ESE532: System-on-a-Chip Architecture

Day 6: September 20, 2017
Data-Level Parallelism

Today

Data-level Parallelism
- For Parallel Decomposition
- Architectures
- Concepts
- NEON

Message

- Data Parallelism easy basis for decomposition
- Data Parallel architectures can be compact – pack more computations onto a die

Preclass 1

- 400 news articles
- Count total occurrences of a string
- How can we exploit data-level parallelism on task?
- How much parallelism can we exploit?

Parallel Decomposition

Data Parallel

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

Exploit

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

Thread Exploit DP

- How exploit threads for data-parallel text search?

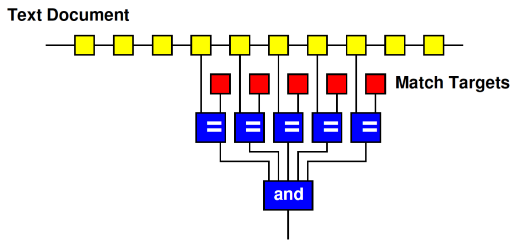
SPMD

Single Program Multiple Data
- Only need to write code once
- Get to use many times

Pipeline Exploit

- How exploit hardware pipeline for text search?

Pipeline Text Search



Common Examples

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

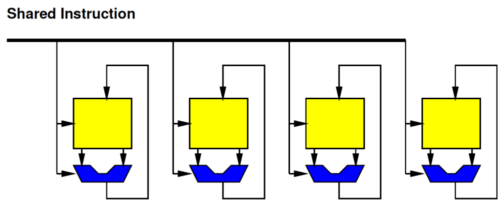
Hardware Architectures

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 die.

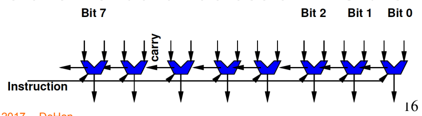
SIMD

- Single Instruction Multiple Data

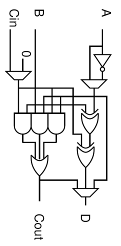


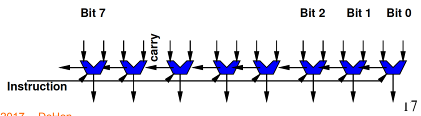
W-bit ALU as SIMD

- 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



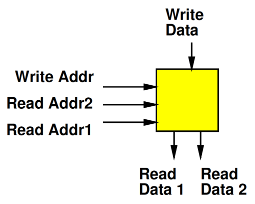
ALU Bit Slice





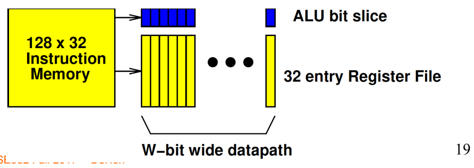
Register File

- 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



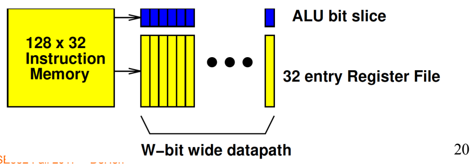
Preclass 2

- Area W=16?
- Area W=128?
- Number in 10^8
  - W=16
  - W=128
- Perfect Pack Ratio?



Preclass 2

- W for single datapath in 10^8?
- Perfect 16b pack ratio?
- Compare W=128 perfect pack ratio?

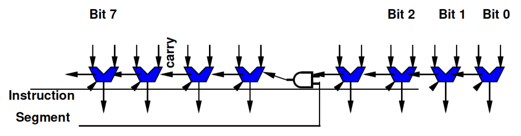


ALU vs. SIMD ?

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

Segmented Datapath

- Relatively easy (few additional gates) to convert a wide datapath into one supporting a set of smaller operations
  - Just need to squash the carry at points

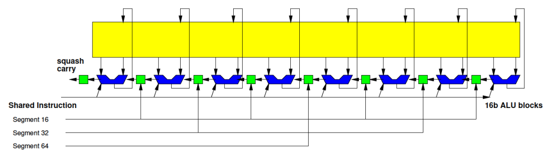


Segmented Datapath

- Relatively easy (few additional 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

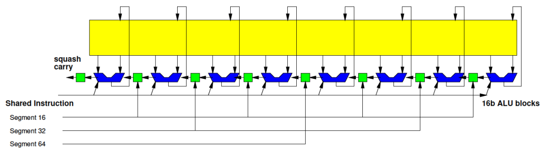
Segmented 128b Datapath

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



Terminology: Vector Lane

- Each of the separate segments called a **Vector Lane**
- For 16b data, this provides 8 vector lanes



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?

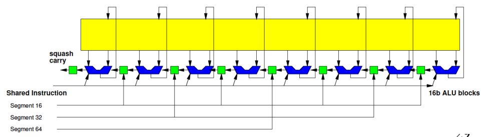
Vector Computation

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

Concepts

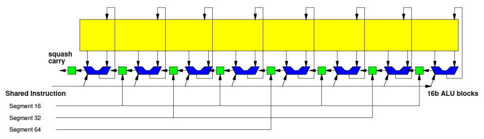
Vector Register File

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



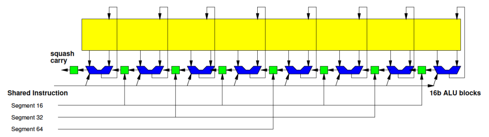
Point-wise Vector Operations

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



Point-wise Vector Operations

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



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

Vector Length

- May not match physical hardware length
- What happens when
  - Vector length > hardware SIMD operators?
  - Vector length < hardware SIMD operators?
  - Vector length % hdw operators !=0
    - E.g. vector length 20, for 8 hdw operators

Skipping Elements?

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

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

Dot Product

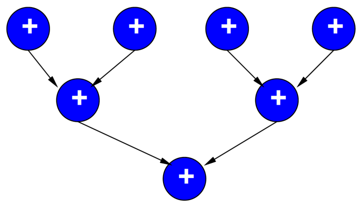
- What happens when need a dot product?
- res=0;
- for (i=0;i<N;i++)
  - res+=a[i]*b[i]

Reduction

- Common operations where want to perform a combining operation to reduce a vector to a scalar
  - Sum values in vector
  - AND, OR
- Reduce Operation

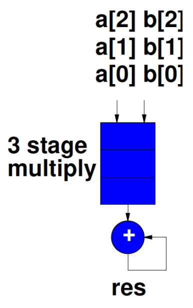
Reduce Tree

- Efficiently handled with reduce tree



Reduce in Pipeline

- Comes almost for free in pipeline



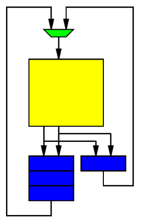
Vector Reduce Instruction

- Usually include support for vector reduce operation
  - Doesn't need to add much to delay
  - Maybe even faster than performing larger operation
    - 8 16x16 multiplies with sum reduce less complex than one 128x128 multiply
    - …can exploit datapath of larger operation

Dot Product Revisited

- for (i=0;i<N;i++)
  - res+=a[i]*b[i]
- a in R0—R4
- b in R4—R7

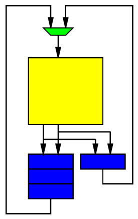
- With 3 cycle pipelined multiply
- What happens if try to implement dot product as:
  - MPY R0, R4, R14
  - ADD R14, R15, R15
  - MPY R1, R5, R14
  - ADD R14, R15, R15
  - …



Dot Product Revisited

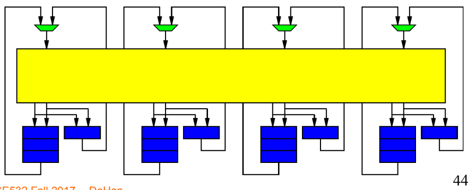
- How should order (reformulate) instructions exploiting data-level parallelism?

- for (i=0;i<N;i++)
  - res+=a[i]*b[i]
- a in R0—R4
- b in R4—R7



Pipelined Vector Units

- Will get both pipelining and parallel vector lanes
- Exploit data-level parallelism for both

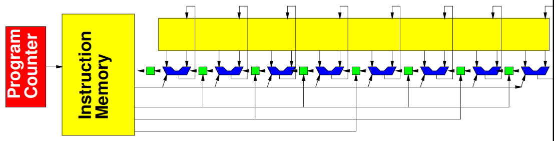


Conditionals?

- What happens if want to do something different?
- For (i=0;i<8;i++)
  - if (a[i]<b[i])
    - d[i]=a[i]+c[i]
  - else
    - d[i]=b[i]+c[i]

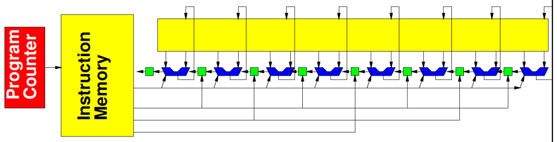
Conditionals

- Only have one Program Counter
  - Cannot implement conditional via branching



Conditionals

- Only have one instruction
  - Cannot perform separate operations on each ALU in datapath



Conditionals

- Only have one Program Counter
  - Cannot implement conditional via branching
- Only have one instruction
  - Cannot perform separate operations on each ALU in datapath
- Must perform an invariant operation sequence
- Simple answer: prevent using SIMD unit
- Better: predicated execution

Predicated Operation

- Many architectures will provide a predicated operation
- Only perform operation when predicate matches instruction

- p[i]=a[i]<b[i]
- p[i]:   d[i]=c[i] + a[i]
- ~p[i]: d[i]=c[i] + b[i]

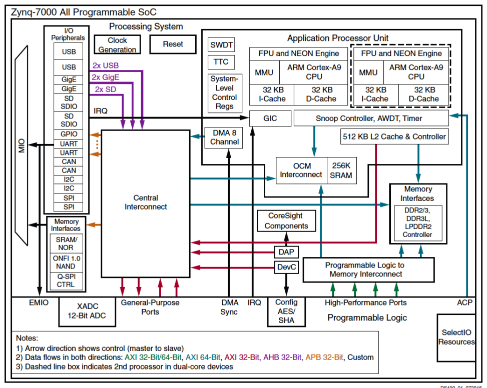
Predicated Operation

- What does this do to instructions must be issued?
- What does this do to efficiency?
  - Useful operations performed per cycle

- p[i]=a[i]<b[i]
- p[i]:   d[i]=c[i] + a[i]
- ~p[i]: d[i]=c[i] + b[i]

Neon

ARM Vector Accelerator on Zynq



Notes:
1) Arrow direction shows control (master to slave)
2) Data flows in both directions: AXI 32-Bit/64-Bit, AXI 64-Bit, AXI 32-Bit, AHB 32-Bit, APB 32-Bit, Custom
3) Dashed line box indicates 2nd processor in dual-core devices

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

Neon Notes

- Didn't see
  - Vector-wide reduce operation
  - Conditionals within vector lanes
- Do need to think about operations being pipelined within lanes

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

Admin

- No reading for day 7
- HW3 due Friday
- HW4 out