

# ESE5320: System-on-a-Chip Architecture

Day 6: September 21, 2022  
Data-Level Parallelism

Masks  
Preclass



Penn ESE5320 Fall 2022 -- DeHon

1

## Today

### Data-level Parallelism

- For Parallel Decomposition (Part 1)
- Architectures (Part 2)
- Concepts (Part 3)
- NEON

Penn ESE5320 Fall 2022 -- DeHon

2

2

## Message

- Data Parallelism easy basis for decomposition
  - When things are data parallel, exploiting for parallelism relatively easy
- Data Parallel architectures can be compact
  - pack more computations onto a fixed-size Integrated Circuit chip
  - OR perform computation in less area

Penn ESE5320 Fall 2022 -- DeHon

3

3

## Preclass 1

- 500 news articles
  - 3000 words each
- Count total occurrences of a string
- How can we exploit data-level parallelism on task?
- How much parallelism can we exploit?
  - If don't exploit parallelism within article?
  - If do exploit parallelism within article?

Penn ESE5320 Fall 2022 -- DeHon

4

4

## Parallel Decomposition

Penn ESE5320 Fall 2022 -- DeHon

5

5

## Data Parallel

- Data-level parallelism can serve as an organizing principle for parallel task decomposition
- Run computation on independent data in parallel

Penn ESE5320 Fall 2022 -- DeHon

6

6

## Exploit

- Can exploit with
  - Threads
  - Pipeline Parallelism
  - Instruction-level Parallelism
  - Fine-grained Data-Level Parallelism

Penn ESE5320 Fall 2022 -- DeHon

7

7

## Performance Benefit

- Ideally linear in number of processors (resources)
- Resource Bound:
  - $T_{dp} = (T_{single} \times N_{data}) / P$
- $T_{single}$  = Latency on single data item
- $T_{dp} = T_{single} \times [N_{data} / P]$

Penn ESE5320 Fall 2022 -- DeHon

8

8

## Preclass 2 Common Examples

- What are common examples of DLP?
  - Simulation
  - Numerical Linear Algebra
  - Signal or Image Processing
  - Optimization

Penn ESE5320 Fall 2022 -- DeHon

9

9

## Hardware Architectures

Part 2

Penn ESE5320 Fall 2022 -- DeHon

10

10

## Idea

- If we're going to perform the same operations on different data, exploit that to reduce area, energy
- Reduced area means can have more computation on a fixed-size IC chip.

Penn ESE5320 Fall 2022 -- DeHon

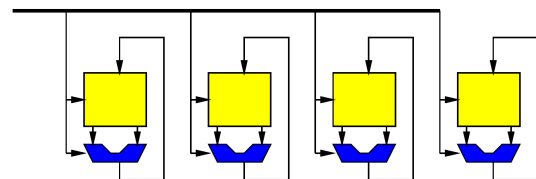
11

11

## SIMD

- Single Instruction Multiple Data

Shared Instruction



Penn ESE5320 Fall 2022 -- DeHon

12

12

## Ripple Carry Addition

- Can define logic for each bit, then assemble:
  - bit slice

Penn ESE5320 Fall 2022 -- DeHon

13

## Arithmetic Logic Unit (ALU)

- Observe:
  - with small tweaks can get many functions with basic adder components

Penn ESE5320 Fall 2022 -- DeHon

14

## Arithmetic and Logic Unit

ALU

Penn ESE5320 Fall 2022 -- DeHon

15

## ALU Functions

- A+B w/ Carry
- B-A
- A xor B (squash carry)
- A\*B (squash carry)
- /A

**Key observation:** every ALU bit does the same thing on different bits of the data word(s).

Penn ESE5320 Fall 2022 -- DeHon

16

## ARM v7 Core

- ALU is Key compute operator in a processor

Penn ESE5320 Fall 2022 -- DeHon

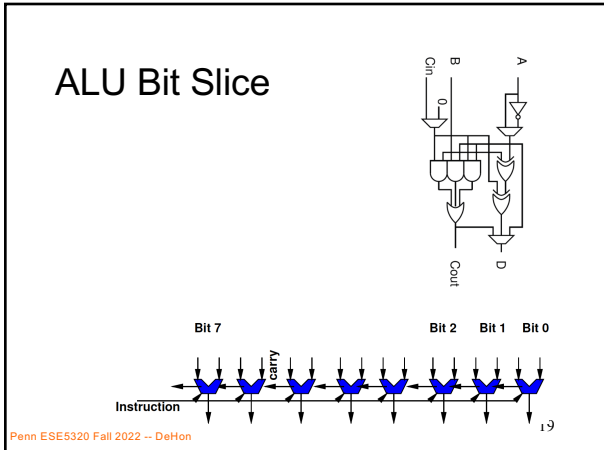
17

## W-bit ALU as SIMD

- Familiar idea
- A W-bit ALU (W=8, 16, 32, 64, ...) is SIMD
- Each bit of ALU works on separate bits
  - Performing the same operation on it
    - Trivial to see bitwise AND, OR, XOR
    - Also true for ADD (each bit performing Full Adder)
- Share one instruction across all ALU bits

Penn ESE5320 Fall 2022 -- DeHon

18



19

### Preclass 4

- What do we get when add 65280 to 257
  - 32b unsigned add?
  - 16b unsigned add?

Penn ESE5320 Fall 2022 -- DeHon

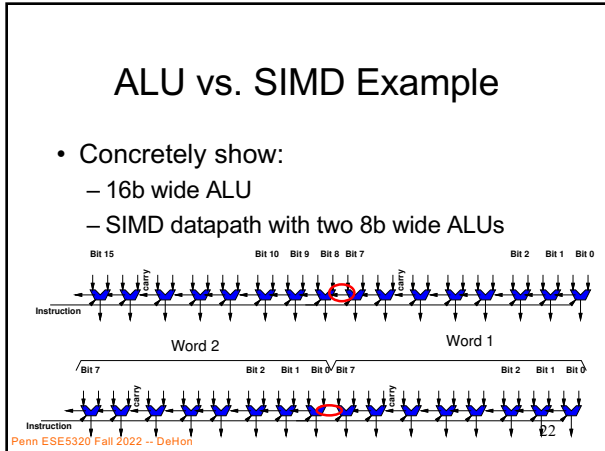
20

### ALU vs. SIMD ?

- What's different between
  - 128b wide ALU
  - SIMD datapath supporting eight 16b ALU operations

Penn ESE5320 Fall 2022 -- DeHon

21



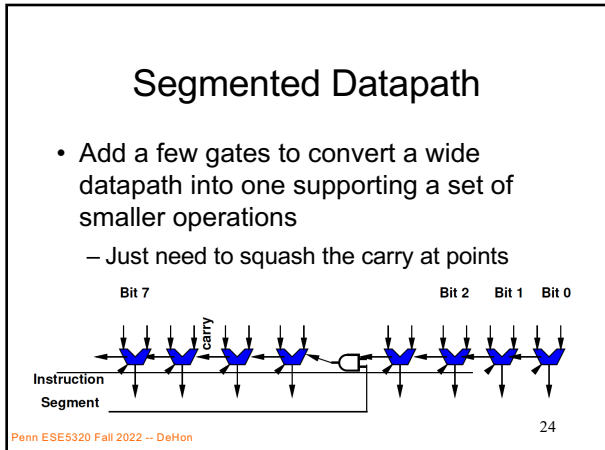
22

### ALU vs. SIMD ?

- How could we get both operations from the **same** hardware?
  - 128b wide ALU
  - SIMD datapath supporting eight 16b ALU operations

Penn ESE5320 Fall 2022 -- DeHon

23



24

## Segmented Datapath

- Add a few gates to convert a wide datapath into one supporting a set of smaller operations
  - Just need to squash the carry at points
- But need to keep instructions (description) small
  - So typically have limited, homogeneous widths supported

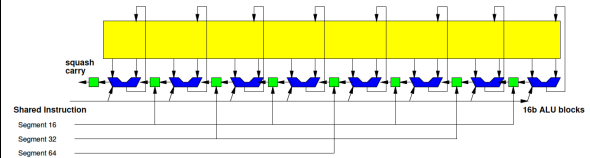
Penn ESE5320 Fall 2022 -- DeHon

25

25

## Segmented 128b Datapath

- 1x128b, 2x64b, 4x32b, 8x16b



Penn ESE5320 Fall 2022 -- DeHon

26

26

## Vector

- In Math – 1D array of numbers
- Here – 1D array of numbers that we operate upon with SIMD operations
  - Perform the same operations on
- Vector A = [2, 3, 5, 7]
- Vector B = [4, 6, 8, 9]
- A+B = [6, 9, 13, 16]

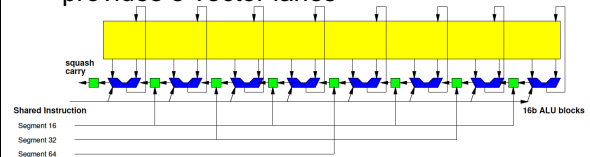
Penn ESE5320 Fall 2022 -- DeHon

27

27

## Terminology: Vector Lane

- Each of the separate segments called a **Vector Lane**
  - Length of vector hardware natively supports
- For 16b data on 128b datapath, this provides 8 vector lanes



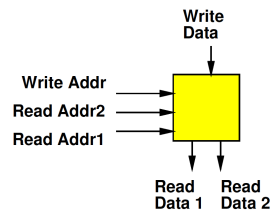
Penn ESE5320 Fall 2022 -- DeHon

28

28

## Register File

- Small Memory
- Usually with multiple ports
  - Ability to perform multiple reads and writes simultaneously
- Small
  - To make it fast (small memories fast)
  - Multiple ports are expensive



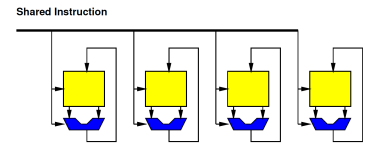
Penn ESE5320 Fall 2022 -- DeHon

29

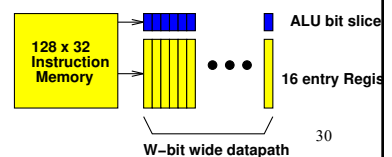
29

## SIMD Datapath

- Logical View:



- Layout View:



Penn ESE5320 Fall 2022 -- DeHon

30

30

## Preclass 5

- Look at Area of datapath
- Imagine filling a die with copies(instances) of datapath
  - If datapath smaller, can fit more of them into fixed die area
- Look at computation this provides

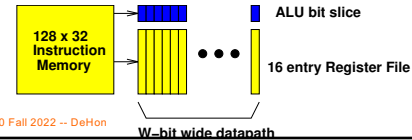
Penn ESE5320 Fall 2022 -- DeHon

31

31

## Preclass 5

W	A(W)	% datapath	Instances	Peak 16b ops
16				
64				
128				
256				



Penn ESE5320 Fall 2022 -- DeHon

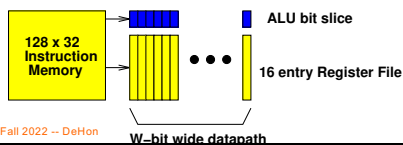
32

32

## Preclass 5

W	A(W)	% datapath	Instances	Peak 16b ops	Ratio
16	1.1	9	91	91	1.0
64	1.8	43	56	448	4.9
256					

Complete: 64, 256

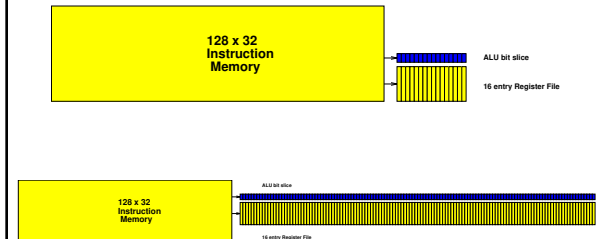


Penn ESE5320 Fall 2022 -- DeHon

33

33

## To Scale Comparison

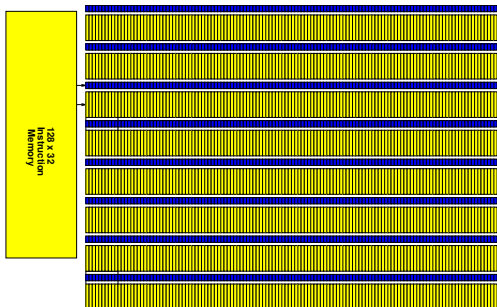


Penn ESE5320 Fall 2022 -- DeHon

34

34

## To Scale W=1024



Penn ESE5320 Fall 2022 -- DeHon

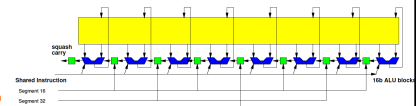
35

35

## Preclass 6: Vector Length

- May not match physical hardware length
  - Logical (application need): Vector Length
  - Physical (hardware provide): Vector Lanes

```
vadd(int *a, int *b, int *res,
     int vector_length) {
    for (int i=0; i<vector_length; i++)
        res[i]=a[i]+b[i];
}
```



Penn ESE5320 Fall 2022 -- DeHon

36

## Preclass 6: Vector Length

- May not match physical hardware length
  - Logical (application need): Vector Length
  - Physical (hardware provide): Vector Lanes
- How support
  - Vector length < Vector Lanes? (5, 8)
  - Vector length > Vector Lanes? (24, 8)

Penn ESE5320 Fall 2022 -- DeHon

37

37

## Preclass 6: Vector Length

- May not match physical hardware length
  - Logical (application need): Vector Length
  - Physical (hardware provide): Vector Lanes
- Efficiency (utilization) when
  - Vector length < Vector Lanes? (5, 8)
  - Vector length > Vector Lanes? (24, 8)
  - Vector length % (Vector Lanes) != 0 (13, 8)
    - E.g. vector length 13, for 8 vector lanes

Penn ESE5320 Fall 2022 -- DeHon

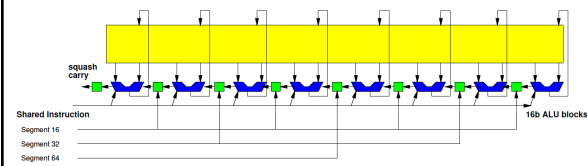
38

38

## Performance

- Resource Bound

$$T_{\text{vector}} = \lceil N_{\text{op}}/VL \rceil \times T_{\text{cycle}}$$



Penn ESE5320 Fall 2022 -- DeHon

39

39

## Preclass 3: Opportunity

- Don't need 64b variables for lots of things
- Natural data sizes?
  - Audio samples?
  - Input from A/D?
  - Video Pixels?
  - X, Y coordinates for 4K x 4K image?

Penn ESE5320 Fall 2022 -- DeHon

40

40

## Vector Computation

- Easy to map to SIMD flow if can express computation as operation on vectors
  - Vector Add
  - Vector Multiply
  - Dot Product

Penn ESE5320 Fall 2022 -- DeHon

41

41

## Concepts

Part 3

Penn ESE5320 Fall 2022 -- DeHon

42

42

## Terminology: Scalar

- Simple: non-vector
- When we have a vector unit controlled by a normal (non-vector) processor core often need to distinguish:
  - Vector operations that are performed on the vector unit
  - Normal=non-vector=scalar operations performed on the base processor core

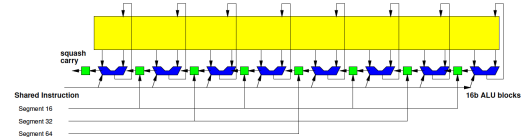
Penn ESE5320 Fall 2022 -- DeHon

43

43

## Vector Register File

- Need to be able to feed the SIMD compute units
  - Not be bottlenecked on data movement to the SIMD ALU
- Wide RF (register file) to supply
- With wide path to memory

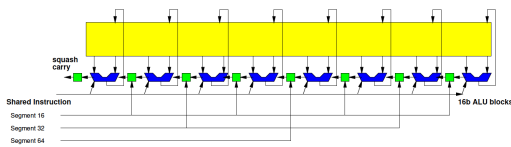


Penn ESE5320 Fall 2022 -- DeHon

44

## Point-wise Vector Operations

- Easy – just like wide-word operations (now with segmentation)



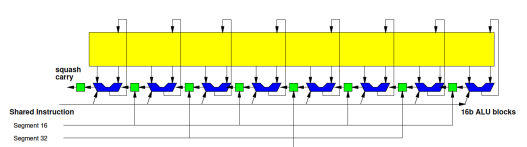
Penn ESE5320 Fall 2022 -- DeHon

45

45

## Point-wise Vector Operations

- ...but alignment matters.
- If not aligned, need to perform data movement operations to get aligned



Penn ESE5320 Fall 2022 -- DeHon

46

46

## Ideal

- for ( $i=0; i<64; i=i++$ )
  - $c[i]=a[i]+b[i]$
- No data dependencies
- Access every element
- Number of operations is a multiple of number of vector lanes

Penn ESE5320 Fall 2022 -- DeHon

47

47

## Skipping Elements?

- How does this work with datapath?
  - Assume loaded  $a[0], a[1], \dots, a[63]$  and  $b[0], b[1], \dots, b[63]$  into vector register file
- for ( $i=0; i<64; i=i+2$ )
  - $c[i/2]=a[i]+b[i]$

Penn ESE5320 Fall 2022 -- DeHon

48

48



## Stride

- Stride: the distance between vector elements used
- for  $(i=0; i<64; i=i+2)$ 
  - $c[i/2]=a[i]+b[i]$
- Accessing data with stride=2

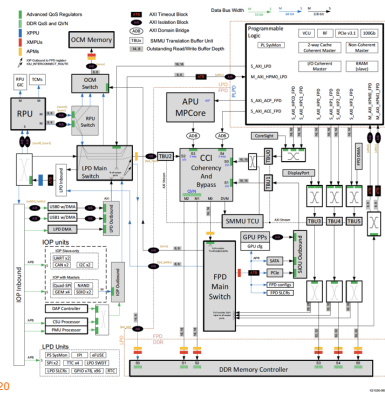
## Load/Store

- Strided load/stores
  - Some architectures will provide strided memory access that compact when read into register file
- Scatter/gather
  - Some architectures will provide memory operations to grab data from different places to construct a dense vector

## Neon

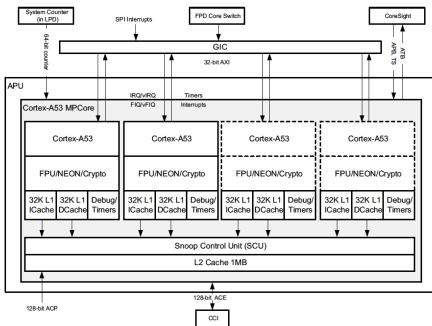
ARM Vector Accelerator

## Programmable SoC



UG1085  
Xilinx  
UltraScale  
Zynq  
TRM  
(p27)

## APU MPcore



UG1085  
Xilinx  
UltraScale  
Zynq  
TRM  
(p53)

## Neon Vector

- 128b wide register file, 16 registers
- Support
  - 2x64b
  - 4x32b (also Single-Precision Float)
  - 8x16b
  - 16x8b

## Sample Instructions

- VADD – basic vector
- VCEQ – compare equal
  - Sets to all 0s or 1s, useful for masking
- VMIN – avoid using if's
- VMLA – accumulating multiply
- VPADAL – maybe useful for reduce
  - Vector pair-wise add
- VEXT – for “shifting” vector alignment
- VLDn – deinterleaving load

Penn ESE5320 Fall 2022 -- DeHon

55

55

## ARM Cortex A53 (Ultra96) (similar to A-7 Pipeline)

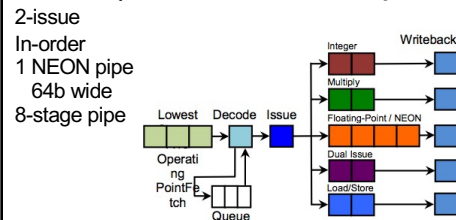


Figure 1 Cortex-A7 Pipeline

<https://www.anandtech.com/show/8718/the-samsung-galaxy-note-4-exynos-review/3>

<https://arstechnica.com/gadgets/2011/10/arms-new-cortex-a7-is-tailor-made-for-android-superphones/>

Penn ESE5320 Fall 2022 -- DeHon

56

56

## Big Ideas

- Data Parallelism easy basis for decomposition
- Data Parallel architectures can be compact – pack more computations onto a chip
  - SIMD, Pipelined
  - Benefit by sharing (instructions)
  - Performance can be brittle
    - Drop from peak as mismatch

Penn ESE5320 Fall 2022 -- DeHon

57

57

## Admin

- Remember feedback
- Reading for Day 7 online
- HW3 due Friday
- HW4 out
  - Including partner assignments
    - Board holders – contact partner soon

Penn ESE5320 Fall 2022 -- DeHon

58

58