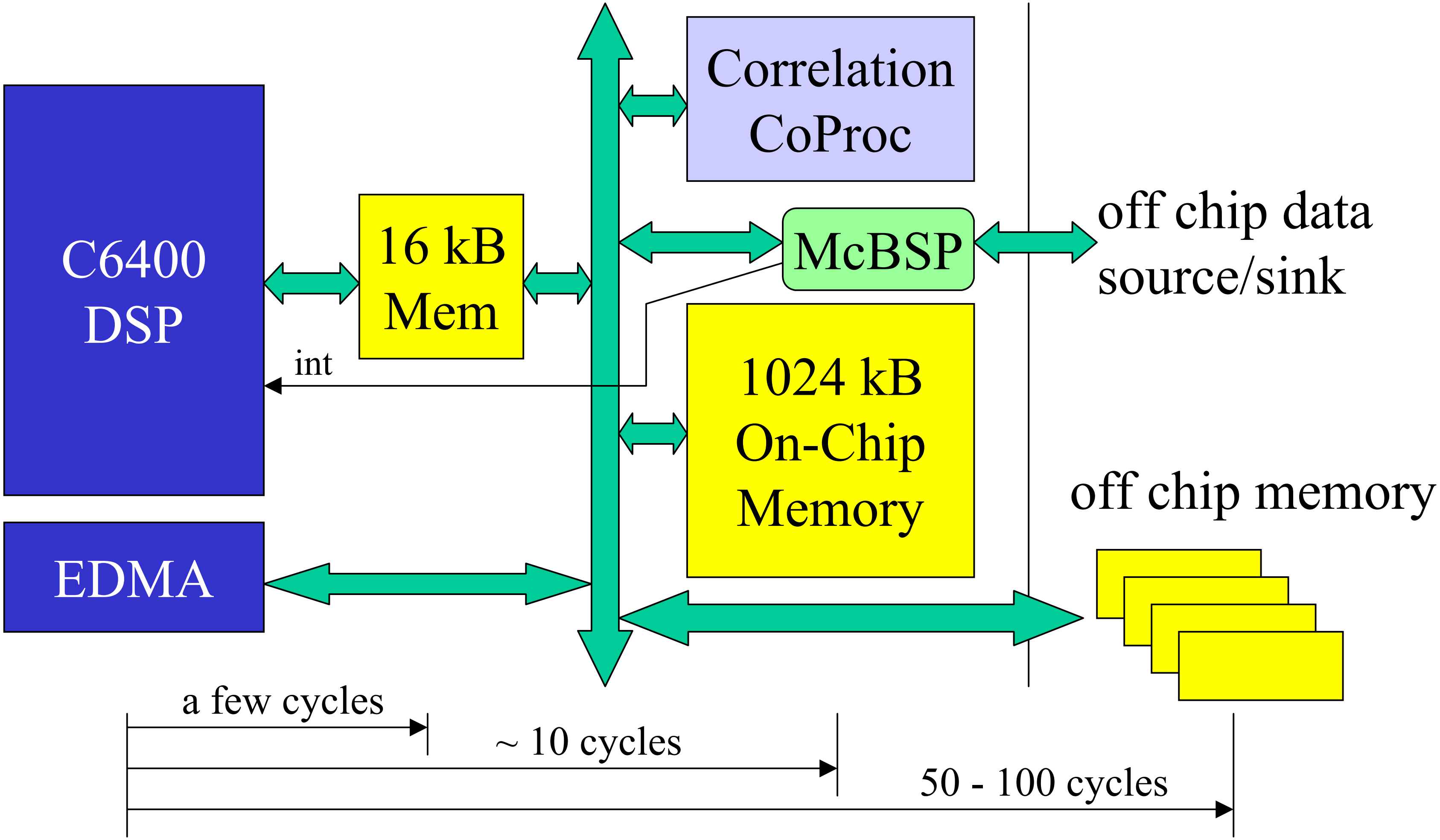
- Can’t process until data is received (often a whole block of data)
- Most algorithms have many memory accesses
- Memory access latency increases with memory size and distance from processor
- Some operations too specialized for general purpose DSP

Model System



Memory Latency

C6400 System



Memory Latency

DMA Solves Two Problems

- I/O interrupt loading slows processor work on tasks
- Data in close memory avoids latency for higher performance

Problem of Interrupt Loading

- Each peripheral usually interrupts to indicate data ready.
- Processor takes interrupt by entering an interrupt handler, transferring the data, then resuming the interrupted processing.
- This interrupt sequence takes at least 20 cycles on most modern architectures — many more on some.

Interrupts of High Speed Peripheral

- A high-speed peripheral can generate a lot of interrupts:
  - say you have a 1 Msps ADC that produces an 8-bit conversion every microsecond.
  - If you interrupt for every conversion, have 1 million interrupts per second.
  - If the interrupt takes 30 cycles, this is using 5% of a C6416 at 600 MHz.
  - What if your system needs 8 of these?

Solutions to Interrupt Problem

- Can buffer at the peripheral and interrupt less frequently. Might interrupt once per 32 conversions
  - increases complexity and cost of peripheral, adds latency to peripheral data, more complex error handling
- Can make peripheral a bus master and have it store data into memory.
  - Increases the complexity and cost of peripheral

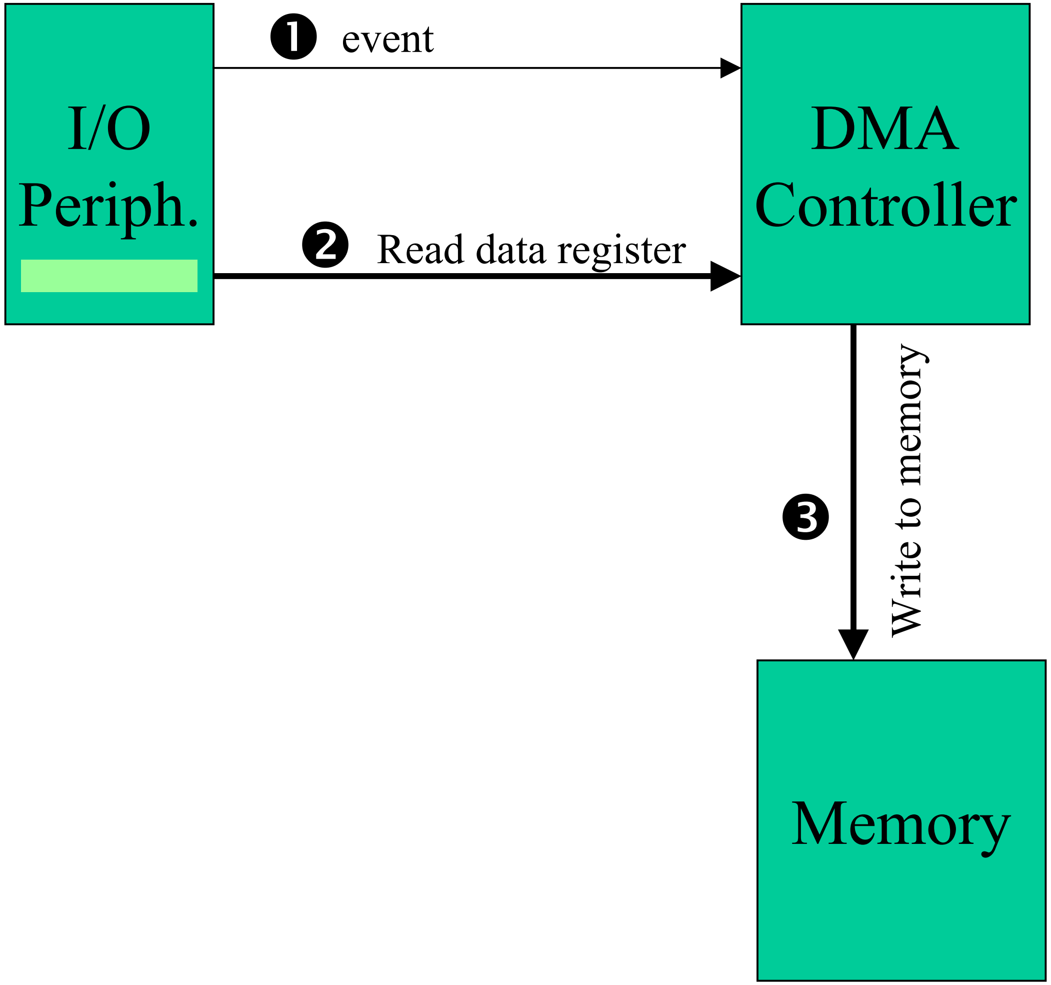
Solutions — continued

- Can have a central bus master to service any peripheral
  - keeps peripherals simple and shares cost between all peripherals
  - This is the basic idea of DMA (Direct Memory Access)

DMA Terms

- **Event** - a hardware signal that initiates a transfer on a DMA channel
- **Channel** - one thread of transfer. It contains source and destination addresses plus control information
- **Element**: 8-, 16-, 32-bit datum
- **Frame**: Group of elements
- **Array**: Group of contiguous elements
- **Block**: Group of Frames or Arrays

Peripheral DMA Flow

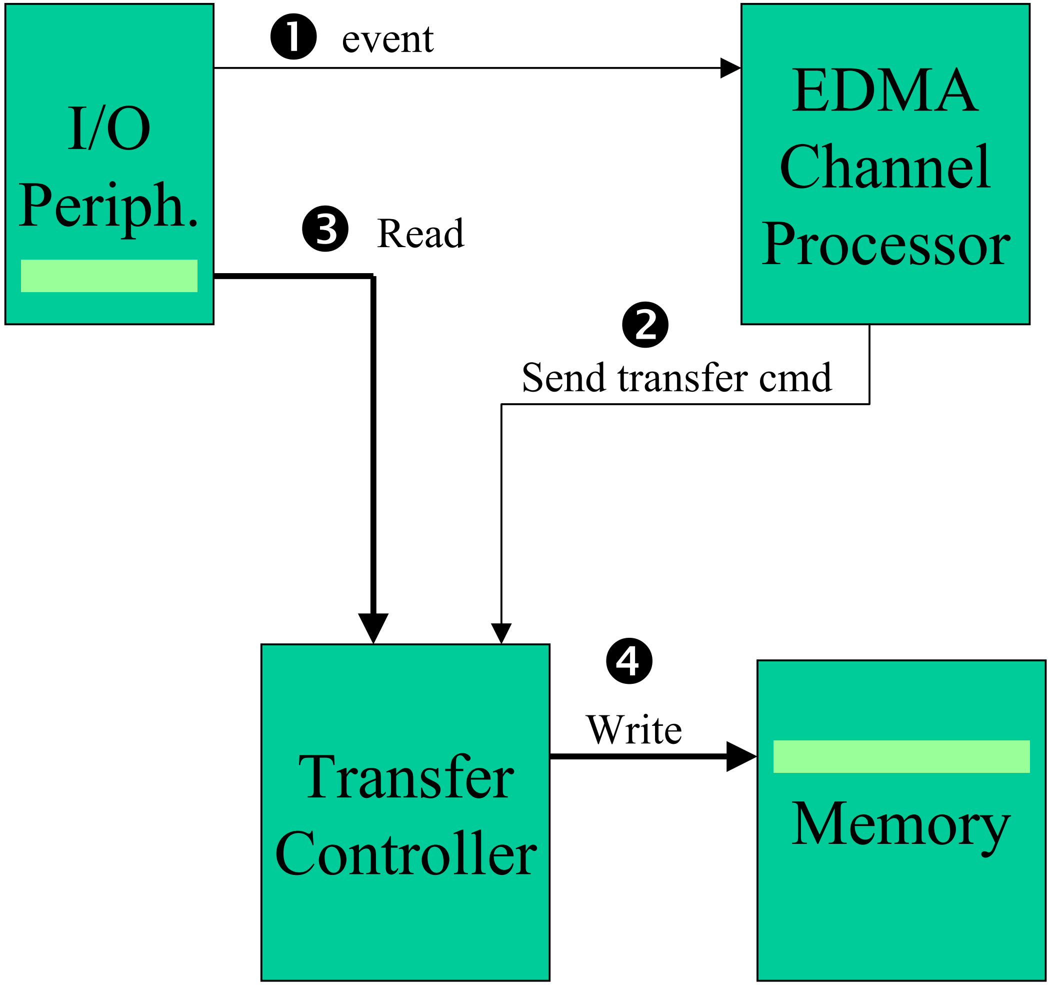


❶ Peripheral signals an event to indicate it has data ready.

❷ DMA Controller reads peripheral data

❸ DMA Controller writes data to memory

TI EDMA Peripheral DMA Flow



❶ Peripheral signals an event to indicate it has data ready.

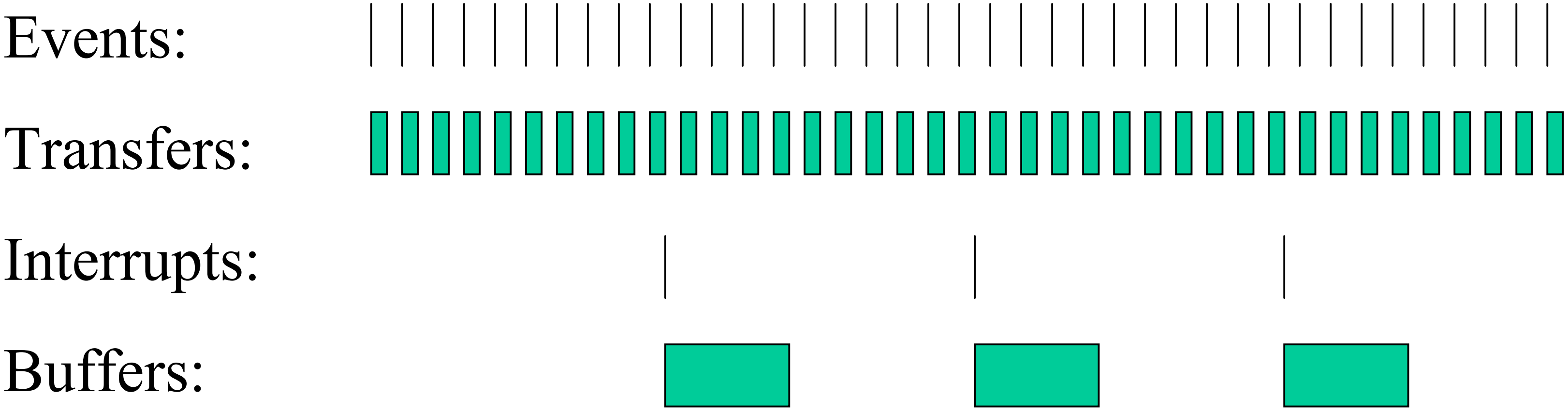
❷ EDMA Controller sends a data transfer command to data move engine

❸ Transfer Controller reads data register

❹ Transfer Controller writes memory

Interrupt Reduction

- DMA processes I/O events to accumulate buffers for processing
- Reduces DSP interrupts by 10-1000 times and increases tolerable interrupt latency by a similar factor



Events:

Transfers:

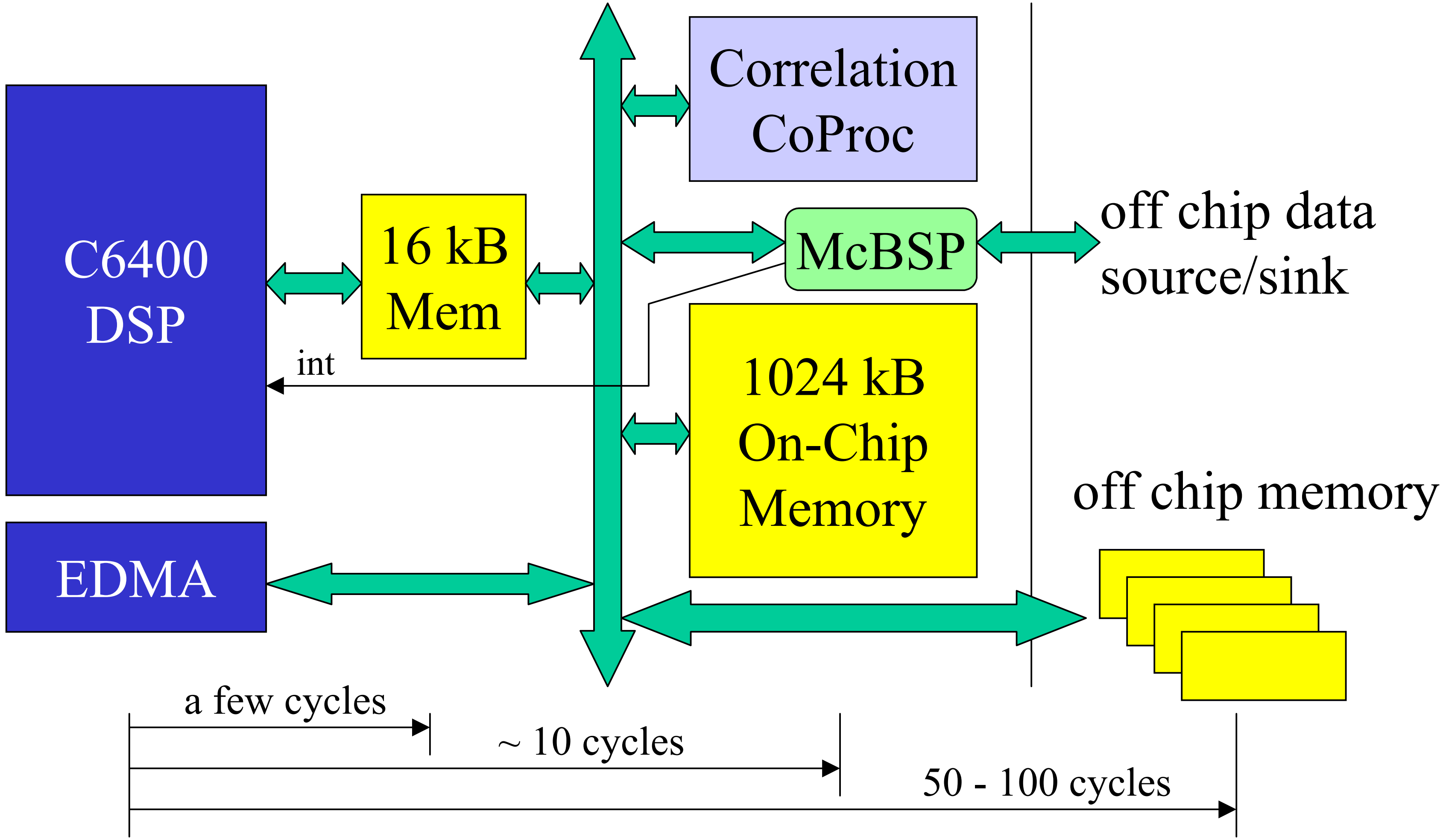
Interrupts:

Buffers:

Memory Latency Reduction

- Program’s performance is reduced when accessing long latency memories.
- Two ways of implementing processors:
  - overall stall - all processor activity waits
  - dependence stall - all dependent operations wait

C6400 System



Memory Latency

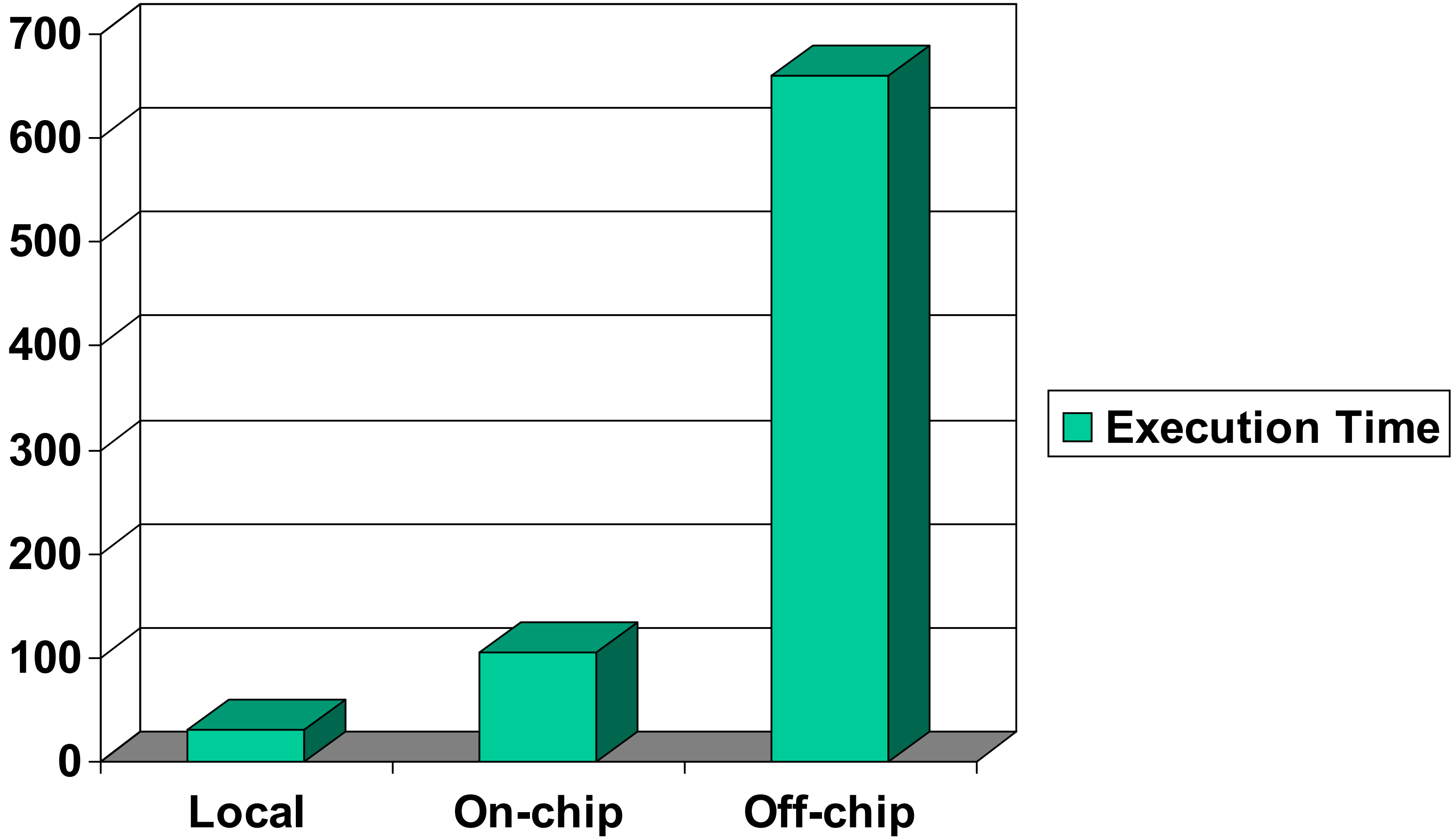
Cost of Dependence Stall

- Dependence stall is expensive to implement. These are the techniques of out-of-order CPU architecture.
- Must examine large “window” of instructions to find those (few) that aren’t dependent on any pending operation.
- Must keep a time-dependent set of processor state (GPRs) so each instruction can run in proper environment.
- Must discard or rewind state on exception or other interruption.

Memory Stalls in DSP Systems

- Dependence stall (out-of-order) techniques are too expensive for DSP systems.
- They also make it harder to predict the real-time behavior.
- No DSPs have used these techniques.
- Instead, all DSPs have used total stall during memory latency.
- Inexpensive but can hurt performance.

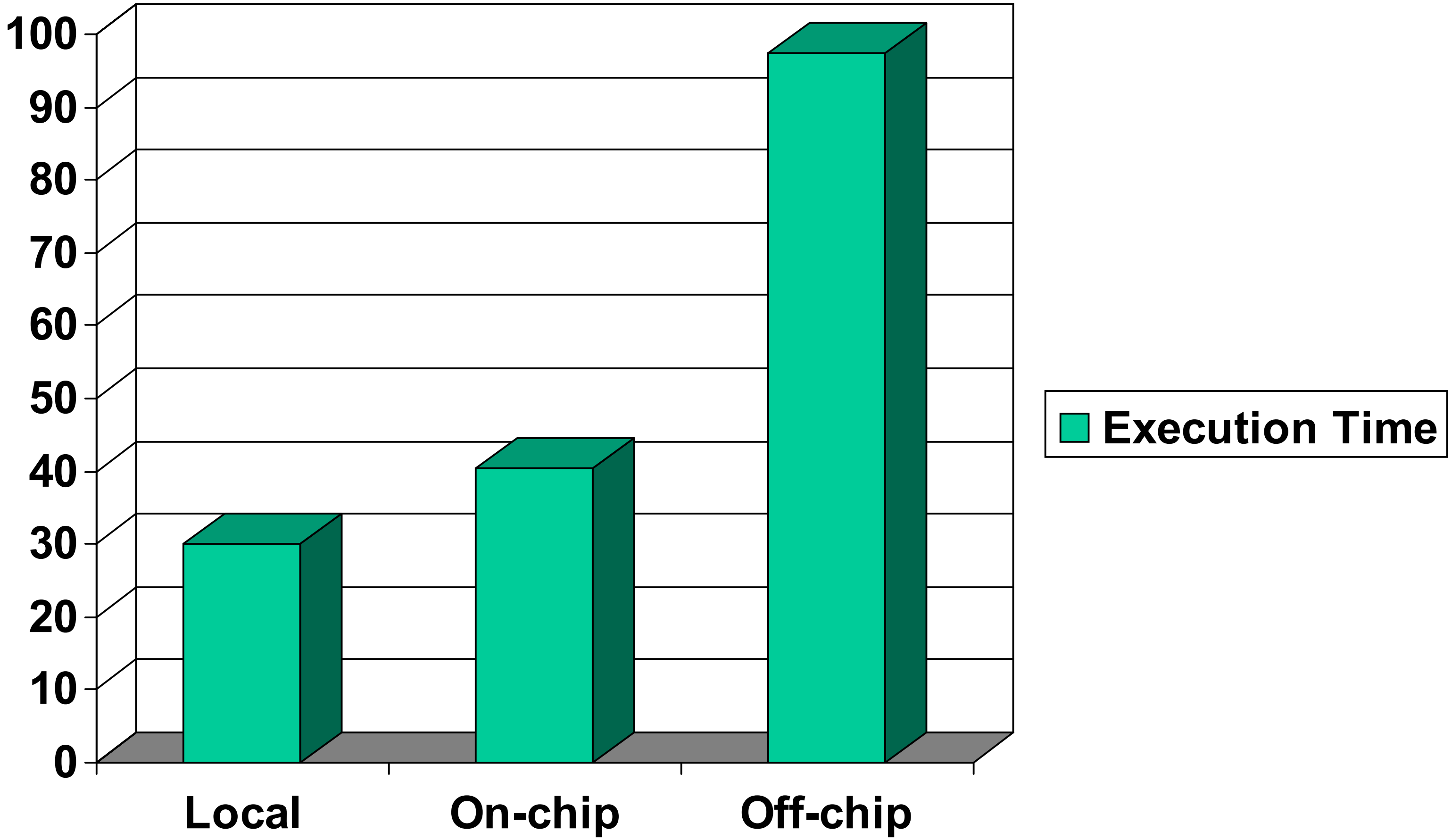
Effects of Memory Latency



Cache Memory

- Cache memory is well known to address the problem of keeping frequently accessed information in a local memory for low-latency access.
- But cache has problems in some DSP applications:
  - Data isn’t always reused, or has limited reuse
  - Cache only loads data when requested the first time
  - Cache line size may not be a good match for data size

Effects of Cache Memory



Addressible Local RAM

- Most newer TI DSPs have selection facilities that allow local RAM to be either cache memory or addressible RAM.
- If a programmer is clever and uses DMA, local RAM can be very effective in many DSP applications, achieving near ideal processing rates.

Stream Processing

- Data supply is an infinite (or practically infinite) stream.
- The arrival of data is predictable.
- Each datum or block of data is processed the same.
- The processing is very regular and predictable.

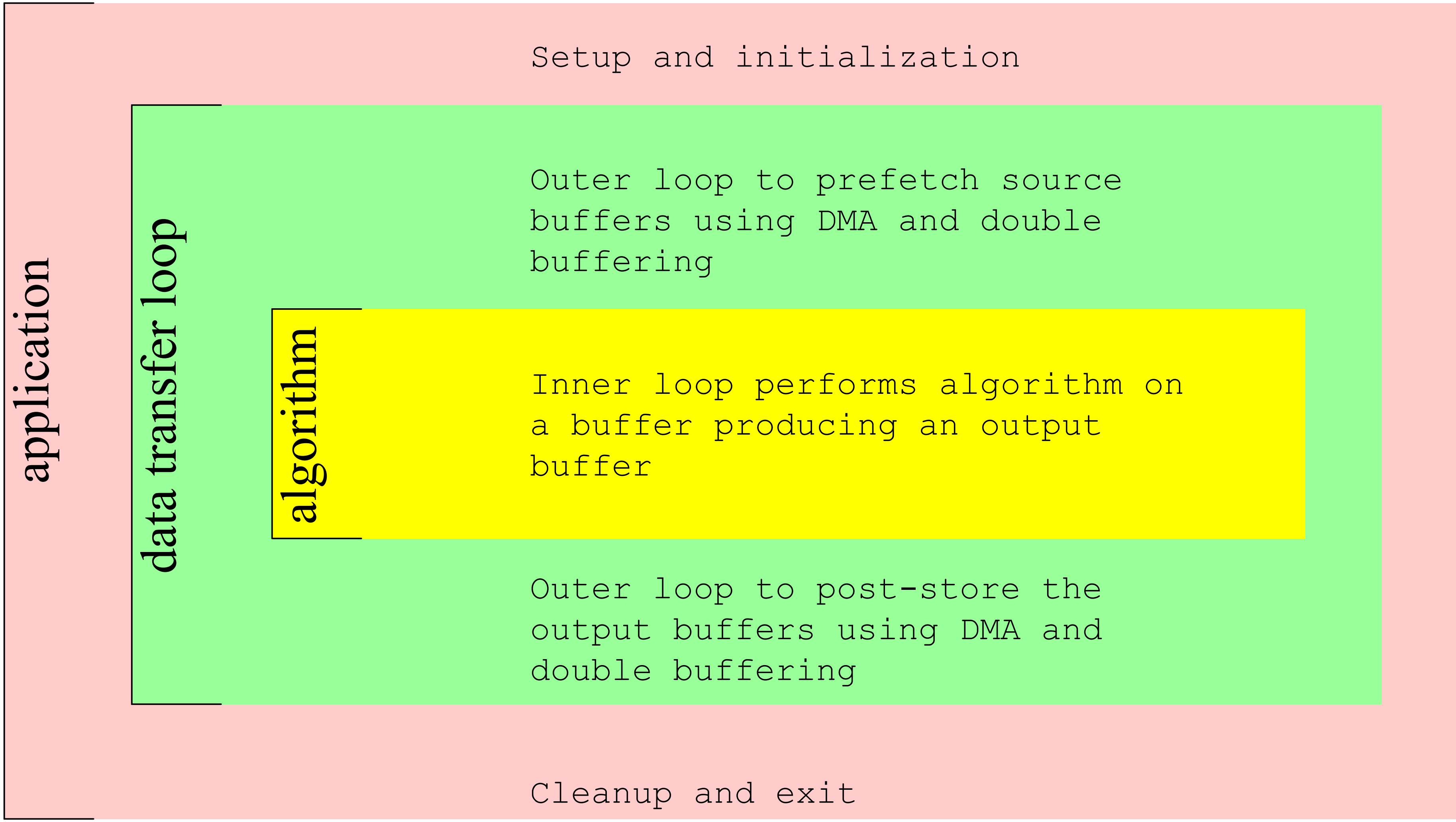
A problem with these characteristics allows for accurate prediction of which data will be needed and when.

Basic Plan

- Block algorithm to process buffers of input samples
- Wrap buffer processing loop with a data transfer loop.
- Perform transfers in parallel with processing.
- Work inbound transfers ahead of processing so will complete before needed.
- Perform outbound transfers once processing complete.

This is a job for DMA!

Streaming Prefetch Loop Nest



application

Setup and initialization

data transfer loop

Outer loop to prefetch source buffers using DMA and double buffering

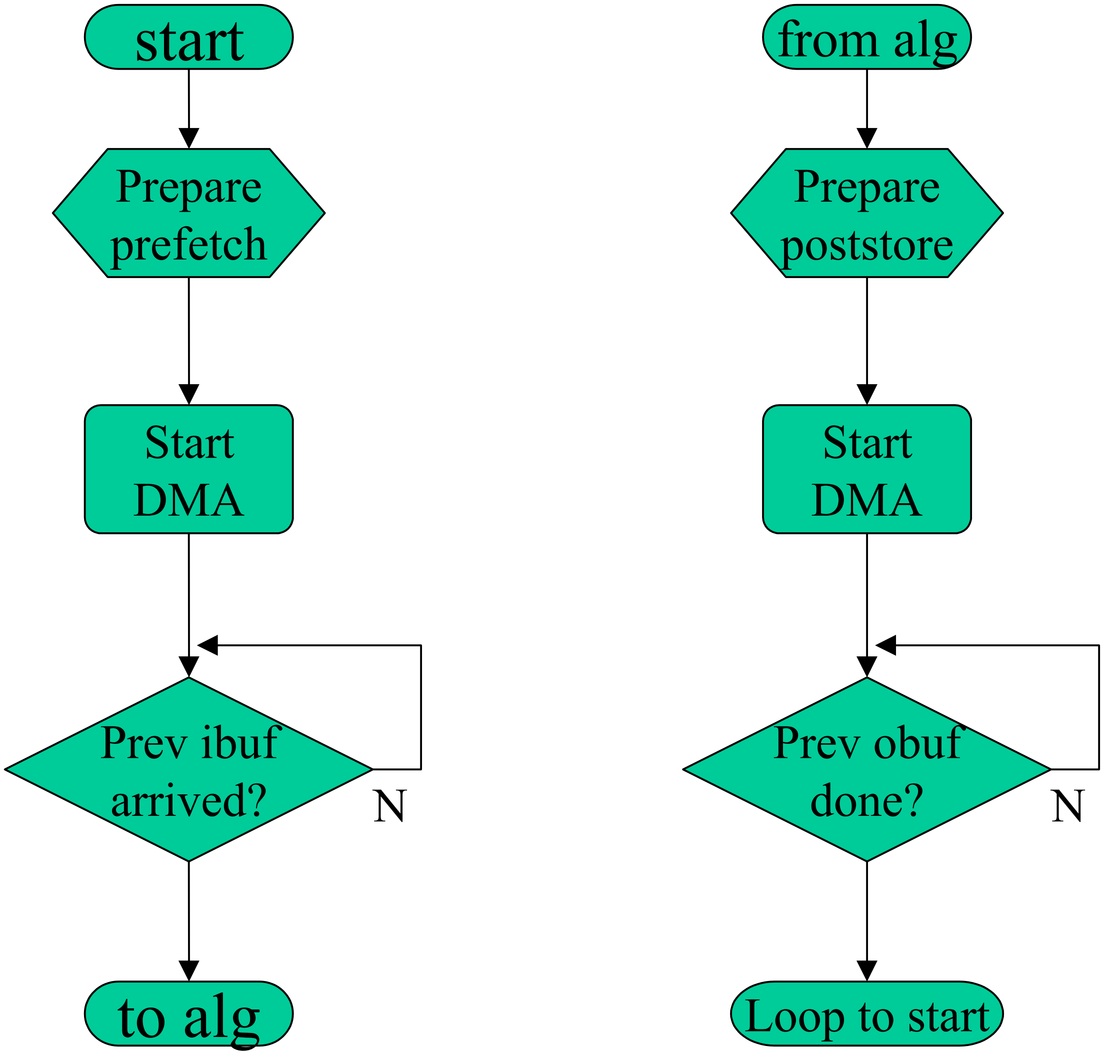
algorithm

Inner loop performs algorithm on a buffer producing an output buffer

Outer loop to post-store the output buffers using DMA and double buffering

Cleanup and exit

Data Transfer Loop



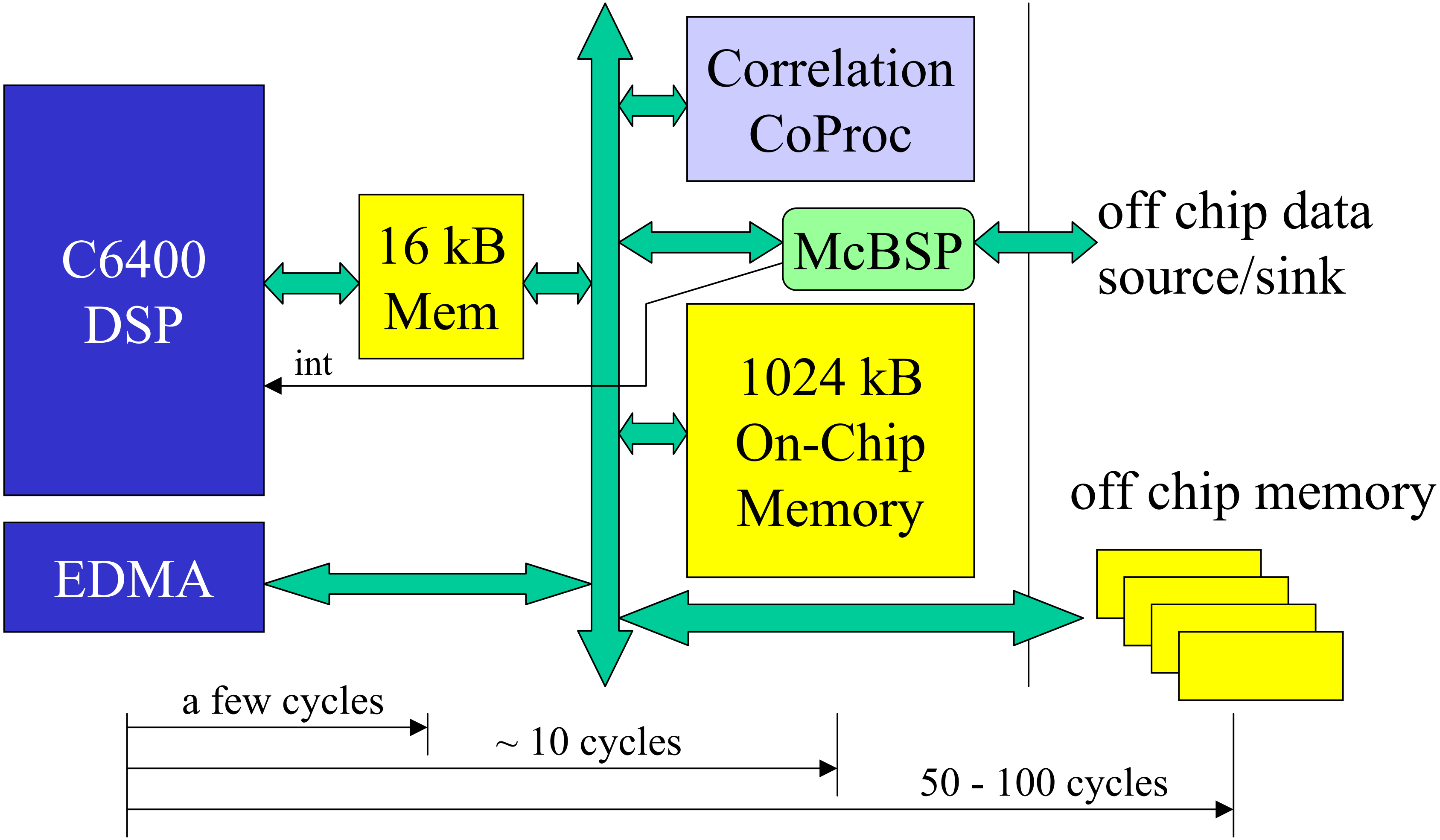
Prepare and Start DMA

- To prepare and start a DMA transfer, need to:
  - write a channel program and store it in the DMA controller
  - signal an event to trigger that channel program
- This can be slow
  - Might write and store once then reuse
  - Might have a faster means for Quick Transfers

Quick DMA

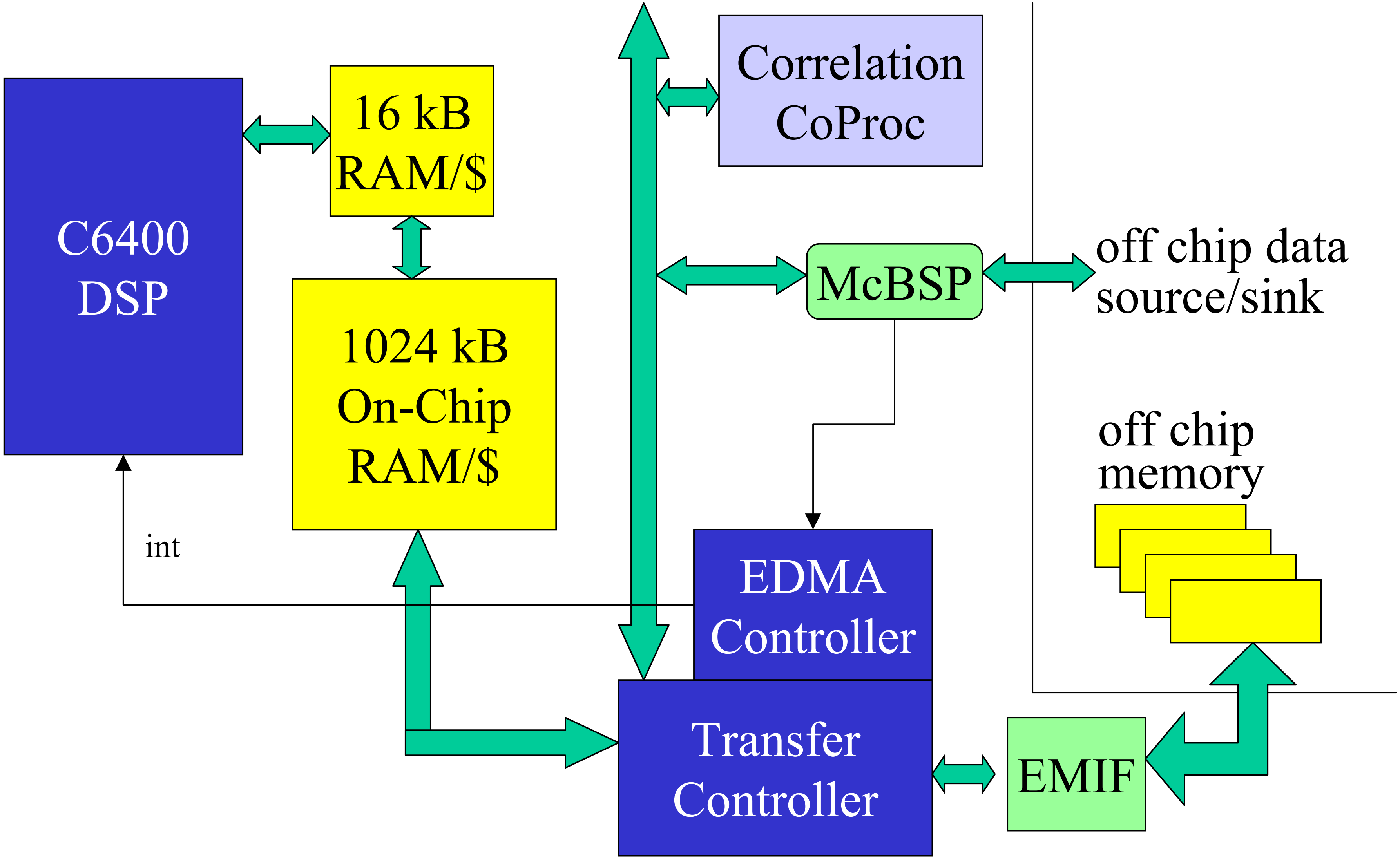
- C6x1x processors have QDMA
- QDMA are a set of control registers that initiate a TC transfer immediately.
- Need only supply:
  - source address
  - destination address
  - a transfer length
  - and control bits

C6400 System



Memory Latency

C6400 System — really



EDMA Highlights

- 16 Channels
  - Each channel may chain multiple transfers directed by parameter sets
  - 60+ additional parameter sets for reload and linked transfers
  - Performs all transfer types done by C6x0x DMA
  - Programmed via a dedicated parameter RAM (PaRAM)
- Highly Efficient Transfer Controller (TC)
  - Crossbar architecture processes multiple transfers concurrently
  - Highly-efficient and fully-pipelined cycle-by-cycle prioritization for low channel turn around
- TC services all DSP and cache memory requests in addition to EDMA channel program transfers

EDMA Controller Features

- High Performance: single cycle throughput
- 2KB Parameter RAM stores up to 85 transfer entries
- 16 channels programmable for
  - Element size (byte, half-word, word)
  - Src/Dst Addressing Modes
  - Transfer Type (2D or non-2D)
  - Priority of the transfer
  - Linked transfers
  - Chaining channels with one event
- Up to 16 Sync events (from external device or peripheral)
- Generates a CPU interrupt upon transfer completion
- Emulation and Endian Support

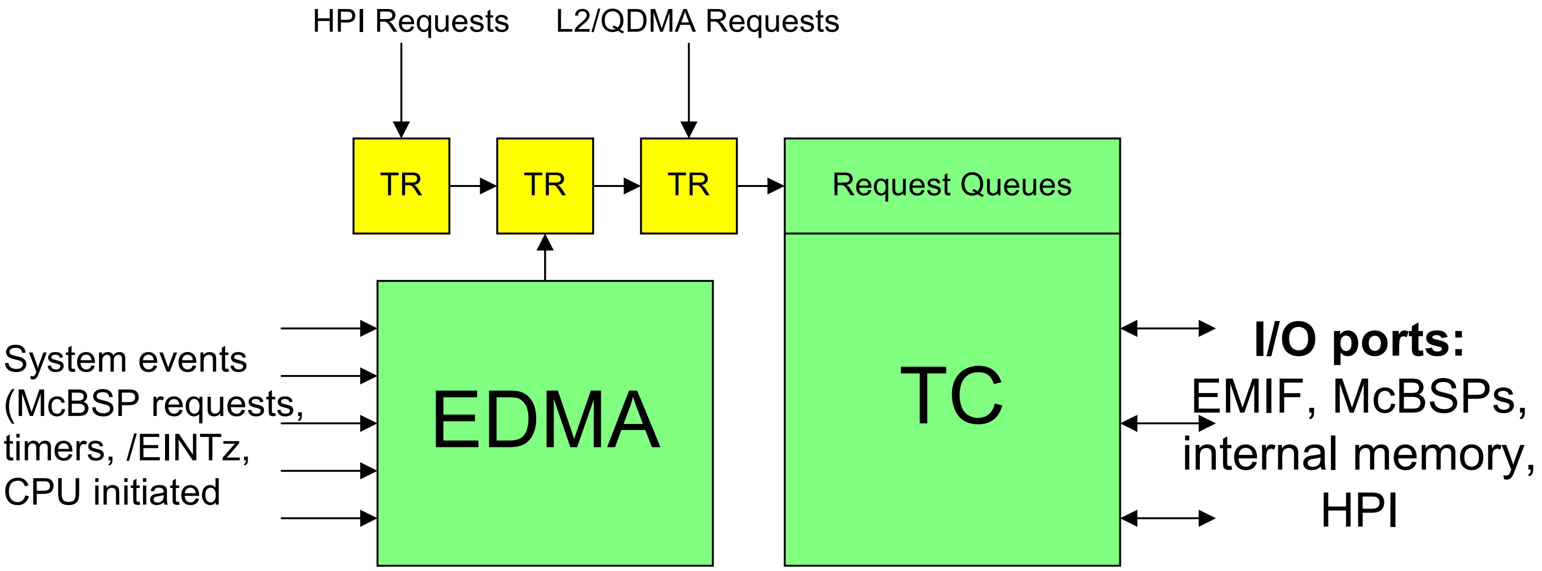
EDMA Controller Concepts

- Terms
  - Element: 8-, 16-, 32-bit
  - Frame: Group of elements
  - Array: Group of contiguous elements
  - Block: Group of Frames or Arrays
  - 2-D Transfer: Block transfer of Arrays
  - Non-2D Transfer: Block Transfer of Frames
- 2KB Parameter RAM
  - Stores transfer parameters for 16 channels
  - Stores reload parameters for up to 69 entries
  - Each entry comprises 6 words; always align on 24-byte boundary

Programmable Addressing

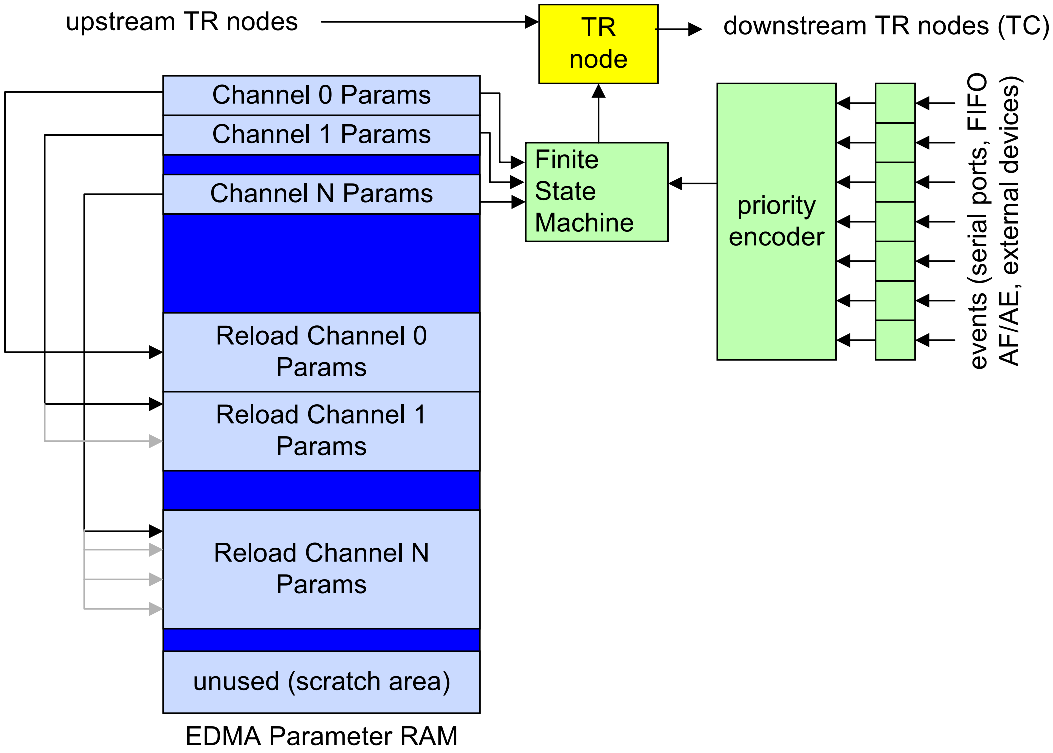
- Src and /or Dst Address can:
  - Remain Static
  - Increment
  - Decrement
  - Modified by signed index values
  - Replaced with Link parameters
- Independently programmable for Source and Destination
- Indexing allows different strides between elements and between frames
- Allows:
  - 2D Block Transfers on a single event
  - Circular Buffering via Linked List
  - Data Sorting/Interleaving

EDMA Architecture



- EDMA architecturally has three sections
  - EDMA controller and parameter RAM
  - Transfer Crossbar (TC)
  - Transfer Request (TR) nodes

EDMA Controller and PaRAM



- Captures events from all system DMA requesters
- Simultaneous events serviced via priority encoder
- FSM reads parameter block from dedicated 2KB PaRAM
- Formatted to create a TRP (Transfer Request Packet), and sent to TC via TR node
- Parameter updates and linking, while TC performs I/O
- Essentially, multi-threaded, special-purpose processor

EDMA Transfer Parameters

Event *N* Parameters

31 16	15 0	
OPTIONS		Word 0
SRC ADDRESS		Word 1
ARRAY/FRAME COUNT	ELEMENT COUNT	Word 2
DST ADDRESS		Word 3
ARRAY/FRAME INDEX	ELEMENT INDEX	Word 4
ELEMENT COUNT RELOAD	LINK ADDRESS	Word 5

Options Field

31 29	28 27	26 25	24 23	22	21	20	19 16	15 2	1	0
PRI	ESIZE	SUM	DUM	2DS	2DD	TCINT	TCC	Resv	LINK	FS

Address	Event Parameters	
0x01A00000	Event 0, Options	
0x01A00004	Event 0, SRC Address	
0x01A00008	Event 0, Array/Frame Count	Event 0, Element Count
0x01A0000C	Event 0, DST Address	
0x01A00010	Event 0, Array/Frame Index	Event 0, Element Index
0x01A00014	Event 0, Element Count Reload	Event 0, Link Address
0x01A00018 to 0x01A0002C	Parameters For Event 1	
…	…	
0x01A00168 to 0x01A0017C	Parameters For Event 15	
Address	**Reload/Link Parameters**	
0x01A00180	Event N, Options	
0x01A00184	Event N, SRC Address	
0x01A00188	Event N, Array/Frame Count	Event N, Element Count
0x01A0018C	Event N, DST Address	
0x01A00190	Event N, Array/Frame Index	Event N, Element Index
0x01A00194	Event N, Element Count Reload	Event N, Link Address
…	…	
0x01A007E0 to 0x01A007F7	Reload Parameters for Event Z	
	Unused RAM	
0x01A007F8 - 0x01A007FF	Scratch Pad Area	

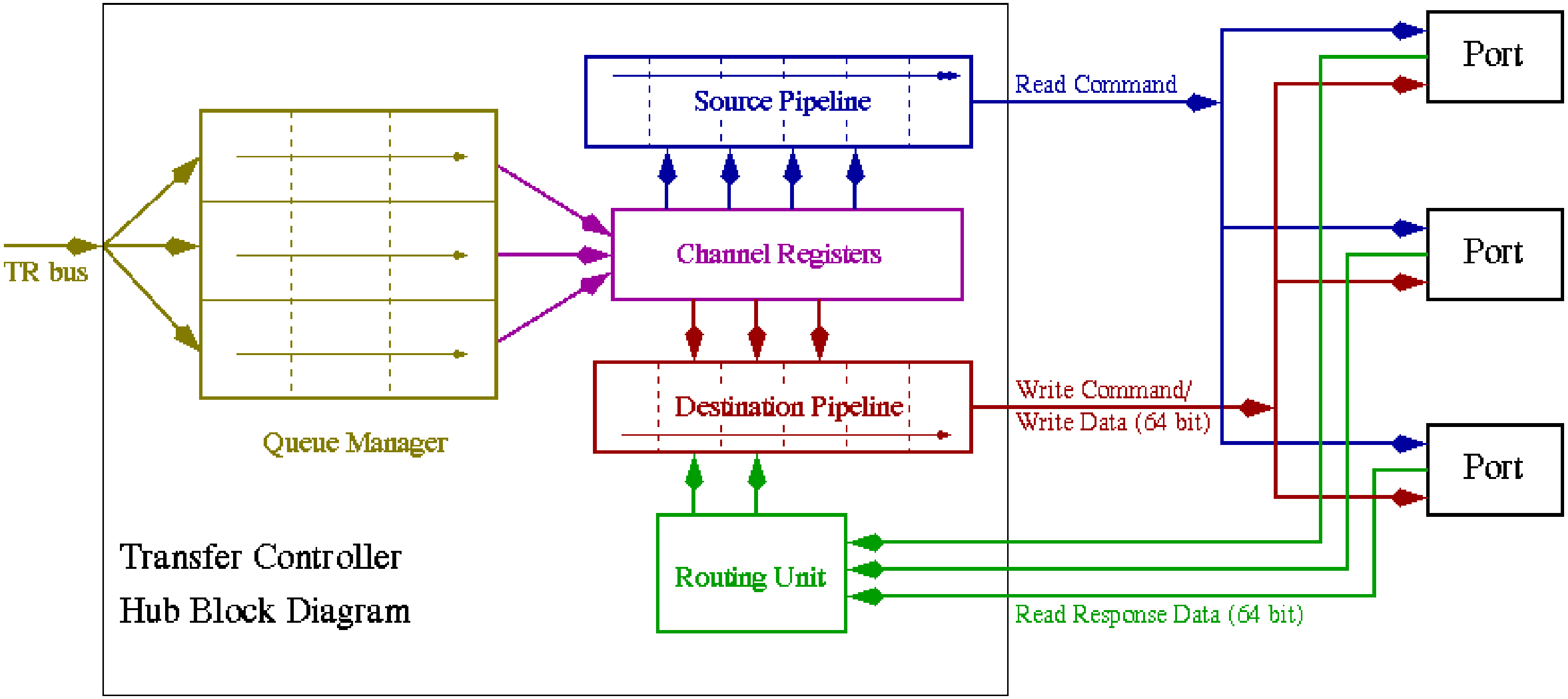
EDMA Transfer Priority

- Channels have no priorities; Instead their transfer parameters have Programmable Priority
- 3 Priority Levels available:
  - Level 0 / Urgent: NOT valid for EDMA Transfers
  - Level 1 / High: Used by EDMA/HPI Transfers
  - Level 2 / Low: Used by EDMA Transfers
- Level 1 and 2 Priorities are independently programmable for 16 channels when competing for:
  - EMIF
  - Peripherals
  - L2 SRAM
- Priority Queue Status Register indicates if a priority queue is empty

EDMA: Interrupt Generation

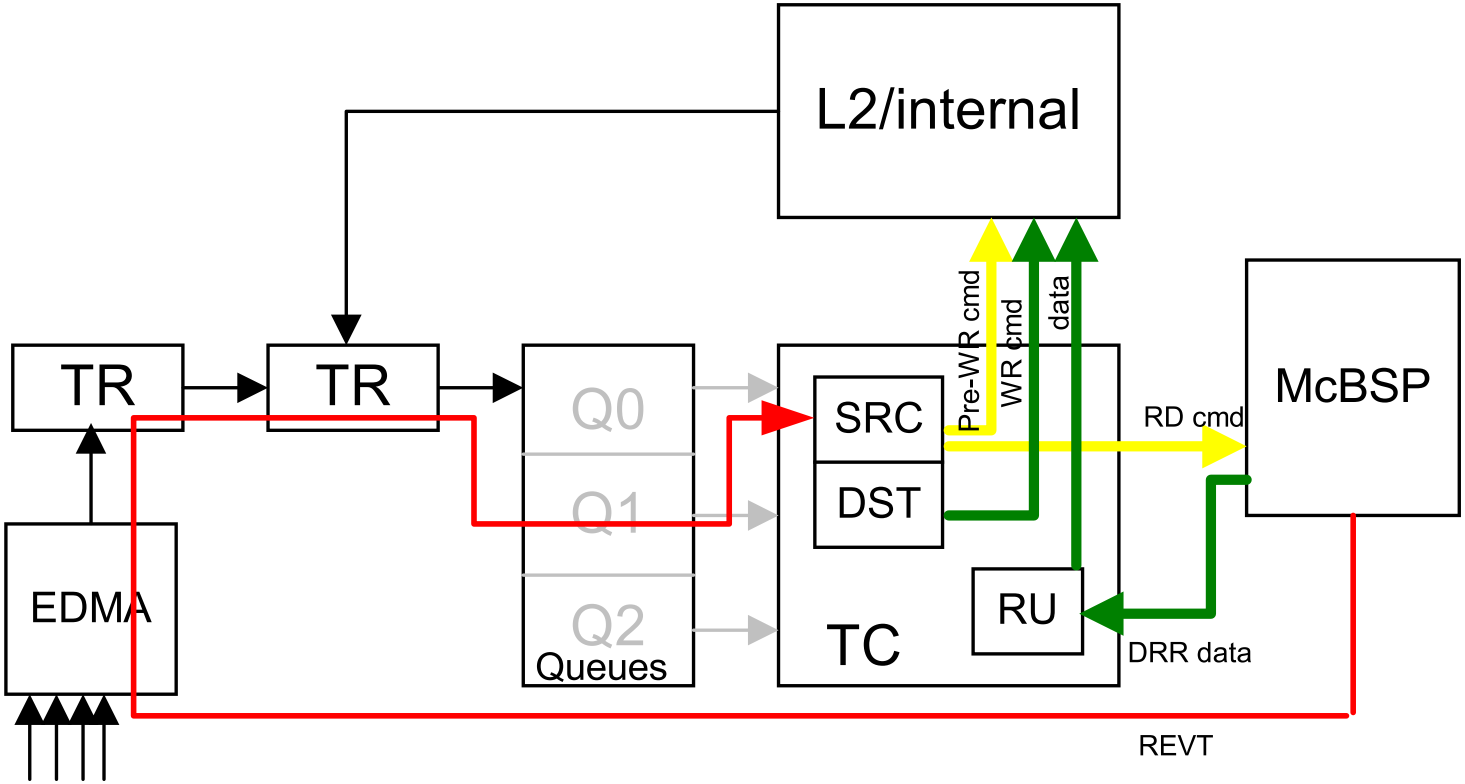
- Generates a single CPU interrupt (EDMA_INT) for all 16 channels
- Transfer Completion Code (TCC:0-15) specified for each channel sets the relevant Channel Interrupt Pending bit in CIPR (assuming that the relevant CIER bit is set)
  - Multiple channels can have same TCC - same ISR for different events
- Channel Complete Interrupt is generated when the channel executes the transfer to completion -- not when the transfer request is submitted

TC Architecture

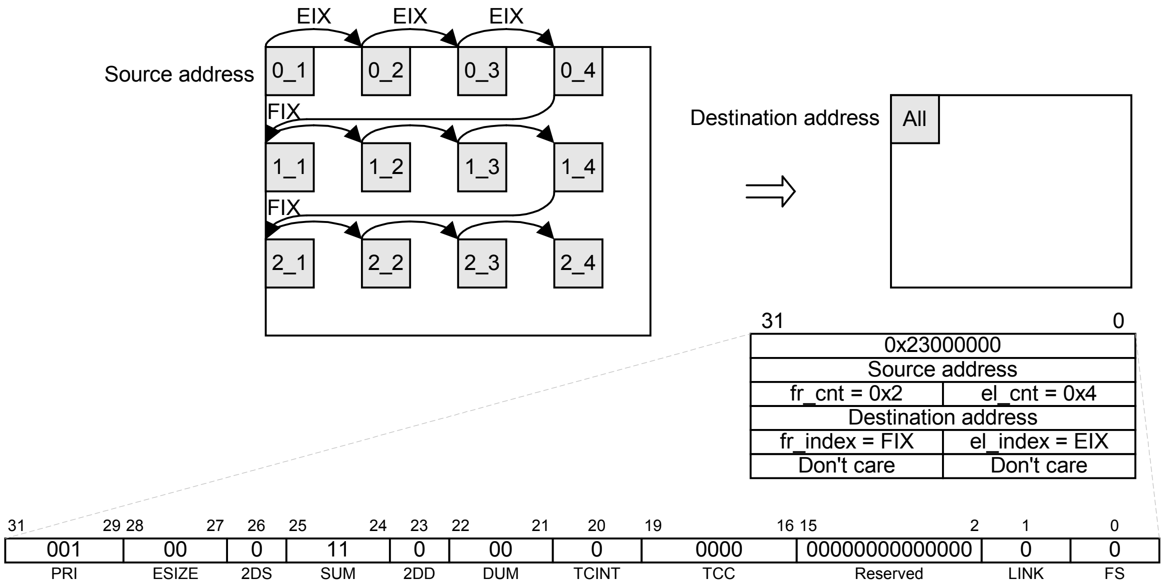


- TR Packets are placed into one of three queues (0 = highest, 2 = lowest)
- Transfers are performed in order within each queue
- TC pipeline processes each set of TR parameters to perform accesses
- All three queues can be active simultaneously

Typical EDMA Flow



EDMA Example #1

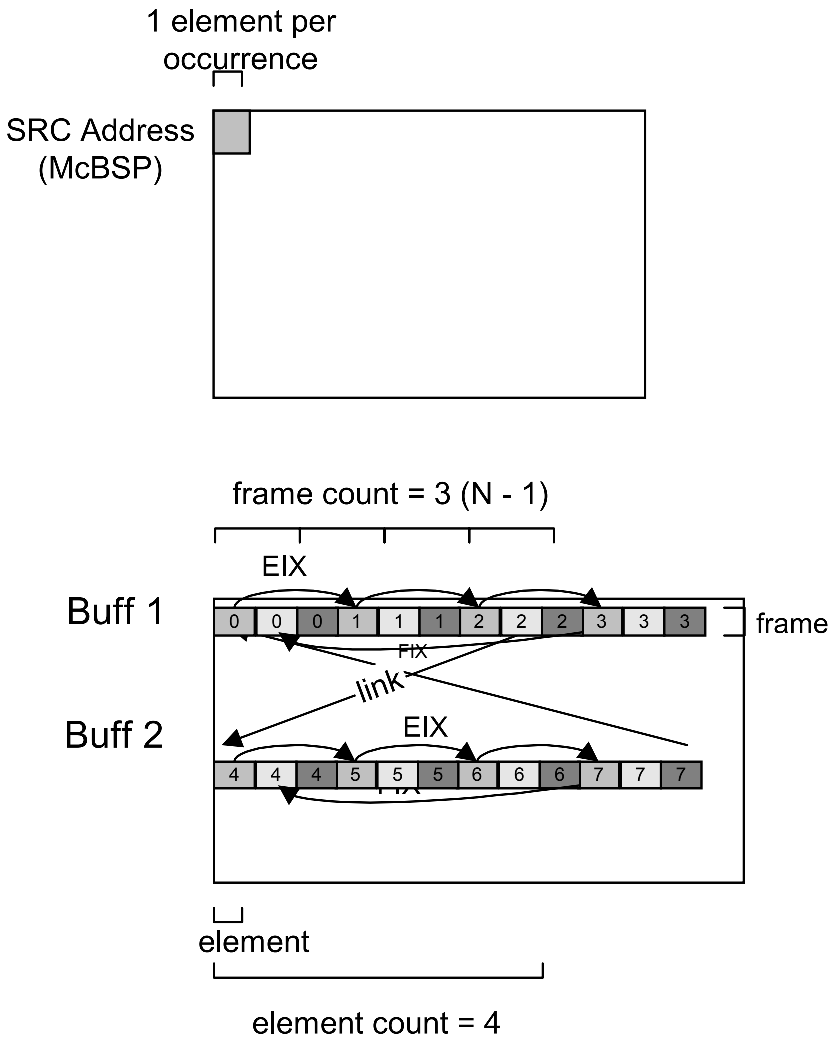


31	0
0x23000000	
Source address	
fr_cnt = 0x2	el_cnt = 0x4
Destination address	
fr_index = FIX	el_index = EIX
Don't care	Don't care

31 29	28 27	26	25 24	23	22 21	20	19 16	15 2	1	0
001	00	0	11	0	00	0	0000	00000000000000	0	0
PRI	ESIZE	2DS	SUM	2DD	DUM	TCINT	TCC	Reserved	LINK	FS

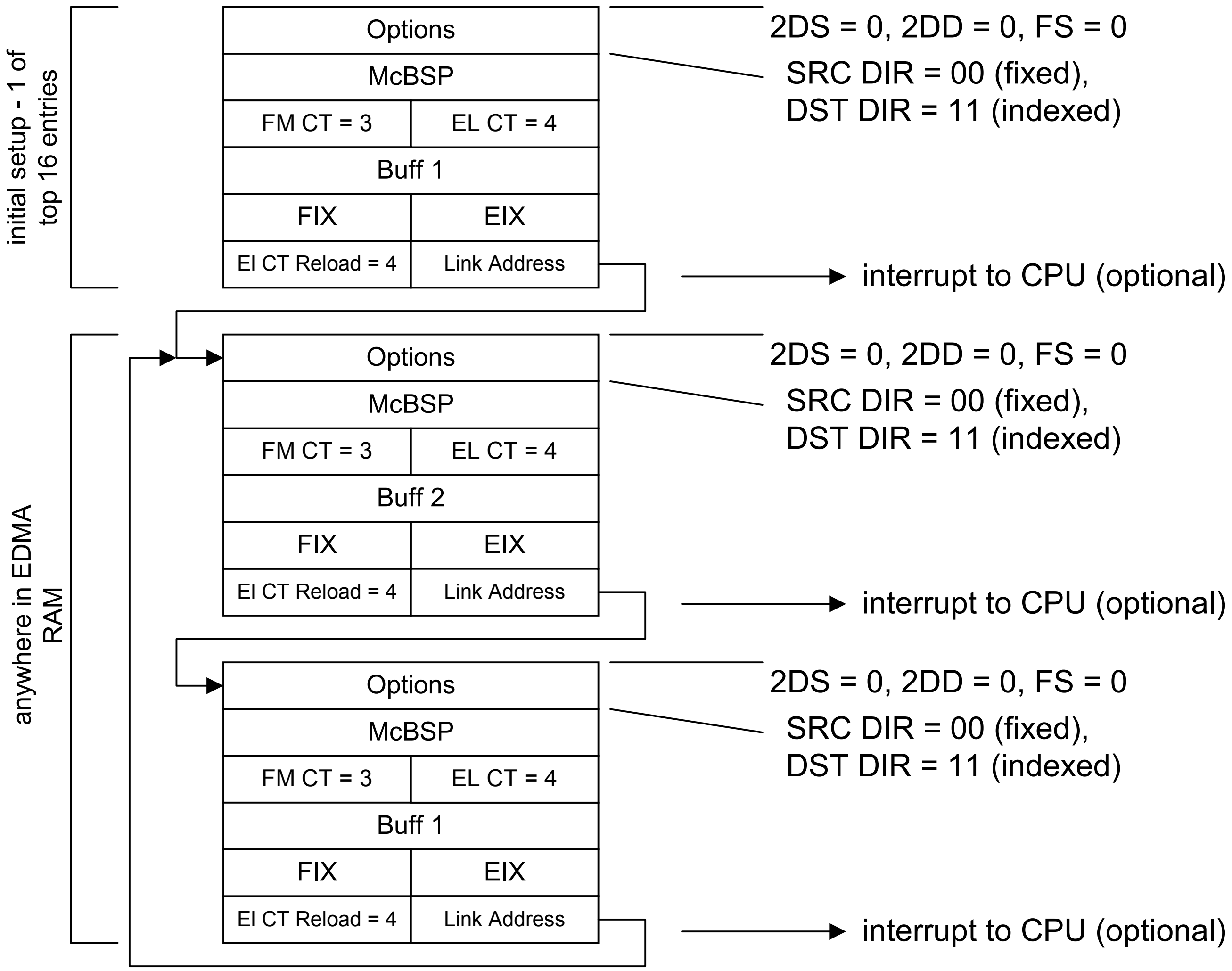
- Source elements spaced by EIX and frames by FIX
- All destination transfers go to a single address
- EDMA supports linear, fixed, decrement, and indexed addressing modes

EDMA Example #2 - Linking



- Double buffering can be easily set up using linking feature of EDMA
- Each buffer can create an interrupt to CPU to inform it data is ready (alternatively an EDMA transfer can set a S/W-visible flag)
- Increased numbers of buffers can easily be added through EDMA parameters
- Many exotic combinations are possible using EDMA options, linking and reload

EDMA Example #2 - Linking



Options	
McBSP	
FM CT = 3	EL CT = 4
Buff 1	
FIX	EIX
El CT Reload = 4	Link Address

2DS = 0, 2DD = 0, FS = 0

SRC DIR = 00 (fixed),
DST DIR = 11 (indexed)

interrupt to CPU (optional)

Options	
McBSP	
FM CT = 3	EL CT = 4
Buff 2	
FIX	EIX
El CT Reload = 4	Link Address

2DS = 0, 2DD = 0, FS = 0

SRC DIR = 00 (fixed),
DST DIR = 11 (indexed)

interrupt to CPU (optional)

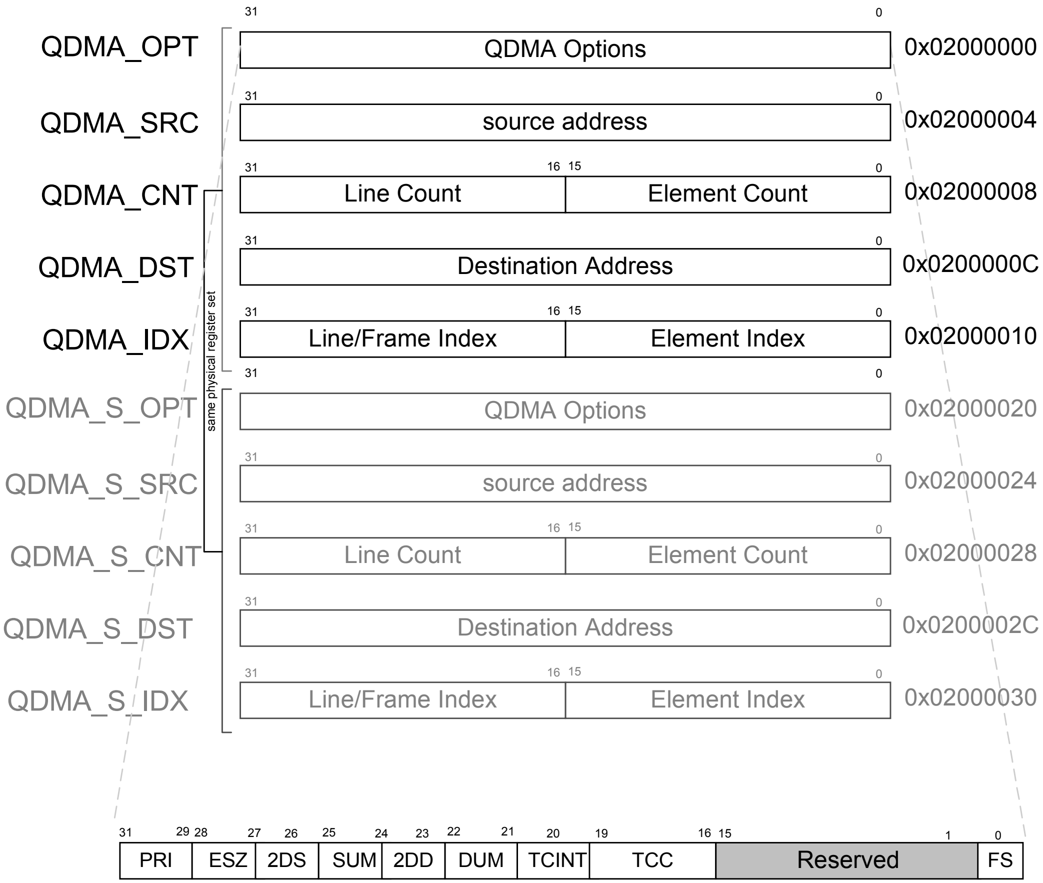
Options	
McBSP	
FM CT = 3	EL CT = 4
Buff 1	
FIX	EIX
El CT Reload = 4	Link Address

2DS = 0, 2DD = 0, FS = 0

SRC DIR = 00 (fixed),
DST DIR = 11 (indexed)

interrupt to CPU (optional)

Quick DMA (QDMA)



- QDMAs can be submitted by CPU as a “fire-n-forget” type block transfer
- QDMA registers are accessible in a single cycle
- Pseudo-mapping of submit registers allow back-to-back similar transfers to be submitted in a single cycle
- QDMAs support all addressing modes as the EDMA
- QDMAs do not support linking, but may be “chained” to EDMA events

QDMA Programming Model

- Initial requests
  - Perform 4 writes to QDMA registers to set up parameters
  - Perform fifth write to QDMA pseudo register to set up fifth parameter and automatically submit transfer request
- Subsequent requests
  - Write only changing parameters to QDMA pseudo registers to update parameters and submit request
- Interrupts and chaining are supported exactly as with the EDMA

Conclusion

- Two ways that DMA is important in DSP systems:
  - Lower interrupt rate and therefore lower overhead from interrupts
  - Scheduling data into low-latency memories
- These have convinced DSP vendors to provide sophisticated DMA controllers in our on-chip systems.