

# ESE5320: System-on-a-Chip Architecture

Day 14: October 18, 2023  
Orchestrating Data in Memories

Midterm Average: 57  
Std. Dev.: 12  
Will take max(midterm,final)  
for midterm grade



Penn ESE5320 Fall 2023 -- DeHon

1

## Today

- DRAM (Part 1)
- DRAM Implications (Part 2)
- Tricks for working with DRAM and generally minimizing use of large memories
  - Data Reuse in Images (Part 3)
  - Sort (time permitting) (Part 4)

Penn ESE5320 Fall 2023 -- DeHon

2

2

## Message

- How we access data matters
- Reuse data to avoid access to large memories
- Think about how organize computation and data to minimize
  - Use of large memory
  - Data movement
- ...and think about what you expect, so can diagnose unnecessary data bottlenecks

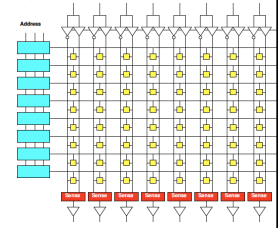
Penn ESE5320 Fall 2023 -- DeHon

3

3

## Day 3: Memory Scaling

- Small memories are fast
  - Large memories are slow
- Small memories low energy
  - Large memories high energy
- Large memories dense
  - Small memories cost more area per bit
- Combining:
  - Dense memories are slow



Penn ESE5320 Fall 2023 -- DeHon

4

## Day 3: Lesson

- Cheaper to access wide/contiguous blocks memory
  - In hardware
  - From the architectures typically build
- Can achieve higher bandwidth on large block data transfer
  - Than random access of small data items

Penn ESE5320 Fall 2023 -- DeHon

5

5

DRAM  
Dynamic  
Random Access Memory  
Usually the largest memory

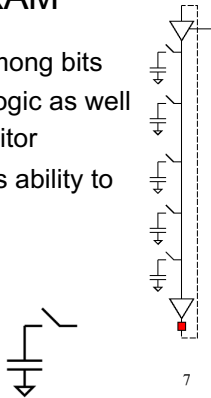
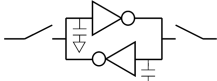
Penn ESE5320 Fall 2023 -- DeHon

6

6

## Dynamic RAM

- Shares even more logic among bits
- Share refresh/restoration logic as well
- Minimal storage is a capacitor
- “Feature” DRAM process is ability to make capacitors efficiently
- Denser, but slower



Penn ESE5320 Fall 2023 -- DeHon

7

## DRAM

- 1GB DDR3 SDRAM from Micron

- <http://www.micron.com/products/dram/ddr3/>
- 96 pin package
- 16b datapath IO
- Operate at 500+MHz
- 37.5ns random access latency

### Options

- Configuration
  - 64 Meg x 16 (8 Meg x 16 x 8 banks)
  - 128 Meg x 8 (16 Meg x 8 x 8 banks)
  - 256 Meg x 4 (32 Meg x 4 x 8 banks)
- FBGA package (lead-free)
  - x4, x8; 96-ball FBGA (9mm x 15.5mm)
  - x16; 96-ball FBGA (9mm x 15.5mm)
- Timing - cycle time
  - 2.5ns @ CL = 6 (DDR3-800)
  - 2.5ns @ CL = 5 (DDR3-800)
  - 1.87ns @ CL = 8 (DDR3-1066)
  - 1.87ns @ CL = 7 (DDR3-1066)
  - 1.5ns @ CL = 10 (DDR3-1333)
  - 1.5ns @ CL = 9 (DDR3-1333)

### Marking

- 64M16
- 128M8
- 256M4
- BY
- LA
- 25
- 25E
- 187
- 187E
- 15
- 15E

Table 1: Key Timing Parameters

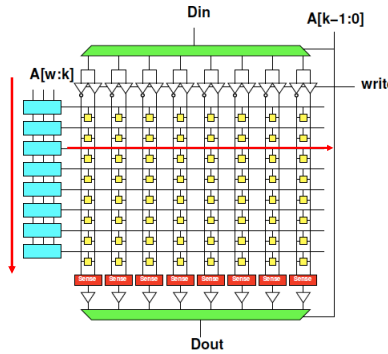
Speed Grade	Data Rate (Mbits)	Target RCD/TP/CL	<sup>t</sup> RCD (ns)	<sup>t</sup> RP (ns)	CL (ns)
-25E	800	5-5-5	12.5	12.5	12.5
-25	800	6-6-6	15	15	15
-187E	1,066	7-7-7	13.1	13.1	13.1
-187	1,066	8-8-8	15	15	15
-15E	1,333	9-9-9	13.5	13.5	13.5
-15	1,333	10-10-10	15	15	15

Penn ESE5320 Fall 2023 -- DeHon

8

## Memory Access Timing

- Access:
  1. Address (RCD)

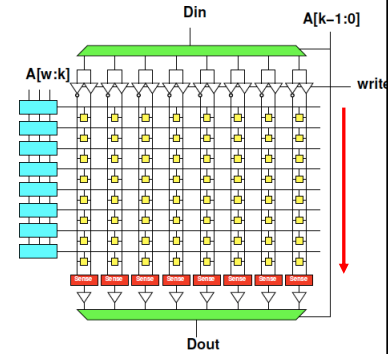


Penn ESE5320 Fall 2023 -- DeHon

9

## Memory Access Timing

- Access:
  1. Address (RCD)
  2. Row fetch (RP)

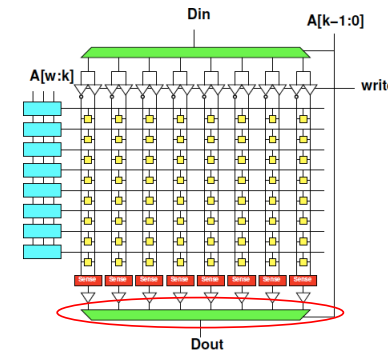


Penn ESE5320 Fall 2023 -- DeHon

10

## Memory Access Timing

- Access:
  1. Address (RCD)
  2. Row fetch (RP)
  3. Column select (CL)

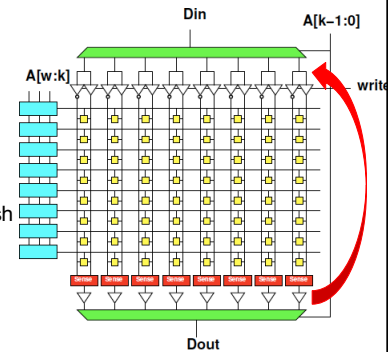


Penn ESE5320 Fall 2023 -- DeHon

11

## Memory Access Timing

- Access:
  1. Address (RCD)
  2. Row fetch (RP)
  3. Column select (CL)
  4. writeback/refresh



Penn ESE5320 Fall 2023 -- DeHon

12

## Dynamic RAM

- Shares even more logic among bits
- Share refresh/restoration logic as well
- Minimal storage is a capacitor
- “Feature” DRAM process is ability to make capacitors efficiently
- Denser, but slower

13

Penn ESE5320 Fall 2023 -- DeHon

13

## Memory Access Timing

- Access:
  1. Address (RCD)
  2. Row fetch (RP)
  3. Column select (CL)
  4. writeback/refresh

14

Penn ESE5320 Fall 2023 -- DeHon

14

## Memory Access Timing

- Access:
  1. Address (RCD)
  2. Row fetch (RP)
  3. Column select (CL)
  4. writeback/refresh
- Optimization for access within a row

15

Penn ESE5320 Fall 2023 -- DeHon

15

## DRAM

- 1GB DDR3 SDRAM from Micron
  - <http://www.micron.com/products/dram/ddr3/>
  - 96 pin package
  - 16b datapath IO
  - Operate at 500+MHz
  - 37.5ns random access latency

Speed Grade	Data Rate (Mbit/s)	Target RCD/TP/CL	tRCD (ns)	tRP (ns)	CL (ns)
-25E	800	5-5-5	12.5	12.5	12.5
-25	800	6-6-6	15	15	15
-187E	1,066	7-7-7	13.1	13.1	13.1
-187	1,066	8-8-8	15	15	15
-15E	1,333	9-9-9	13.5	13.5	13.5
-15	1,333	10-10-10	15	15	15

16

Penn ESE5320 Fall 2023 -- DeHon

16

## Memory Access Timing

- Access
  1. Address (RCD)
  2. Row fetch (RP)
  3. Column select
    - CL: Can repeat
  4. writeback/refresh
- Row  
1024b—8192b
- Faster to access within row

17

Penn ESE5320 Fall 2023 -- DeHon

17

## Preclass 4

- Preclass 4 toy warmup
  - Reading n 16b words takes: 10+n cycles
    - 10 cycles for access latency (RCD, RP)
    - 1 cycle to select each word (CL)
    - n cycles for n of them
- % time in access latency
  - Reading 1 word
  - Reading 16 words
  - Reading 128 words

18

Penn ESE5320 Fall 2023 -- DeHon

18

# DRAM

- 1GB DDR3 SDRAM from Micron

- <http://www.micron.com/products/dram/ddr3/>
- 96 pin package
- 16b datapath IO
- Operate at 500+MHz
- 37.5ns random access latency

- |  |  |
|--|--|
| <b>Options</b>   | <b>Marking</b>   |
| <ul style="list-style-type: none"> <li>Configuration                     <ul style="list-style-type: none"> <li>64 Meg x 16 (8 Meg x 16 x 8 banks)</li> <li>128 Meg x 8 (16 Meg x 8 x 8 banks)</li> <li>256 Meg x 4 (32 Meg x 4 x 8 banks)</li> </ul> </li> <li>FBGA package (lead-free)                     <ul style="list-style-type: none"> <li>x4, x8; 96-ball FBGA (9mm x 15.5mm)</li> <li>x16; 96-ball FBGA (9mm x 15.5mm)</li> </ul> </li> <li>Timing - cycle time                     <ul style="list-style-type: none"> <li>2.5ns @ CL=6 (DDR3-800)</li> <li>2.5ns @ CL=5 (DDR3-800)</li> <li>1.87ns @ CL=8 (DDR3-1066)</li> <li>1.87ns @ CL=7 (DDR3-1066)</li> <li>1.5ns @ CL=10 (DDR3-1333)</li> <li>1.5ns @ CL=9 (DDR3-1333)</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>64M16</li> <li>128M8</li> <li>256M4</li> <li>BY</li> <li>LA</li> <li>-25</li> <li>-25E</li> <li>-187</li> <li>-187E</li> <li>-15</li> <li>-15E</li> </ul> |

Table 1: Key Timing Parameters

Speed Grade	Data Rate (Mbits)	Target RCD/RP/CL	tRCD (ns)	tRP (ns)	CL (ns)
-25E	800	5-5-5	12.5	12.5	12.5
-25	800	6-6-6	15	15	15
-187E	1,066	7-7-7	13.1	13.1	13.1
-187	1,066	8-8-8	15	15	15
-15E	1,333	9-9-9	13.5	13.5	13.5
-15	1,333	10-10-10	15	15	15

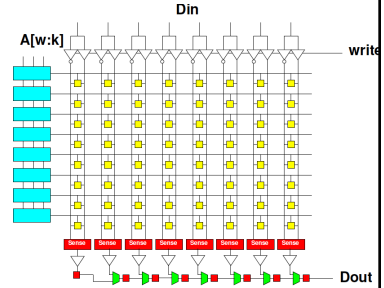
Penn ESE5320 Fall 2023 -- DeHon

19

19

# DRAM Streaming

- Can design to shift out large data from row



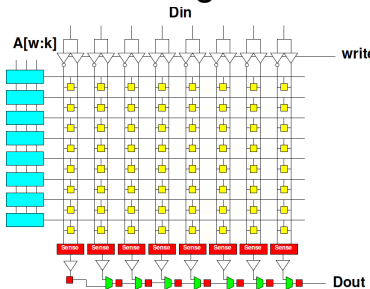
Penn ESE5320 Fall 2023 -- DeHon

20

20

# DRAM Streaming

- Reading row is 2x12.5ns
  - (RCD,RP)
- 16b @ 500MHz
- 1024b row
- 1024/16
  - 64 words per row



- How long (ns) to provide full row (64, 16b words) from address?

Penn ESE5320 Fall 2023 -- DeHon

21

21

# DRAM Streaming

- Can read from wide memory, but only have 16 pins to bring off chip
- But, can clock across those 16 pins at high clock rate compared to memory reference (2ns vs. 12.5ns)
- So, can sequentially shift the data out the 16 pins, if grabbing all the data for the row

Penn ESE5320 Fall 2023 -- DeHon

22

22

# DRAM

- 1GB DDR3 SDRAM from Micron

- <http://www.micron.com/products/dram/ddr3/>
- 96 pin package
- 16b datapath IO
- Operate at 500+MHz
- 37.5ns random access latency

- |  |  |
|--|--|
| <b>Options</b>   | <b>Marking</b>   |
| <ul style="list-style-type: none"> <li>Configuration                     <ul style="list-style-type: none"> <li>64 Meg x 16 (8 Meg x 16 x 8 banks)</li> <li>128 Meg x 8 (16 Meg x 8 x 8 banks)</li> <li>256 Meg x 4 (32 Meg x 4 x 8 banks)</li> </ul> </li> <li>FBGA package (lead-free)                     <ul style="list-style-type: none"> <li>x4, x8; 96-ball FBGA (9mm x 15.5mm)</li> <li>x16; 96-ball FBGA (9mm x 15.5mm)</li> </ul> </li> <li>Timing - cycle time                     <ul style="list-style-type: none"> <li>2.5ns @ CL=6 (DDR3-800)</li> <li>2.5ns @ CL=5 (DDR3-800)</li> <li>1.87ns @ CL=8 (DDR3-1066)</li> <li>1.87ns @ CL=7 (DDR3-1066)</li> <li>1.5ns @ CL=10 (DDR3-1333)</li> <li>1.5ns @ CL=9 (DDR3-1333)</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>64M16</li> <li>128M8</li> <li>256M4</li> <li>BY</li> <li>LA</li> <li>-25</li> <li>-25E</li> <li>-187</li> <li>-187E</li> <li>-15</li> <li>-15E</li> </ul> |

Table 1: Key Timing Parameters

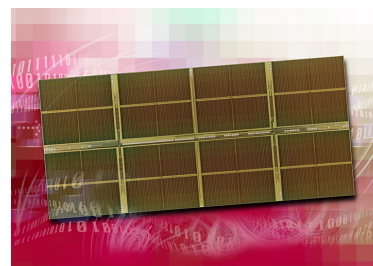
Speed Grade	Data Rate (Mbits)	Target RCD/RP/CL	tRCD (ns)	tRP (ns)	CL (ns)
-25E	800	5-5-5	12.5	12.5	12.5
-25	800	6-6-6	15	15	15
-187E	1,066	7-7-7	13.1	13.1	13.1
-187	1,066	8-8-8	15	15	15
-15E	1,333	9-9-9	13.5	13.5	13.5
-15	1,333	10-10-10	15	15	15

Penn ESE5320 Fall 2023 -- DeHon

23

23

# 1 Gigabit DDR2 SDRAM



[Source: <http://www.elpida.com/en/news/2004/11-18.html>]

Penn ESE5320 Fall 2023 -- DeHon

24

24

## DIMMS multi-DRAM

- DIMMs usually pack multiple DRAM chips, c, in parallel
  - Can access  $c \cdot 16b$  per transfer
  - Typically 64, 128, 256b wide
    - 8 chips = 128b wide



Penn ESE5320 Fall 2023 -- DeHon

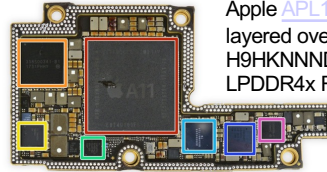
<https://commons.wikimedia.org/wiki/File:DIMMs.jpg>

25

25

## Smart Phone

- Will use single DRAM chip
- iPhone/iPod – flip chip in package with SoC Processor



Apple [APL1W72](#) A11 Bionic SoC layered over SK Hynix H9HKNNNDBMAUUR 3 GB LPDDR4x RAM

Penn ESE5320 Fall 2023 -- DeHon

Source:

<https://www.ifixit.com/Teardown/iPhone+X+Teardown/98975>

26

26

## DRAM Implications

### Part 2

Penn ESE5320 Fall 2023 -- DeHon

27

27

## Terminology Watch

- Need to talk about
  - DRAM Rows
  - Image Rows
  - Relation between DRAM and Image Rows

Penn ESE5320 Fall 2023 -- DeHon

28

28

## Preclass 1

```
uint16_b dram_image[1024*1024]; // assum
for (y=0; y<1024;y++)
    for (x=0; x<1024;x++)
        astream.write(dram_image[1024*y+x])
```

- How many cycles to run loop?
  - 10 cycle on dram row change (RCD,RP,CL)
  - 3 cycles same row (CL)
  - 128 uint16\_b per DRAM row (2048b)

Penn ESE5320 Fall 2023 -- DeHon

29

29

## Image Row and DRAM Rows

Address	MSB	-- DRAM Row Bytes				-- LSB
0x027FF0	[y-1,1023]					
0x028000	[y,127]					[y,0]
0x028100						[y,128]
0x028200						
0x028300						
0x028400						
0x028500						
0x028600						
0x028700	[y,1023]					[y,896]
0x028800						[y+1,0]
0x028900						
0x028a00						

Penn ESE5320 Fall 2023 -- DeHon

30

30

## Preclass 2

```
uint16_b dram_image[1024*1024];
for (x=0; x<1024;x++)
    for (y=0; y<1024;y++)
        astream.write(dram_image[1024*y+x]);
```

Note loop  
interchange

- How many cycles to run loop?
  - 10 cycle on row change
  - 3 cycles same row
  - 128 uint16\_b per DRAM row (2048b)

Penn ESE5320 Fall 2023 -- DeHon

31

31

## Preclass 3

```
uint16_b dram_image[1024*1024];
for (y=0; y<1024;y++)
    for (x=0; x<1024;x+=128)
        // assume the following for loop can be converted to a stream
        // as noted above, so takes 10+n cycles for n uint16_b's
        for (i=0; i<128; i++) astream.write(dram_image[1024*y+x+i])
```

- How many cycles to run loop?
  - 10+n cycles to fetch n<=128 Half-words on row

Penn ESE5320 Fall 2023 -- DeHon

32

32

## DRAM

- Latency is large (10s of ns)
- Throughput can be high (GB/s)
  - If accessed sequentially
  - If exploit wide-word block transfers
- Throughput low on random accesses
  - As we saw for random access on wide-word memory
- Integer factor – order of magnitude -- difference

Penn ESE5320 Fall 2023 -- DeHon

33

33

## Xilinx/Vitis DMA?

- This section focused on what the **DRAM** can do
  - What should happen with an optimized DRAM controller
  - Only DRAM is contributing latency

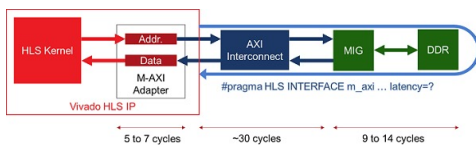
Penn ESE5320 Fall 2023 -- DeHon

34

34

## Xilinx AXI Connection to DRAM

- Additional Latency from on-chip interconnect and interface



Source: [https://www.xilinx.com/html\\_docs/xilinx2020\\_1/vitis\\_doc/programmingvitis.html](https://www.xilinx.com/html_docs/xilinx2020_1/vitis_doc/programmingvitis.html)

Penn ESE5320 Fall 2023 -- DeHon

35

35

## Xilinx/Vitis DMA?

- This section focused on what the **DRAM** can do
  - What should happen with an optimized DRAM controller
- Vitis DMA requires setup time
  - Which will be **in addition** to DRAM access time
  - High overhead [as you will see on HW6]
    - Even higher premium for large block transfers

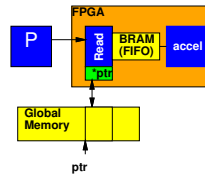
Penn ESE5320 Fall 2023 -- DeHon

36

36

## Day 13: Data Movement (Options)

- DMA burst copy into BRAM memories
  - Built on top of other with DMA copy loop



```
hls::stream<datatype_t> str;
INPUT_READ: for(int i=0; i<INPUT_SIZE; i++) {
    #pragma HLS PIPELINE
    str.write(inp[i]); // Reading from Input interface
}
```

Penn ESE5320 Fall 2023 -- DeHon

37

37

## Maximize Reuse Part 3

Minimize need to go to large memory

Penn ESE5320 Fall 2023 -- DeHon

38

38

## Data Management

- Large, central memory
  - Bottleneck
  - High energy
- Minimize need to return to it

Penn ESE5320 Fall 2023 -- DeHon

39

39

## Window Filter

- Compute based on neighbors
  - $F(d[y-1][x-1], d[y-1][x], d[y-1][x+1], d[y][x-1], d[y][x], d[y][x+1], d[y+1][x-1], d[y+1][x], d[y+1][x+1])$
- Common idiom
  - Image filtering
  - CNN (Convolution stage in Deep Neural Network)
  - Numerical simulation
  - Cellular automata
  - Search, matching

Penn ESE5320 Fall 2023 -- DeHon

40

40

## Window Filter

- Compute based on neighbors
  - $F(d[y-1][x-1], d[y-1][x], d[y-1][x+1], d[y][x-1], d[y][x], d[y][x+1], d[y+1][x-1], d[y+1][x], d[y+1][x+1])$
- Generalize for different neighborhood sizes
  - for (yoff=YMIN; yoff<YMAX;yoff++)
    - for (xoff=XMIN; xoff<XMAX;xoff++)
      - res=fun(res,d[y+yoff][x+xoff]);

Penn ESE5320 Fall 2023 -- DeHon

41

41

## Window Filter

- Compute based on neighbors
- for (y=0;y<YMAX;y++)
  - for (x=0;x<XMAX;x++)
    - $o[y][x]=F(d[y-1][x-1], d[y-1][x], d[y-1][x+1], d[y][x-1], d[y][x], d[y][x+1], d[y+1][x-1], d[y+1][x], d[y+1][x+1]);$
- How work with image in DRAM
  - As written references
  - Reads per row?

Penn ESE5320 Fall 2023 -- DeHon

42

42

## Window Filter

- Compute based on neighbors
- for ( $y=0; y<YMAX; y++$ )  
 for ( $x=0; x<XMAX; x++$ )  

$$o[y][x]=F(d[y-1][x-1], d[y-1][x], d[y-1][x+1],$$

$$d[y][x-1], d[y][x], d[y][x+1],$$

$$d[y+1][x-1], d[y+1][x], d[y+1][x+1]);$$
- Data reused between inner loop invocations when  $x$  increments?  
 – How many of the 9 values are reused at  $x+1$ ?

Penn ESE5320 Fall 2023 -- DeHon

43

43

## Window Shift $x++$

	23	24	25	26	27
17					
18		Y-1,X-1	Y-1,X	Y-1,X+1	
19		Y,X-1	Y,X	Y,X+1	
20		Y+1,X-1	Y+1,X	Y+1,X+1	
21					

	23	24	25	26	27
17					
18			Y-1,X-1	Y-1,X	Y-1,X+1
19			Y,X-1	Y,X	Y,X+1
20			Y+1,X-1	Y+1,X	Y+1,X+1
21					

Penn ESE5320 Fall 2023 -- DeHon

44

44

## Window Filter

- Compute based on neighbors
- for ( $y=0; y<YMAX; y++$ )  
 for ( $x=0; x<XMAX; x++$ )  

$$o[y][x]=F(d[y-1][x-1], d[y-1][x], d[y-1][x+1],$$

$$d[y][x-1], d[y][x], d[y][x+1],$$

$$d[y+1][x-1], d[y+1][x], d[y+1][x+1]);$$
- Data reused between  $y$  values in outer loop?  
 – How many of 9 values reused for  $y+1$ ?  
 – Reuse distance [in cycles]?
  - ...assume  $ll=1$  pipeline inner loop
  - i.e. how many  $o[y][x]$ 's do we compute between using as  $d[y+1][x]$  and  $d[y][x]$ ?

Penn ESE5320 Fall 2023 -- DeHon

45

45

## Window Shift $y++$

	23	24	25	26	27
17					
18		Y-1,X-1	Y-1,X	Y-1,X+1	
19		Y,X-1	Y,X	Y,X+1	
20		Y+1,X-1	Y+1,X	Y+1,X+1	
21					

	23	24	25	26	27
17					
18					
19		Y-1,X-1	Y-1,X	Y-1,X+1	
20		Y,X-1	Y,X	Y,X+1	
21		Y+1,X-1	Y+1,X	Y+1,X+1	

Penn ESE5320 Fall 2023 -- DeHon

46

46

## Window Filter

- Compute based on neighbors
- for ( $y=0; y<YMAX; y++$ )  
 for ( $x=0; x<XMAX; x++$ )  

$$o[y][x]=F(d[y-1][x-1], d[y-1][x], d[y-1][x+1],$$

$$d[y][x-1], d[y][x], d[y][x+1],$$

$$d[y+1][x-1], d[y+1][x], d[y+1][x+1]);$$
- How much data need to store to hold on to  $d[y]$  values that will use again?
- Where can we store?

Penn ESE5320 Fall 2023 -- DeHon

47

47

## Line Buffers

- Only need to save last 2 rows ( $d[y][*]$ )
  - General case:  $(window\_height-1)$  rows
- Store in local buffers
  - Call them  $dy[XMAX]$ ,  $dym[XMAX]$ ;
    - $dym$  for  $dy$  minus 1
  - Stick value there between uses.

Penn ESE5320 Fall 2023 -- DeHon

48

48



## Variable Naming

		+xm		+xp	
dym		Y-1,X-1	Y-1,X	Y-1,X+1	
dy		Y,X-1	Y,X	Y,X+1	
dyp		Y+1,X-1	Y+1,X	Y+1,X+1	

		+xm		+xp	
dym		Y-1,X-1	Y-1,X	Y-1,X+1	
dy		Y,X-1	Y,X	Y,X+1	
dyp		dypxm	dypx	dnew	

Penn ESE5320 Fall 2023 -- DeHon 49

59

## Window Filter

- With line buffers dy, dym
- for (y=0;y<YMAX;y++)  
 for (x=0;x<XMAX;x++) {  
   dypxm=dypx; dypx=dnew; dnew=d[y+1][x+1];  
   o[y][x]=F(dym[x-1],dym[x],dym[x+1],  
           dy[x-1],dy[x],dy[x+1],  
           dypxm,dypx,dnew);  
   dym[x-1]=dy[x-1];  
   dy[x-1]=dypxm;  
 }

Penn ESE5320 Fall 2023 -- DeHon 50

50

## Lines in Buffers

		+xm		+xp	
dym		Y-1,X-1	Y-1,X	Y-1,X+1	
dy		Y,X-1	Y,X	Y,X+1	
dyp		Y+1,X-1	Y+1,X	Y+1,X+1	

Penn ESE5320 Fall 2023 -- DeHon 51

51

## Window Filter

- With line buffers dy, dym
- for (y=0;y<YMAX;y++)  
 for (x=0;x<XMAX;x++) {  
   dypxm=dypx; dypx=dnew; dnew=d[y+1][x+1];  
   o[y][x]=F(dym[x-1],dym[x],dym[x+1],  
           dy[x-1],dy[x],dy[x+1],  
           dypxm,dypx,dnew);  
   dym[x-1]=dy[x-1];dy[x-1]=dypxm; }  
- **Avoid multiple read dy, dym?**  
 – **Save memory ports**

Penn ESE5320 Fall 2023 -- DeHon 52

52

## Variable Naming

		+xm		+xp	
dym		Y-1,X-1	Y-1,X	Y-1,X+1	
dy		Y,X-1	Y,X	Y,X+1	
dyp		Y+1,X-1	Y+1,X	Y+1,X+1	

		+xm		+xp	
dym		dymxm	dymx	dymxp	
dy		dyxm	dyx	dyxp	
dyp		dypxm	dypx	dnew	

Penn ESE5320 Fall 2023 -- DeHon 53

53

## Window Filter

- Single read and write from dym, dy
- for (y=0;y<YMAX;y++)  
 for (x=0;x<XMAX;x++) {  
   dypxm=dypx; dypx=dnew; dnew=d[y+1][x+1];  
   dymxm=dymx; dymx=dymxp; dymxp=dym[x+1];  
   o[y][x]=F(dymxm,dymx,dymxp,  
           dyxm,dyx,dyxp,  
           dypxm,dypx,dnew);  
   dym[x-1]=dymxm;dy[x-1]=dypxm; }

Penn ESE5320 Fall 2023 -- DeHon 54

54

## Pipelined Circuit

- Single **sequential** read from big memory (DRAM) per result
- Shift registers
- Line buffers

55

Penn ESE5320 Fall 2023 -- DeHon

55

## Window Filter

- Convert to stream ... use Preclass 3 load
- for (y=0;y<YMAX;y++)  
 for (x=0;x<XMAX;x++) {  
   dypxm=dypx; dypx=dnew; **astream.read(dnew);**  
   dyxm=dyx; dyx=dyxp; dyxp=dy[x+1];  
   dymxm=dymx; dymx=dymxp; dymxp=dym[x+1];  
   o[y][x]=F(dymxm,dymx,dympx,  
           dyxm,dyx,dyxp,  
           dypxm,dypx,dnew);  
   dym[x-1]=dymx;dy[x-1]=dypxm; }

56

Penn ESE5320 Fall 2023 -- DeHon

56

## Window Filter

- Convert to stream ... use Preclass 3 load
- for (y=0;y<YMAX;y++)  
 for (x=0;x<XMAX;x++) {  
   dypxm=dypx; dypx=dnew; astream.read(dnew);  
   dyxm=dyx; dyx=dyxp; dyxp=dy[x+1];  
   dymxm=dymx; dymx=dymxp; dymxp=dym[x+1];  
   **ostream.write**(F(dymxm,dymx,dympx,  
                   dyxm,dyx,dyxp,  
                   dypxm,dypx,dnew));  
   dym[x-1]=dymx;dy[x-1]=dypxm; }

57

Penn ESE5320 Fall 2023 -- DeHon

57

## Multiple Filters

- What if want to run multiple filters?
- for (f=0;f<FMAX;f++)  
 for (y=0;y<YMAX;y++)  
 for (x=0;x<XMAX;x++)  
   **of[f][y][x]=fun[f](d[y-1][x-1],d[y-1][x],d[y-1][x+1],  
                       d[y][x-1],d[y][x],d[y][x+1],  
                       d[y+1][x-1],d[y+1][x],d[y+1][x+1]);**
- Forces multiple reads through d
- **Avoidable?**

58

Penn ESE5320 Fall 2023 -- DeHon

58

## Take-Away

- Can process with a single, streaming read through data in large memory
  - Specifically for window filters on images
  - General goal to try to achieve
- Careful to orchestrate data in small, local buffers
- DMA/streaming transfers to achieve near peak DRAM bandwidth

59

Penn ESE5320 Fall 2023 -- DeHon

59

## Sort Part 4

Cover in lecture session if time permits

[Skip to wrapup](#)

60

Penn ESE5320 Fall 2023 -- DeHon

60

## Sort

- Order data  
– 7 3 4 2 1 9 8 0 → 0 1 2 3 4 7 8 9
- Data movement can be challenging  
– Last value may become first (vice-versa)  
– Especially when not fit in small memories
- Many sequential sort solutions access data irregularly  
– Want to avoid with DRAM
- Example illustrates can reformulate as streaming

Penn ESE5320 Fall 2023 -- DeHon

61

61

## Observation: Merge

- Merging two sorted list is a streaming operation
- `int aptr; int bptr;`
- `for (i=0;i<MCNT;i++)`  
  `If ((aptr<ACNT) && (bptr<BCNT))`  
    `If (a[aptr]>b[bptr])`  
      `{ o[i]=a[aptr]; aptr++; }`  
    `Else`  
      `{ o[i]=b[bptr]; bptr++; }`  
  `Else // copy over remaining from a or b`

Penn ESE5320 Fall 2023 -- DeHon

62

62

## Merge Memory Access

- `for (i=0;i<MCNT;i++)`  
  `If ((aptr<ACNT) && (bptr<BCNT))`  
    `If (a[aptr]>b[bptr])`  
      `{ o[i]=a[aptr]; aptr++; }`  
    `Else`  
      `{ o[i]=b[bptr]; bptr++; }`  
  `Else // copy over remaining from a or b`

- How work with DRAM rows?

Penn ESE5320 Fall 2023 -- DeHon

63

63

## Merge and DRAM

- May alternate between DRAM rows
- `int aptr; int bptr;`
- `For (i=0;i<MCNT;i++)`  
  `If ((aptr<ACNT) && (bptr<BCNT))`  
    `If (a[aptr]>b[bptr])`  
      `{ o[i]=a[aptr]; aptr++; }`  
    `Else`  
      `{ o[i]=b[bptr]; bptr++; }`  
  `Else // copy over remaining from a or b`

Penn ESE5320 Fall 2023 -- DeHon

64

64

## Merge and DRAM

- Want to fetch a as block into local memory (into stream), then can do this
- `int aptr; int bptr;`
- `For (i=0;i<MCNT;i++)`  
  `If ((aptr<ACNT) && (bptr<BCNT))`  
    `If (a[aptr]>b[bptr])`  
      `{ o[i]=a[aptr]; aptr++; }`  
    `Else`  
      `{ o[i]=b[bptr]; bptr++; }`  
  `Else // copy over remaining from a or b`

Penn ESE5320 Fall 2023 -- DeHon

65

65

## Merge using Streams

- Merging two sorted list is a streaming operation
- `int aptr; int bptr;`
- `astream.read(ain); bstream.read(bin)`
- `for (i=0;i<MCNT;i++)`  
  `If ((aptr<ACNT) && (bptr<BCNT))`  
    `If (ain>bin)`  
      `{ ostream.write(ain); aptr++; astream.read(ain);}`  
    `Else`  
      `{ ostream.write(bin) bptr++; bstream.read(bin);}`  
  `Else // copy over remaining from astream/bstream`

Penn ESE5320 Fall 2023 -- DeHon

66

## Merge using Streams and DRAM

- Streaming accomplishes if Streaming is implemented to fetch row into FIFO buffer
- Int ptr; int bptr;
- astream.read(ain); bstream.read(bin)
- For (i=0;i<MCNT;i++)
  - If ((aptr<ACNT) && (bptr<BCNT))
  - If (ain>bin)
  - { ostream.write(ain); aptr++; astream.read(ain);}
  - Else
  - { ostream.write(bin) bptr++; bstream.read(bin);}
- Else // copy over remaining from astream/bstream

Penn ESE5320 Fall 2023 -- DeHon

67

## Building from Merge

- Know how to get sorted data of length N given two, independently sorted lists of length N/2
- A list of length 1 is sorted.
- Merging two lists of length 1 gives us?
  - How many lists of what length?
- Merging two lists of length 2 gives us?

Penn ESE5320 Fall 2023 -- DeHon

68

## Merge Sort

- Strategy
  - For  $i=1$  to **passes**
    - Pass over data creating merged sequences of length  $2^i$  from pairs of length  $2^{i-1}$
  - What is **passes**?
- Can, at least, limit to **passes** reads through data from large memory.

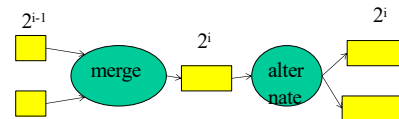
Penn ESE5320 Fall 2023 -- DeHon

69

69

## On-chip Merge

- As long as can fit sequence ( $2^i$ ) on chip, can stream merge stages without going back to big memory (DRAM)



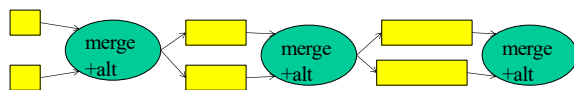
Penn ESE5320 Fall 2023 -- DeHon

70

70

## On-chip Merge

- As long as can fit sequence ( $2^i$ ) on chip, can stream merge stages without going back to big memory (DRAM)
  - E.g. with 36Kb BRAMs ~ first 10-12 stages
    - Up to ~16 using multiple BRAMs



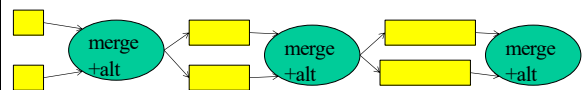
Penn ESE5320 Fall 2023 -- DeHon

71

71

## On-chip Merge

- For sorts up to ~100K elements may be able to cover in one read/write stream to/from DRAM
- Larger:  $\sim \log_2(N)$ -15 read/writes



Penn ESE5320 Fall 2023 -- DeHon

72

72

## Big Ideas

- Must pay attention to orchestrate data movement
  - How we access and reuse data
- Minimize use of large memories
- Regular (sequential) access to large memories higher performance
  - Easier to get bandwidth on wide-word reads

Penn ESE5320 Fall 2023 -- DeHon

73

73

## Admin

- Feedback
- HW6 due Friday
- Reading for Monday on web
- HW7: out soon....

Penn ESE5320 Fall 2023 -- DeHon

74

74