DMA Controller
(8237 Programming Examples)

Dr A Sahu

Dept of Comp Sc & Engg.

IIT Guwahati

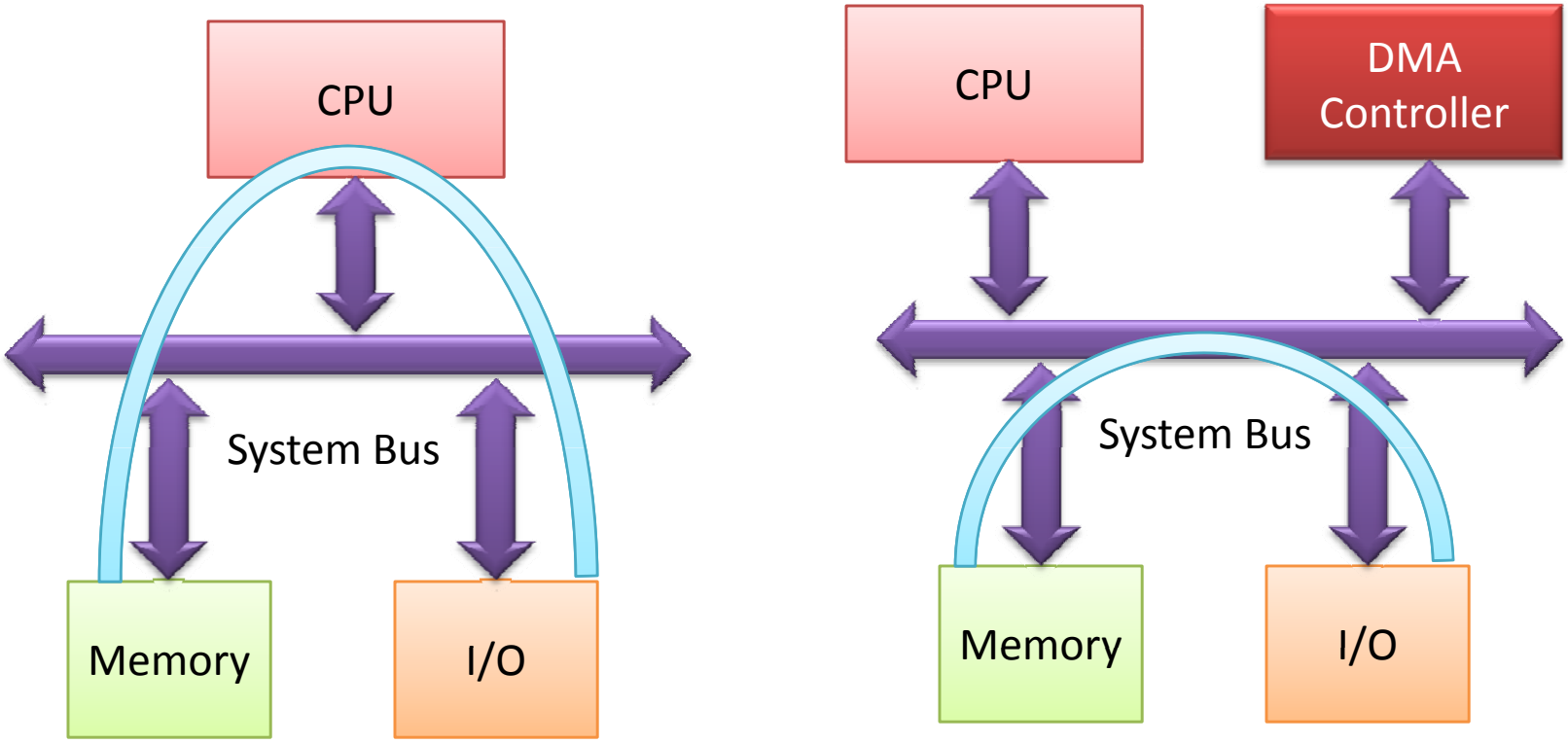
Outline

- DMA controller
- DMA Architecture
  - Registers
- Introduction to Programming DMA
  - How to configure
- Enhancing performance by DMA
  - Processor Vs Memory Speed Gap
  - In GPU (Nvidia)
  - In MP ASIPs (Cradle)

DMA

- Direct Memory Access
- DMA Controller
- DMA mode of I/O
- Programmed mode I/O vs DMA mode I/O

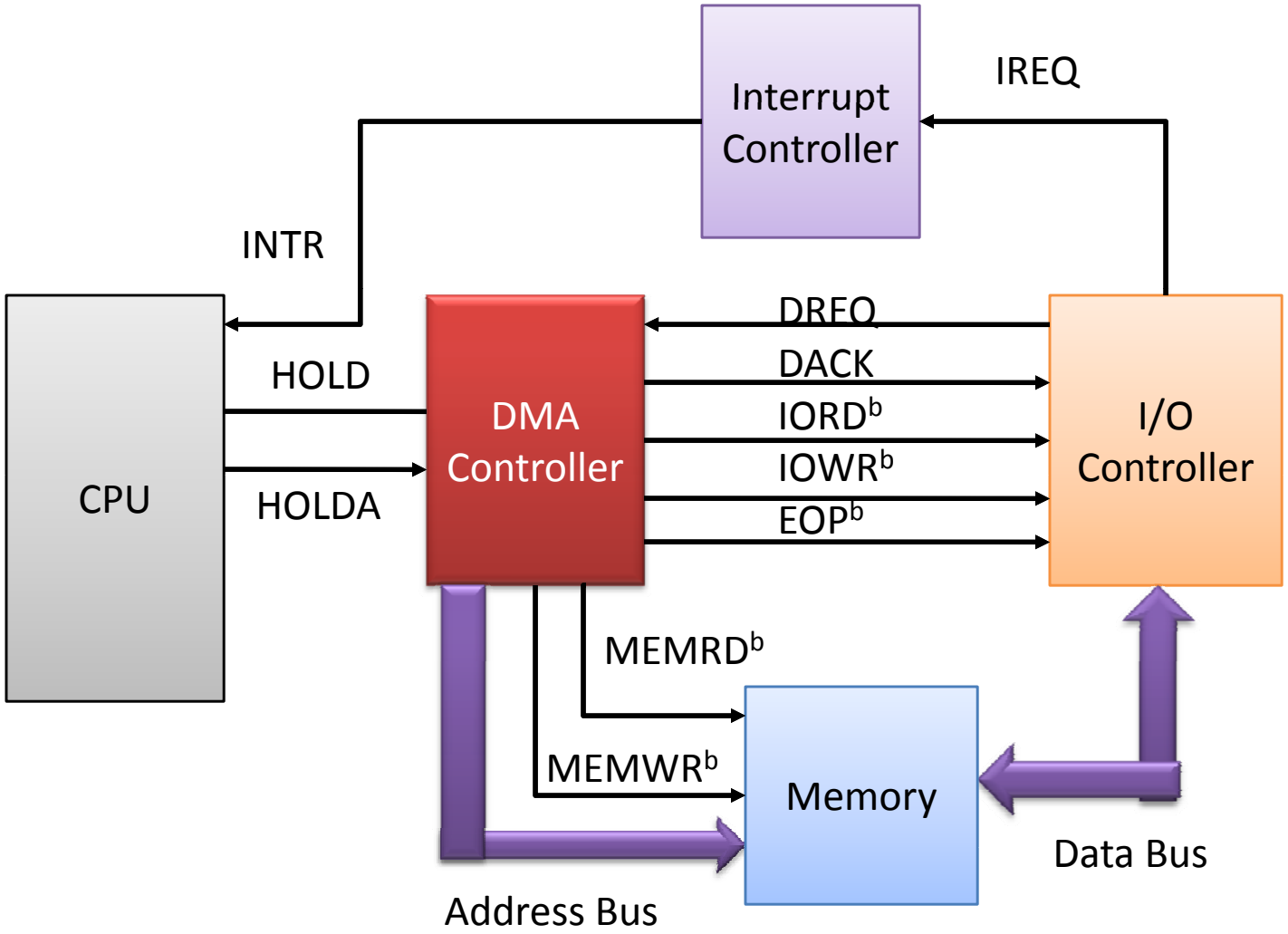
Data Transfer DMA mode



Data Transfer : DMA

- Problems with programmed I/O
  - Processor wastes time polling
  - Lets take example of Key board
    - Waiting for a key to be pressed,
    - Waiting for it to be released
    - May not satisfy timing constraints associated with some devices : **Disk read or write**
- DMA
  - Frees the processor of the data transfer responsibility

DMA Controller



DMA Controller

- DMA is implemented using a DMA controller
- DMA controller
  - Acts as slave to processor
  - Receives instructions from processor
  - Example: Reading from an I/O device
    - Processor gives details to the DMA controller
    - I/O device number
    - Main memory buffer address
    - Number of bytes to transfer
    - Direction of transfer (memory $\rightarrow$ I/O device, or vice versa)

DMA: HOLD and HOLDA

- HOLD: DMA to CPU
  - DMA Send HOLD High to CPU
  - I (DMA) want BUS Cycles
- HOLDA
  - CPU send HOLDA
  - BUS is granted to DMA to do the transfer
  - DMA is from Slaves to Master mode
- HOLD Low to CPU
  - I (DMA) finished the transfer
- Cycle Stealing if One BUS
- Other wise Separate process independent of processing

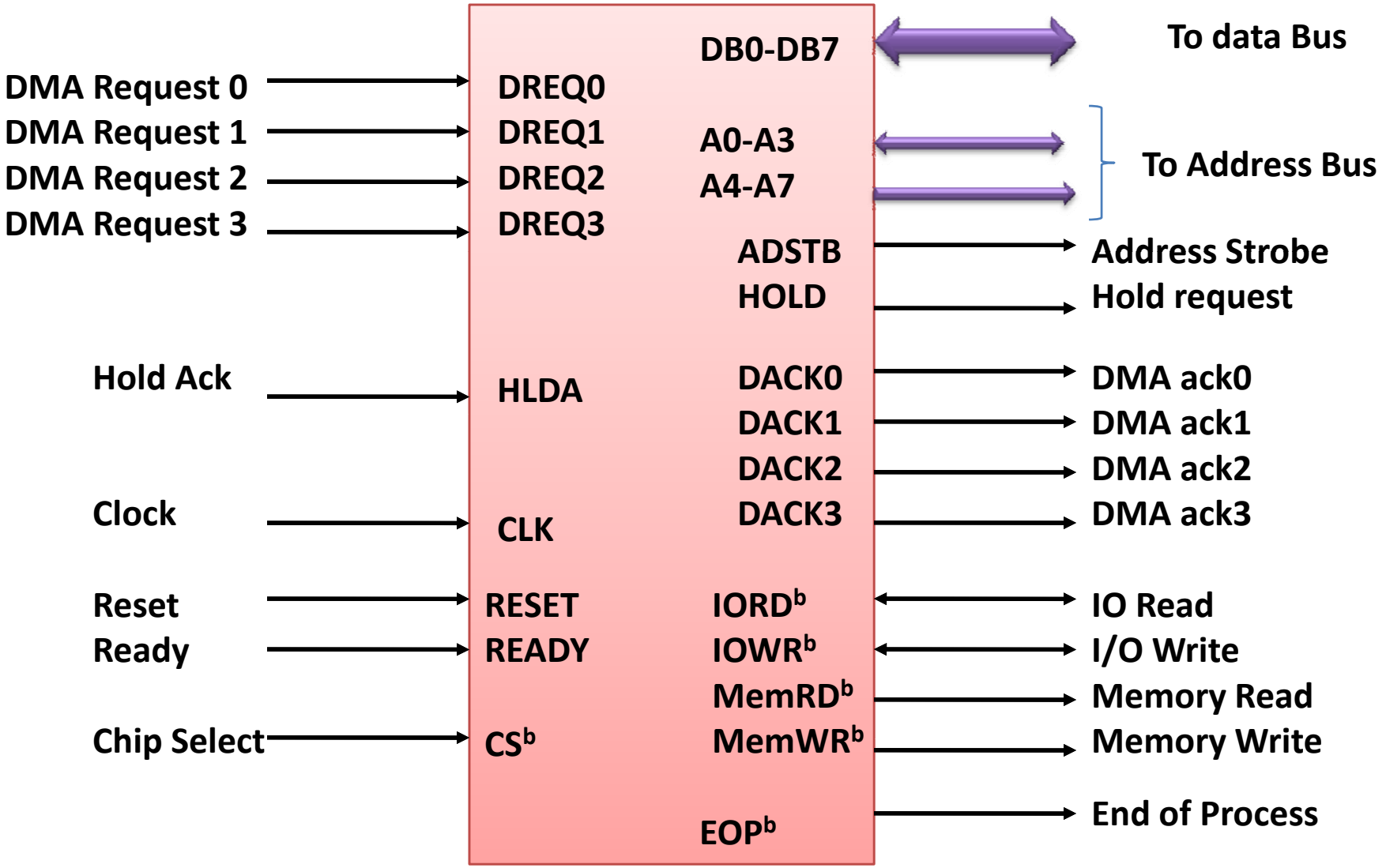
Steps in a DMA operation

- Processor initiates the DMA controller
  - Gives device number, memory buffer pointer, … Called ***channel initialization***
  - Once initialized, it is ready for data transfer
- When ready, I/O device informs the DMA controller
  - DMA controller starts the data transfer process
    - » Obtains bus by going through bus arbitration
    - » Places memory address and appropriate control signals
    - » Completes transfer and releases the bus
    - » Updates memory address and count value
    - » If more to read, loops back to repeat the process
- Notify the processor when done
  - Typically uses an interrupt

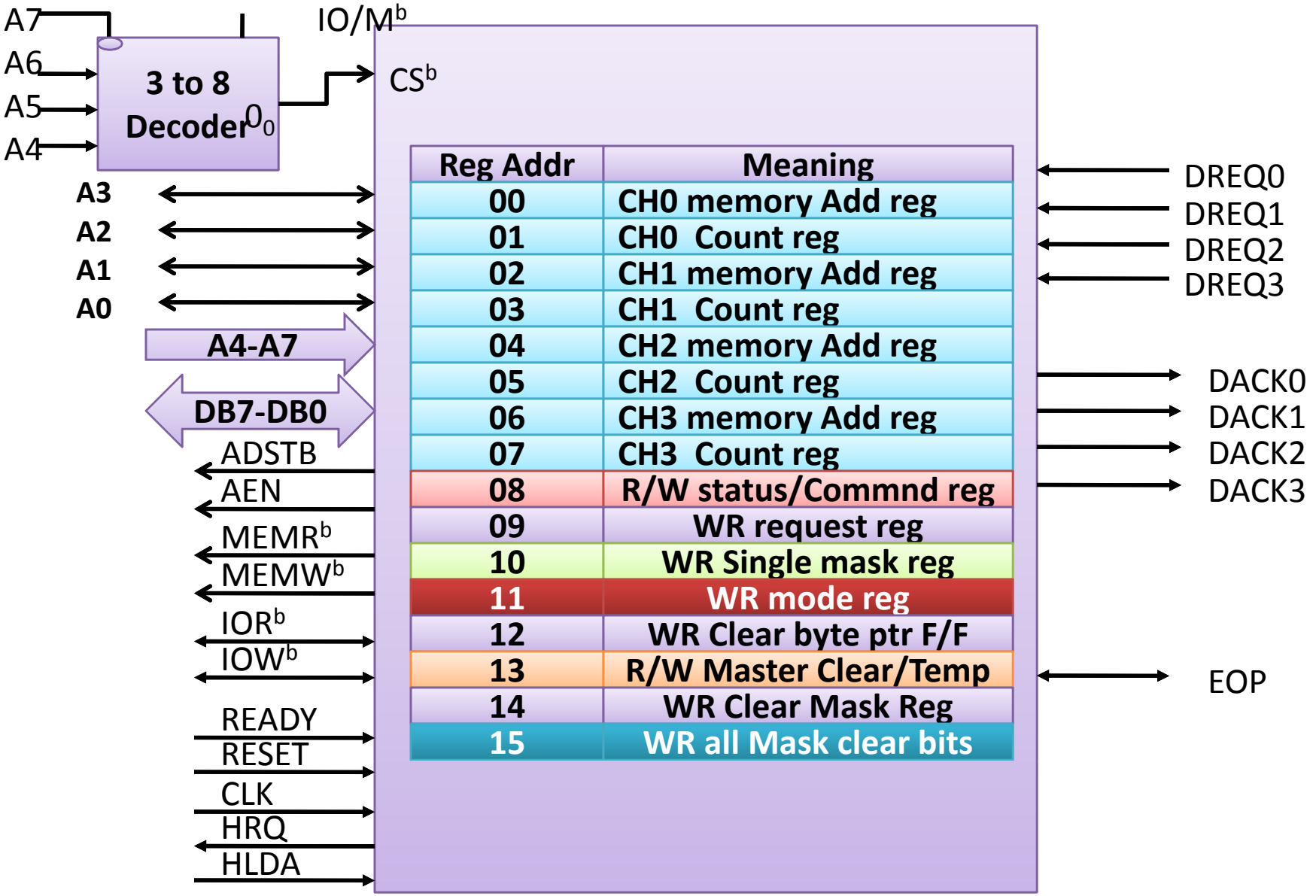
8237 DMA Controller

- Enable/Disable control of Individual DMA Req
- Four Independent DMA Channel
- Independent Auto initialization for all channel
- Memory to Memory transfer
- Memory Block initialization
- Address Increment and Decrement
- Cascade (Directly expandable to any #CHN)
- End of Process input for terminating transfers
- Software DMA requests

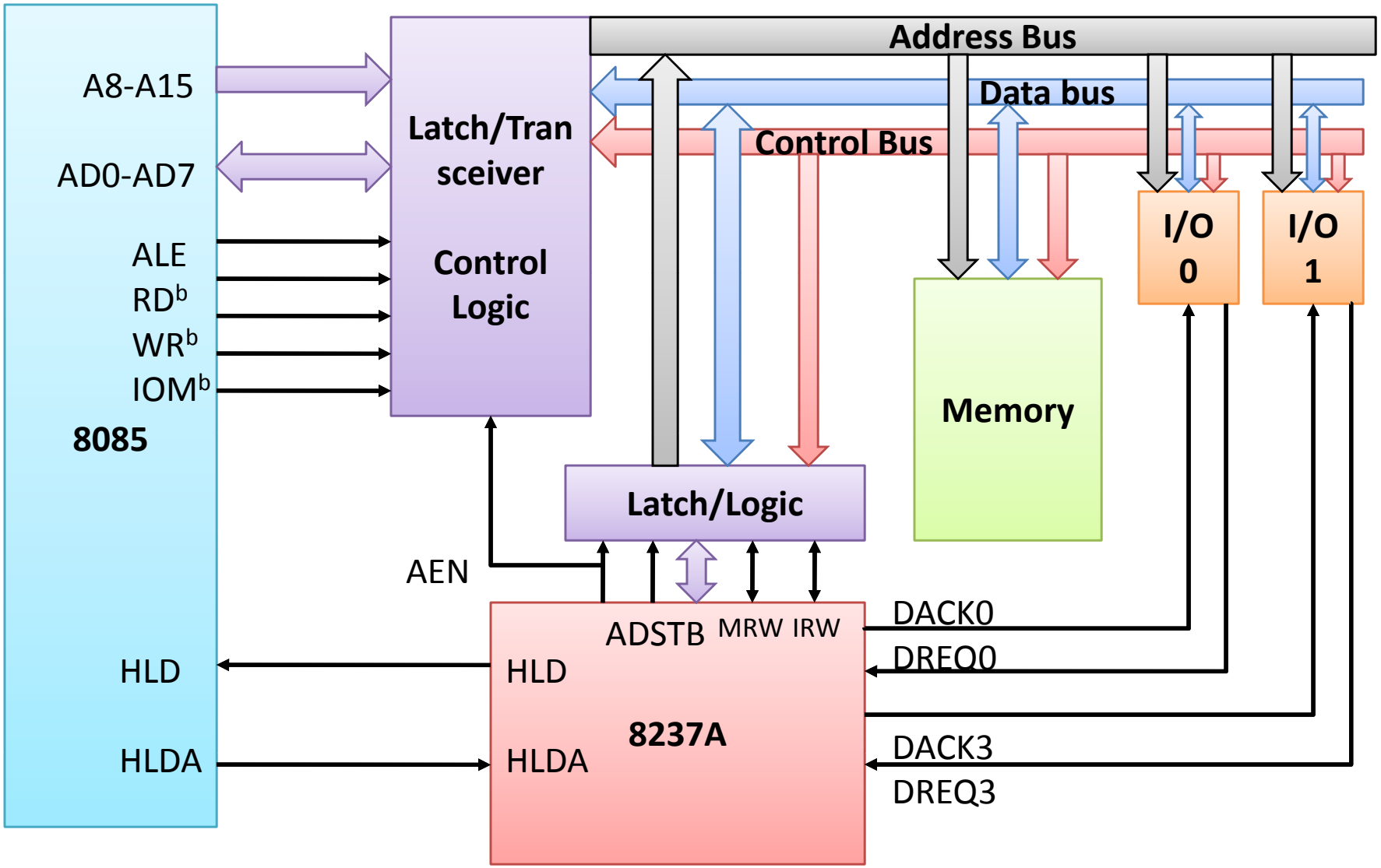
8237 DMA Controller



DMA Controller with Internal Registers



DMA interface to I/O Device



8237 DMA channels

- 8237 supports four DMA channels
- Handshake of I/O
  - DREQ3-DREQ0 (DMA Request)
  - DACK3-DACK0 (DMA Acknowledge)
- AEN & ADSTB : Address Enable & Add Strobe
- A3-A0 and A4-A7: Address line
- HRQ and HLDA: Hold Request and Hold Ack
  - Request hold BUS

Registers of 8237A

- Command registers

D7	D6	D5	D4	D3	D2	D1	D0
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
DACK Sense Active Low/High	DREQ Sense Active Low/High	Late/Extended Write Selection	Fixed/ Rotating Priority	Normal/ Compressed Timing	Enable/Disable Controller	Dis/En CH0 Address Hold	Dis/En Mem-mem Transfer

- Mode register

D7	D6	D5	D4	D3	D2	D1	D0
		0/1	0/1				
00=Demand Mode 01=Single Mode 10=Block Mode 11=Cascade mode		INC/DEC Address	Dis/Ena Auto Initialisation	00=verify Transfer 01=Write Transfer 10=Read transfer 11=Ilega		Channel Select 00=CH0, 01=CH1 10=CH2, 11=CH3	

DMA Registers

- Request register

D7	D6	D5	D4	D3	D2	D1	D0
XXXXX Don't Care					Reset/Set REQ bit	00-11 Channel Select 00=CH0, 01=CH1, 10=CH2, 11 CH3	

- Mask register
  - Used to disable a specific channel

D7	D6	D5	D4	D3	D2	D1	D0
XXXX Don't Care				0/1	0/1	0/1	0/1
				Clear/Set CH3 Mask Bit	Clear/Set CH2 Mask Bit	Clear/Set CH2 Mask Bit	Clear/Set CH0 Mask Bit

- Status register

D7	D6	D5	D4	D3	D2	D1	D0
0/1, CH3	0/1,CH2	0/1, CH1	0/1, CH0	0/1, CH3	0/1, CH2	0/1, CH1	0/1, CH0
CH# Request				CH# Has reached TC			

Registers

- Temporary register
  - Used for memory-to-memory transfers
- Current address register
  - One 16-bit register for each channel
  - Holds address for the current DMA transfer
- Current word register
  - Keeps the byte count
  - Generates terminal count (TC) signal when the count goes from zero to FFFFH

Type of Data Transfer using 8237 DMA

- Single transfer
  - Useful for slow devices, word count upto TC
  - Each time increment & Decrement occurs
- Block transfer mode
  - Transfers data until TC is generated or external EOP^b signal is received
- Demand transfer mode
  - Similar to the block transfer mode
  - In addition to TC and EOP, transfer can be terminated by deactivating DREQ signal
- Cascade mode
  - Useful to expand the number channels beyond four

Programming the 8237

- Write a control word in Mode registers
  - Select Channel, Type of Transfer (R,W, Verify)
  - DMA mode (block, single byte, demand mode)
- Write a control word in Command registers
  - Priority among channel
  - Enable the 8237, DREQ & DACK active level
- Write the staring address of the data block to be transferred in the Channel MAR
- Write the count in the Channel Count Reg

Transfer 1K of memory from to Floppy disk at CH3

- Disable DMA controller & begin initialization Instructions
- Initialize CH3
- Starting address of Memory Block (Say 4075H) and subsequent bytes are increasing address order
- Command parameters: Normal timing, Fixed priority, late write, DREQ&DACK both active low
- Set up the demand mode so that DMA can complete the transfer without any interruption

Program to transfer

```
MVI     A, 0000100B     ; (Disable DMA)
OUT     08H                     ;  Send to Command Reg
MVI     A,00000111V     ;00 Demand mode
                         0 increAddress, 0 Disable
                        autoload, 01 write, 11=CH3
OUT     0BH             ; Send to Mode Reg
MVI     A,75H           ; Higher Address to
OUT     06H             ; MAR CH3
MVI     A,04H           ; 7504 Address
OUT     06H
MVI     A,FFH           ; lower byte of Tc =03FF,  1K Byte
OUT     07H             ; CH3 count Register
MVI     A,03H           ; higher byte of 03FF
OUT     07H             ; CH3 Count Register
MVI     A, 1000000B     ; Command for DACK High
OUT     08H             ; Send to command Register
```

High Level: DMA data transfer Code

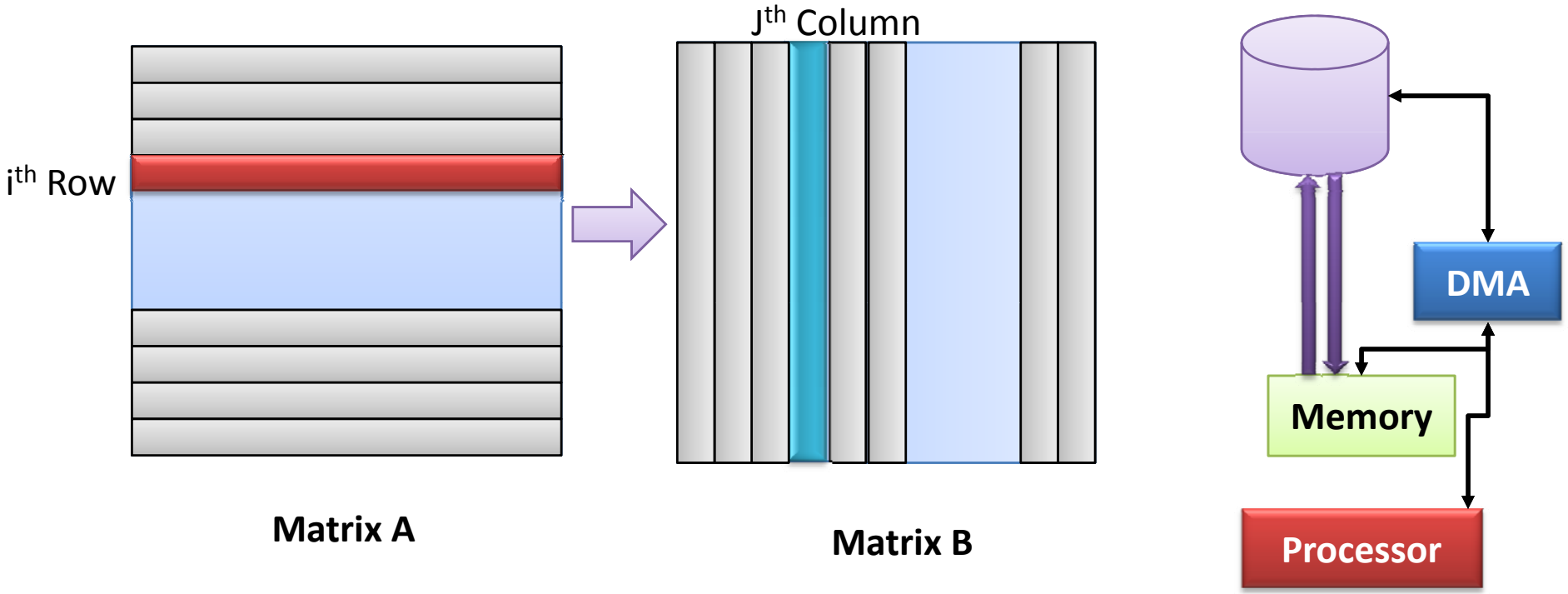
```
DMA_Struct {
      int   NB; // Number of Byte
      int   SA; // Source Address
      int   DA; // Destination address;
      int   CN;  // Channel Number
} DMA_Channel_Struct;
typedef DMA_Channel_Struct DCS;

DCS my_dataTransfer, *dcs_p;
dcs_p = & my_dataTransfer;
dcs_p->NB=1024; dcs_p->SA=0x15000;
dcs_p->DA =0x10000; dcs_p->CN=0;

DataMoveDo(Mydcs_ptr);

DataMovewait(Mydcs_ptr)
```

Some time Mem-Mem transfer use two channel
CH0 (SRC) & CH1 (DST)

Processor Vs Memory Speed Gap

- Processor speed is very high
- Memory speed is low
- Cache (Unpredictable)
- Scratch Pad (Predictable but user have to do)
- Shared address space
  - Both SP & DRAM have same memory address
  - May be even Disk (if SP/DRAM virtually addressable)

Matrix Multiplication: using DMA



Software Pipelining

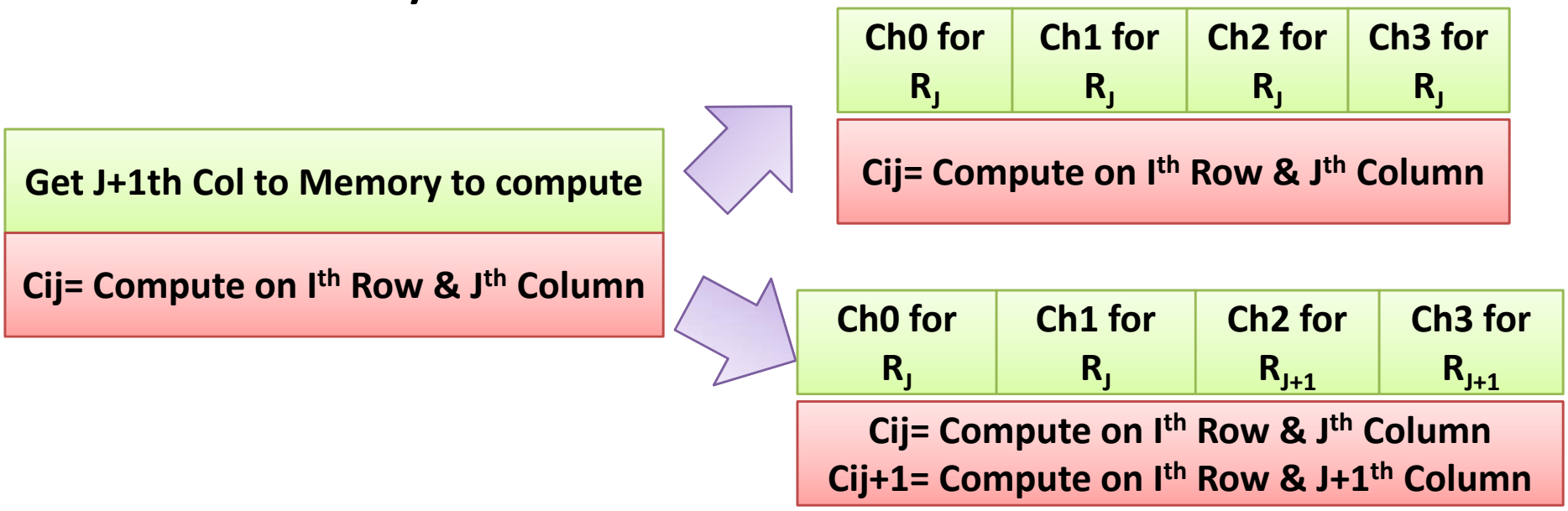
Get J+1th Col to Memory to compute

Cij= Compute on Ith Row & Jth Column

Use DMA to get Data to memory from Disk

Matrix Multiplication: using DMA

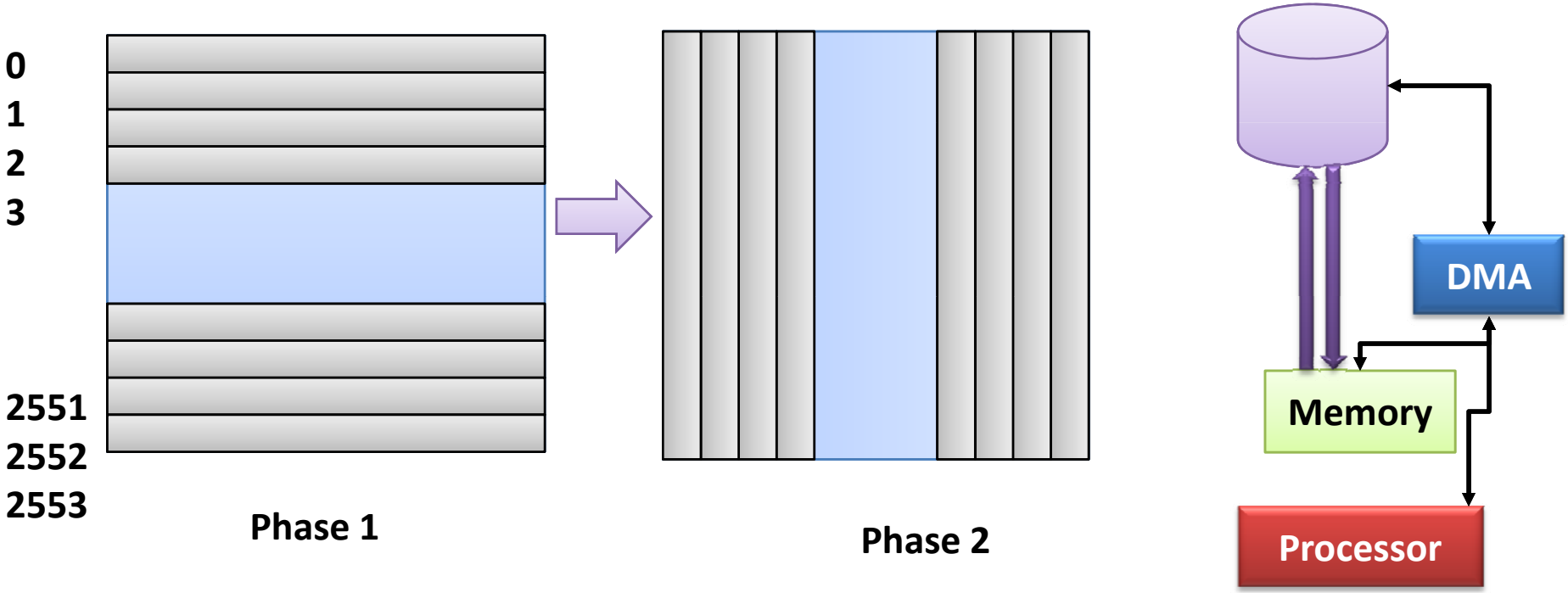
- Assume Compute for a C_{ij} takes= 2µS
- Transferring a row to/from memory = 8µS using a DMA channel
- You can use 4 DMA channel of 8237A effectively to utilize the transfer



DMA transfer for Matrix Multiplication

```
DCS DT0,DT1,DT2,DT3, *dcs_p0,*dcs_p1,*dcs_p2,*dcs_p3;
dcs_p0 = &DT0; dcs_p1 = &DT1; dcs_p2 = &DT2; dcs_p3 = &DT4;

dcs_p0->NB=4096; dcs_p->SA=SA0;dcs_p0->DA =DA0; dcs_p->CN=0;
dcs_p0->NB=4096; dcs_p->SA=SA1;dcs_p0->DA =DA1; dcs_p->CN=1;
dcs_p0->NB=4096; dcs_p->SA=SA2;dcs_p0->DA =DA2; dcs_p->CN=2;
dcs_p0->NB=4096; dcs_p->SA=SA3;dcs_p0->DA =DA3; dcs_p->CN=3;

DataMove(dcs_p0); DataMove(dcs_p1);
DataMove(dcs_p2); DataMove(dcs_p3);

S=0;
for(k=;k<R;k++){
    S=S+A[i][k]*B[i][k]; // Assume B in transpose format
}                        // So B[i][k] instead of B[k][i]
C[i][j]=S;

DataMovewait(dcs_p0); DataMovewait(dcs_p1);
DataMovewait(dcs_p2); DataMovewait(dcs_p3);
```

FFT Example using DMA



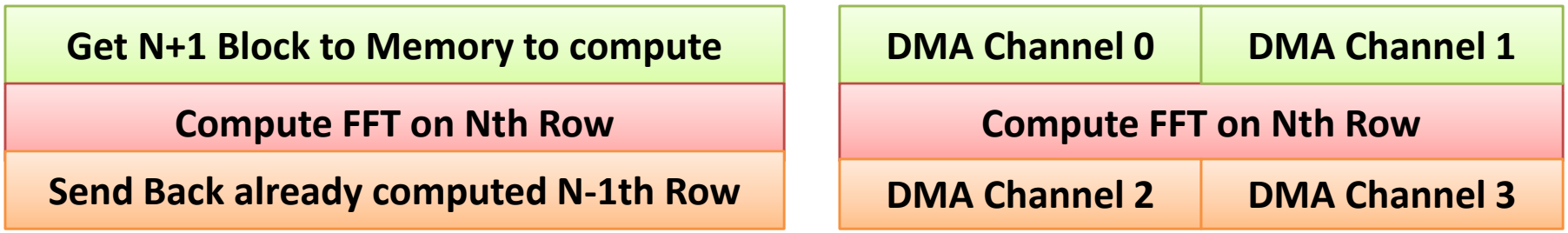
Software Pipelining

- Get N+1 Block to Memory to compute
- Compute FFT on Nth Row
- Send Back already computed N-1th Row

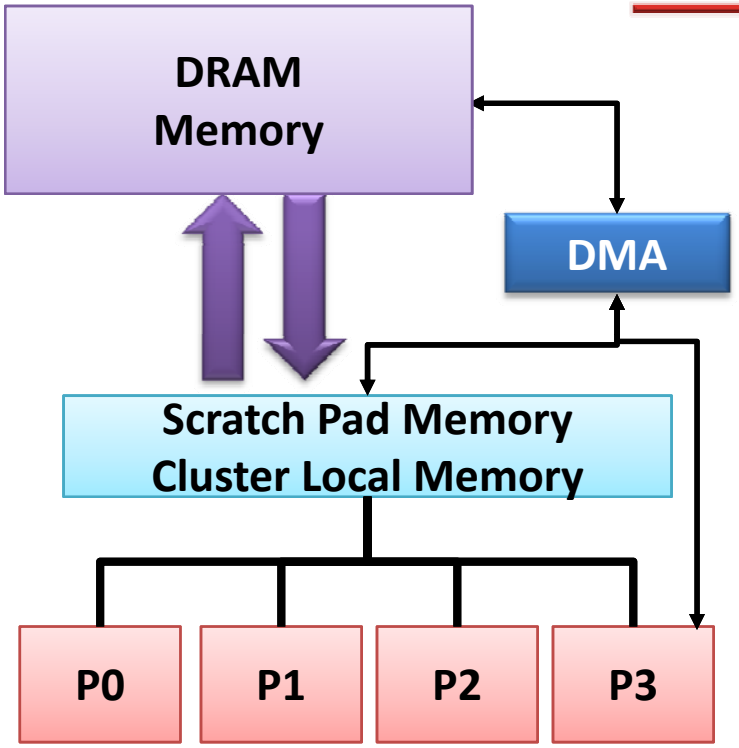
Use DMA to get Data to memory from Disk and Send Data to disk from memory

DMA data transfer in FFT

- Assume FFT compute for a Row takes= 2µS
- Transferring a row to/from memory = 4µS using DMA channel
- You can use 4 DMA channel of 8237A effectively to utilize the transfer
- 2 DMA channel for Receiving Data to memory and 2 DMA channel for Sending data to Disk



Realistic Advances Cases: Multiprocessor




```
DataMove(dcs_p0); DataMove(dcs_p1);
DataMove(dcs_p2); DataMove(dcs_p3);

DoForP0(); DoForP1(); DoForP2(); DoForP3();
waitP0(); waitP1(); waitP2(); waitP3();

DataMovewait(dcs_p0); DataMovewait(dcs_p1);
DataMovewait(dcs_p2); DataMovewait(dcs_p3);
```

Example: Cradle CT3400, NVidia Graphics Card

Reference

- R S Gaonkar, “Microprocessor Architecture”, Chapter 15
- R S Gaonkar, “Microprocessor Architecture”, Appendix D
- Software Pipelineing:
  - pages.cs.wisc.edu/~fischer/cs701.f08/softpipe.pdf
- NVidia GPU
- Cradle Multi DSP Chip
- FFT & Matrix multiplication

Thanks