


ESE5320: System-on-a-Chip Architecture

Day 3: September 11, 2023
Memory

Preclass



Penn ESE5320 Fall 2023 -- DeHon

1

Today

- Memory
 - Memory (part 1)
 - Bottleneck
 - Scaling
 - Latency Engineering (part 2)
 - Scheduling
 - Data Reuse: Scratchpad and cache
 - Throughput [Bandwidth] Engineering (part 3)
 - Wide word

Penn ESE5320 Fall 2023 -- DeHon

2

Message

- Memory throughput and latency can be bottlenecks
- Minimize data movement
- Exploit small, local memories
- Exploit data reuse

Penn ESE5320 Fall 2023 -- DeHon

3

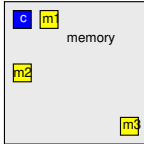
Memory Scaling

Penn ESE5320 Fall 2023 -- DeHon

4

On-Chip Delay

- Delay is proportional to distance travelled
- Make a wire twice the length
 - Takes twice the latency to traverse
 - (can pipeline)
- Modern chips
 - Run at GHz speeds
 - (1ns, sub 1ns cycle times)
 - Take 10s of ns to cross the chip
 - Takes 100s of ns to reference off-chip data
- **What does this say about placement of computations and memories?**



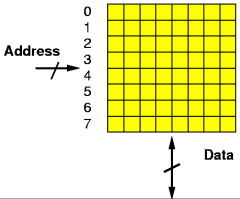
Penn ESE5320 Fall 2023 -- DeHon

5

Random Access Memory

- A Memory:
 - Series of locations (slots)
 - Can write values to a slot (specified by address)
 - Read values from (by address)
 - Return last value written

Notation:
slash on wire
means multiple bits wide



Penn ESE5320 Fall 2023 -- DeHon

6

Memory Operation

Memory Read

1. Present Address
2. Decoder Matches
3. Charge word line
4. Enables Memory Cell to charge bitlines (column)
5. SenseAmp detects bitline swing
6. Buffered for output

Cartoon memory
illustrative of physical layout

Penn ESE5320 Fall 2023 -- DeHon

7

Memory Block

- Linear wire delay with distance
- Assume constant area/memory bit
- N-bit memory, arranged in square
 - As shown $N=64$, width=depth=8
 - $N=\text{width} \times \text{depth}$ (since square)
- 4x capacity (N) delay change?
 - (e.g. 1Gbit to 4Gbit)

Penn ESE5320 Fall 2023 -- DeHon

8

Memory Block

- Linear wire delay with distance
- Assume constant area/memory bit
- N-bit memory, arranged in square
- Width relate to N?
- Depth?
- Delay scale N?

Penn ESE5320 Fall 2023 -- DeHon

9

Terminology Bandwidth

- Bandwidth – throughput for data movement
 - How many bits per second can we transfer

Penn ESE5320 Fall 2023 -- DeHon

10

Memory Block Bandwidth

- Bandwidth: data movement throughput
 - How many bits per second can we transfer
- N-bit memory, arranged in square
- Bits/cycle scale with N?
 - (not quite bandwidth)

Penn ESE5320 Fall 2023 -- DeHon

11

Memory Block Energy

- Assume read full width,
 - Energy scales with N
 - Activate \sqrt{N} wires
 - Each wire \sqrt{N} long
 - Or, energy/bit scales as \sqrt{N}
- Larger memories cost more energy

Penn ESE5320 Fall 2023 -- DeHon

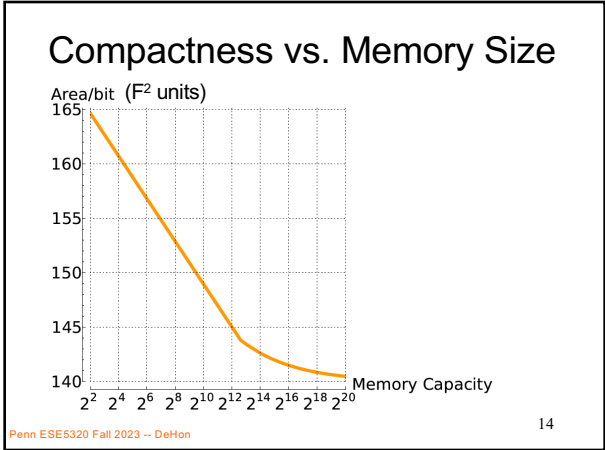
12

Memory Block Density

- Address, sense amplifiers are large
- **How do address bits scale with N?**
- Scale slower than bit area
 - Bits scale N
 - Address $\sqrt{N} \cdot \log(N)$
 - Sense amps \sqrt{N}
- **Large memories pay less per bit**
 - denser

Penn ESE5320 Fall 2023 -- DeHon

13



14

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

15

Latency Engineering Scheduling

Part 2

Penn ESE5320 Fall 2023 -- DeHon

16

Preclass 2

- Issue one read/write per cycle
- 10 cycle latency to memory
- Memory read concurrent with compute
 - only wait for memory when need value
- **Throughput (iters/cycle) in each case?**

Case 1:

```
for(i=0;i<MAX;i++) {
  in=a[i]; // memory read
  out=f(in); // 10 cycle compute
  b[i]=out;
}
```

Case 2:

```
next_in=a[0];
for(i=1;i<MAX;i++) {
  in=next_in;
  next_in=a[i];
  out=f(in);
  b[i-1]=out;
}
b[MAX-1]=f(next_in);
```

Pe

17

Preclass 2

- **Throughput (iters/cycle) in each case?**

Case 1:

```
for(i=0;i<MAX;i++) {
  in=a[i]; // memory read
  out=f(in); // 10 cycle compute
  b[i]=out;
}
```

Case 2:

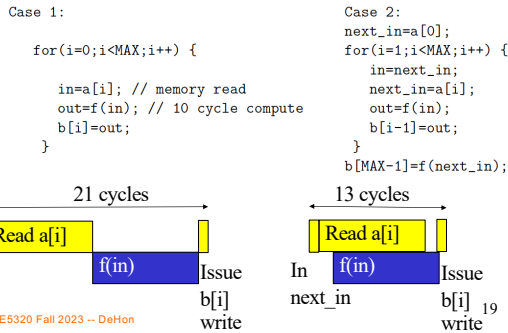
```
next_in=a[0];
for(i=1;i<MAX;i++) {
  in=next_in;
  next_in=a[i];
  out=f(in);
  b[i-1]=out;
}
b[MAX-1]=f(next_in);
```

Penn ESE5320 Fall 2023 -- DeHon

18

Preclass 2

- Throughput (iters/cycle) in each case?



19

Lesson

- Long memory latency can impact throughput
 - When must wait on it
 - When part of a cyclic dependency
- Overlap memory access with computations when possible
 - exploit parallelism between compute and memory access

Penn ESE5320 Fall 2023 -- DeHon

20

20

Latency Engineering Data Reuse

Penn ESE5320 Fall 2023 -- DeHon

21

21

Preclass 3abc

- MAX=10⁶ WSIZE=5
- How many times execute `t+=x[]*w[]` ?
- How many reads to `x[]` and `w[]`?

Runtime read x,w 20
t+=... 5

```
for (i=0;i<MAX;i++) {
    t=0;
    for (j=0;j<WSIZE;j++)
        t+=x[i+j]*w[j];
    y[i]=t;
}
```

Penn ESE5320 Fall 2023 -- DeHon

22

22

Preclass 3c

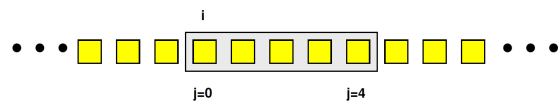
- $T = \text{MAX} * \text{WSIZE} (T_{\text{comp}} + T_x + T_w)$
 - $T = 10^6 * 5 * (5 + 20 + 20)$
 - $T = 2.25 * 10^8$
- ```
for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}
```

Penn ESE5320 Fall 2023 -- DeHon

23

23

## Preclass 3d



- How many distinct `x[]`, `w[]` read?

```
for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}
```

Penn ESE5320 Fall 2023 -- DeHon

24

24

## Strategy

- Add small, local memory bank
  - Small memories are faster

Proc  
local mem    large mem

Penn ESE5320 Fall 2023 -- DeHon 25

25

## Preclass 3e

Proc  
local mem    large mem

- How use local memory to reduce large memory reads to 3d?
- How much local memory need?

```

for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}

```

Penn ESE5320 Fall 2023 -- DeHon

26

## Preclass 3d

i  
j=0    j=4

- Which w's read in each iteration of j loop?
- Which x's?
  - i=0,
  - i=1,
  - Same i=0, i=1?
- When never read x[0] again? x[4]?

```

for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}

```

Penn ESE5320 Fall 2023 -- DeHon 27

27

## Recall from earlier in lecture: Memory Block

- Linear wire delay with distance
- Assume constant area/memory bit
- N-bit memory, arranged in square
- Width relate to N?
- Depth?
- Delay scale N?

width  
depth

Penn ESE5320 Fall 2023 -- DeHon 28

28

## Impact Small Memory

Proc  
local mem    large mem

- Remember how delay scales with N
- How much faster might small [10 element] memory be? [compare 10<sup>6</sup> as powers of 2: 2<sup>4</sup> vs. 2<sup>20</sup>]

```

for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}

```

Penn ESE5320 Fall 2023 -- DeHon

29

## Preclass 3e

```

for (j=0;j<WSIZE;j++) local_w[j]=w[j];
for (j=0;j<WSIZE-1;j++) local_x[j+1]=x[j];
for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE-1;j++)
 local_x[WSIZE-1]=x[i+WSIZE-1];
 t+=local_x[j]*local_w[j];
 y[i]=t;
}

```

Penn ESE5320 Fall 2023 -- DeHon 30

30

## Preclass 3e

- $WSIZE * T_w$
- $WSIZE * T_x$
- $WSIZE * MAX * T_{local}$
- $MAX * T_x$
- $WSIZE * MAX * T_{comp}$
- $WSIZE * MAX * 2 * T_{local}$

```

for (j=0; j<WSIZE; j++)
 local_w[j]=w[j];
for (j=0; j<WSIZE-1; j++)
 local_x[j+1]=x[j];
for (i=0; i<MAX; i++) {
 t=0;
 for (j=0; j<WSIZE-1; j++)
 local_x[j]=local_x[j+1];
 local_x[WSIZE-1]=x[i+WSIZE-1];
 for (j=0; j<WSIZE; j++)
 t+=local_x[j]*local_w[j];
 y[i]=t;
}

```

Penn ESE5320 Fall 2023 -- DeHon 31

31

## Preclass 3e iii

- $WSIZE * MAX * (T_{comp} + 3 * T_{local})$
- $+ WSIZE * (T_w + T_x)$
- $+ MAX * T_x$
- $5 * 10^6 * (5 + 3)$
- $+ 5 * (20 + 20)$
- $+ 10^6 * 20$
- $6 * 10^7$

```

for (j=0; j<WSIZE; j++)
 local_w[j]=w[j];
for (j=0; j<WSIZE-1; j++)
 local_x[j+1]=x[j];
for (i=0; i<MAX; i++) {
 t=0;
 for (j=0; j<WSIZE-1; j++)
 local_x[j]=local_x[j+1];
 local_x[WSIZE-1]=x[i+WSIZE-1];
 for (j=0; j<WSIZE; j++)
 t+=local_x[j]*local_w[j];
 y[i]=t;
}

```

Penn ESE5320 Fall 2023 -- DeHon 32

32

## Preclass 3e iii

```

for (i=0; i<MAX; i++) {
 t=0;
 for (j=0; j<WSIZE; j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}

```

```

for (j=0; j<WSIZE; j++)
 local_w[j]=w[j];
for (j=0; j<WSIZE-1; j++)
 local_x[j+1]=x[j];
for (i=0; i<MAX; i++) {
 t=0;
 for (j=0; j<WSIZE-1; j++)
 local_x[j]=local_x[j+1];
 local_x[WSIZE-1]=x[i+WSIZE-1];
 for (j=0; j<WSIZE; j++)
 t+=local_x[j]*local_w[j];
 y[i]=t;
}

```

- $2.25 * 10^8$
- $6 * 10^7$
- $3.75x$

Penn ESE5320 Fall 2023 -- DeHon 33

33

## Lesson

- Data can often be reused
    - Keep data needed for computation in
      - Closer, smaller (faster, less energy) memories
    - Reduces **latency** costs
    - Reduces **bandwidth** required from large memories
  - Reuse hint: value used multiple times
    - Or value produced/consumed
- Penn ESE5320 Fall 2023 -- DeHon 34

34

## Processor Data Caches

- Traditional Processor Data Caches are a heuristic instance of this
  - Add a small memory local to the processor
    - It is fast, low latency
  - Store anything fetched from large/remote memory in local memory
    - Hoping for reuse in near future
  - On every fetch, check local memory before go to large memory



Penn ESE5320 Fall 2023 -- DeHon

35

## Cache

- **Goal:** performance of small memory with density of large memory



Penn ESE5320 Fall 2023 -- DeHon

36

## Processor Data Caches

- Demands more than a small memory
  - Need to sparsely store address/data mappings from large memory
  - Makes more area/delay/energy expensive than just a simple memory of capacity
- Don't need explicit data movement
- Cannot control when data moved/saved
  - Bad for determinism
- Limited ability to control what stays in small memory simultaneously

Penn ESE5320 Fall 2023 -- DeHon

37

37

## Terminology

- Cache
  - Hardware-managed small memory in front of larger memory
- Scratchpad
  - Small memory
  - Software (or logic) managed
  - Explicit reference to scratchpad vs. large (other) memories
  - Explicit movement of data

Penn ESE5320 Fall 2023 -- DeHon

38

38

## Bandwidth Engineering

### Part 3

Penn ESE5320 Fall 2023 -- DeHon

39

39

## Terminology: Unroll Loop

- Replace loop with instantiations of body

```
For WSIZE=5
for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}

for (i=0;i<MAX;i++) {
 t=0;
 t+=x[i+0]*w[0];
 t+=x[i+1]*w[1];
 t+=x[i+2]*w[2];
 t+=x[i+3]*w[3];
 t+=x[i+4]*w[4];
 y[i]=t;
}
```

Penn ESE5320 Fall 2023 -- DeHon

40

40

## Terminology: Unroll Loop

- Replace loop with instantiations of body

```
For WSIZE=5
for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}

for (i=0;i<MAX;i++) {
 t=x[i+0]*w[0]+x[i+1]*w[1]+x[i+2]*w[2]+
 x[i+3]*w[3]+x[i+4]*w[4];
 y[i]=t;
}
```

Penn ESE5320 Fall 2023 -- DeHon

41

41

## Terminology: Unroll Loop

- Can unroll partially

```
For WSIZE even
for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j++)
 t+=x[i+j]*w[j];
 y[i]=t;
}

for (i=0;i<MAX;i++) {
 t=0;
 for (j=0;j<WSIZE;j+=2) {
 t+=x[i+j]*w[j];
 t+=x[i+j+1]*w[j+1];
 }
 y[i]=t;
}
```

Penn ESE5320 Fall 2023 -- DeHon

42

42

# Throughput Engineering (Bandwidth Engineering)

43

# Terminology Word

- Word
  - set of bits act on together in our computer system
    - E.g. In our 64b processors, work on 64b words
  - Often bits used to represent a value
    - 64b to represent a double-precision float or 64b integer
    - 32b to represent a single-precision float or 32b integer

44

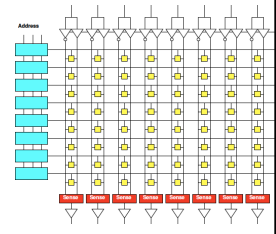
# Bandwidth Engineering

- High bandwidth is easier to engineer than low latency
  - Wide-word
  - Banking (not today → come back to later)
    - Decompose memory into independent banks
    - Route requests to appropriate bank

45

# Wide Memory

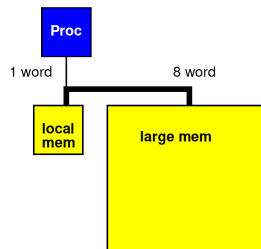
- Can make memory wider (less square)
  - Get more bits out per cycle
- As long as we share the address
  - One address to select wide word or bits
- Efficient if all read together



46

# Revisit Preclass 3

- Use wide memory access to move 8 words ( $x[]$ ) between large and small memory in a single cycle



47

# Preclass 3 + wide

- Impact of 8 word datapath between large and local memory?

$$\bullet \text{WSIZE} * \text{MAX} * (T_{\text{comp}} + 3 * T_{\text{local}})$$

$$\bullet + \text{WSIZE} * (T_w + T_x)$$

$$\bullet + \text{MAX} * T_x$$

$$\bullet 5 * 10^6 * (5 + 3)$$

$$\bullet + 5 * (20 + 20)$$

$$\bullet + 10^6 * 20$$

$$\bullet 6 * 10^7$$

- What term(s) impacted?

```

for (j=0; j<WSIZE; j++)
 local_w[j]=w[j];
for (j=0; j<WSIZE-1; j++)
 local_x[j+1]=x[j];
for (i=0; i<MAX; i++) {
 t=0;
 for (j=0; j<WSIZE-1; j++)
 local_x[j]=local_x[j+1];
 local_x[WSIZE-1]=x[WSIZE-1];
 for (j=0; j<WSIZE; j++)
 t+=local_x[j]*local_w[j];
 y[i]=t;
}

```

48



## Preclass 3 + wide

- $WSIZE * MAX * (T_{comp} + 3 * T_{local})$
  - $+ WSIZE * (T_w + T_x)$
  - $+ MAX / 8 * T_x$
  - $+ MAX * (T_{local} + 2 * T_{comp})$
  - $5 * 10^6 * (5 + 3)$
  - $+ 5 * (20 + 20)$
  - $+ 10^6 * 20 / 8$
  - $+ 10^6 * 3$
  - $4.55 * 10^7$
- ```

for (j=0; j<WSIZE; j++)
    local_w[j]=w[j];
for (j=0; j<WSIZE-1; j++)
    local_x[j+1]=x[j];
offset=0;
for (i=0; i<MAX; i++) {
    t=0;
    for (j=0; j<WSIZE-1; j++)
        local_x[j]=local_x[j+1];
    if (offset==0)
        local_x[WSIZE-1;WSIZE-1+8]=
            x[i+WSIZE-1;WSIZE-1+8]; // 8 words
    else
        local_x[WSIZE-1]=local_x[WSIZE-1+offset];
    offset=(offset+1)&(0x07); // mod 8
    for (j=0; j<WSIZE; j++)
        t+=local_x[j]*local_w[j];
    y[i]=t;
}
    
```
- Penn ESE5320 Fall 2023 -- DeHon

49

Amdahl's Law

Penn ESE5320 Fall 2023 -- DeHon

50

Amdahl's Law

- If you only speedup Y(%) of the code, the most you can accelerate your application is $1/(1-Y)$
- $T_{before} = 1 * Y + 1 * (1-Y) = 1$
- Speedup by factor of S
- $T_{after} = (1/S) * Y + 1 * (1-Y)$
- Limit $S \rightarrow \infty$ $T_{before} / T_{after} = 1/(1-Y)$

Penn ESE5320 Fall 2023 -- DeHon

51

51

Amdahl's Law

- $T_{before} = 1 * Y + 1 * (1-Y) = 1$
- Speedup by factor of S
- $T_{after} = (1/S) * Y + 1 * (1-Y)$
- $Y = 70\%$
 - Possible speedup ($S \rightarrow \infty$) ?
 - Speedup if $S = 10$?

Penn ESE5320 Fall 2023 -- DeHon

52

52

Amdahl's Law

- If you only speedup Y(%) of the code, the most you can accelerate your application is $1/(1-Y)$
- Implications
 - Amdahl: good to have a fast sequential processor
 - Keep optimizing
 - $T_{after} = (1/S) * Y + 1 * (1-Y)$
 - For large S, bottleneck now in the 1-Y

Penn ESE5320 Fall 2023 -- DeHon

53

53

Memory Organization

- Architecture contains
 - Large memories
 - For density, necessary sharing
 - Small memories local to compute
 - For high bandwidth, low latency, low energy
- Need to move data
 - Among memories
 - Large to small and back
 - Among small

Penn ESE5320 Fall 2023 -- DeHon

54

54

Big Ideas

- Memory bandwidth and latency can be bottlenecks
- Exploit small, local memories
 - Easy bandwidth, low latency, energy
- Exploit data reuse
 - Keep in small memories
- Minimize data movement
 - Small, local memories keep distance short
 - Minimally move into small memories

Penn ESE5320 Fall 2023 -- DeHon

55

55

Admin

- Reading for Wednesday on canvas
- If never seen a Lego
 - Plan to watch short (<3min) video before preclass for Wednesday
- Board distribution on Wednesday
 - 10:05am to lecture start

- HW2 due Friday

Penn ESE5320 Fall 2023 -- DeHon

56

56