ESE532: System-on-a-Chip Architecture

Day 18: Nov. 3, 2021
Hash Tables
Design Space

Penn

Today

- Software Hash Tables (Part 1)
- Hardware (FPGA) Hash Maps (Part 2)
- Design-Space Exploration
  - Generic (Part 3)
  - Concrete (Part 4): Fast Fourier Transform (FFT)
    - Time permitting

Message

- Hash tables are useful tools
- The universe of possible implementations (design space) is large
  - Many dimensions to explore
- Formulate carefully
- Approach systematically
- Use modeling along the way for guidance

4K Chunk LZW Search Story so far….

	BRAMs	Operations
Brute Search	1	4K
Tree with Dense RAM	512	1
Tree with Full Assoc	175	1
Tree with Tree	14	12

36Kb BRAMs on ZU3EG = 216

Hash Tables

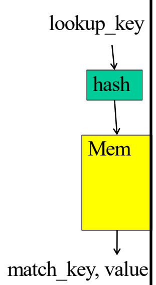
Part 1

High Performance Map

- Would prefer not to search
- Want to do better than $\log_2(N)$ time
  - $\log_2(N)$ achieved with tree
- Direct lookup in arrays (memory) is good…

Hash Table

- Attempt to turn into direct lookup
- Compute some function of key
  - A hash
- Perform lookup at that point
- If hash maps a single entry (or no entry)
  - Great, got direct lookup
    - Like sparse table case



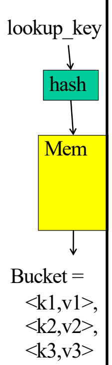
Miss = (match_key!=lookup_key)

Preclass 1

- Average number of entries per hash when N > HASH_CAPACITY?
  - Concrete example
    - N= 4096
    - HASH_CAPACITY=256

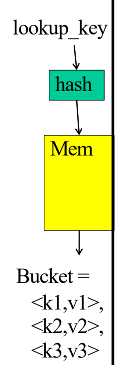
Hash Table

- Attempt to turn into direct lookup
- Compute some function of key
  - A hash
- Perform lookup at that point
- Typically, prepared for several keys to map to same hash → call it a bucket
  - Keep list or tree of things in each bucket



Hash Table

- Compute some function of key
  - A hash
- Perform lookup at that point
- Find bucket with small number of entries
  - Searching that bucket easier
  - …but no absolute guarantee on maximum bucket size

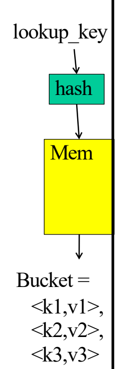


Hardware Hash Tables

Part 2

Hardware Hash

- Want to avoid variable size buckets
  - So can read in one lookup
    - Can make wider for some fixed number of slots
  - So can resolve in one cycle



Hash Size Distribution

- Look at what the distribution looks like for number of entries
- N – number of entries
- C – HASH_CAPACITY
- m – number of items in a slot
- Compute distribution for each bucket size

Preclass 2

N=1024

m→	0	1	2	3	4+
C=1024	0.37				
C=2048					
C=4096					

$$\binom{N}{m}\left(\frac{1}{C}\right)^{m}\left(1-\frac{1}{C}\right)^{N-m}$$

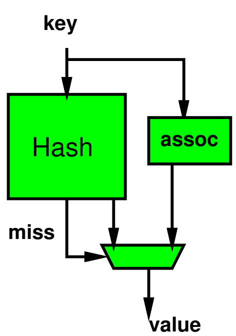
Hash

- Can tune hash parameters to control distribution
- Spend more memory → smaller buckets → less work finding things in buckets
  - Memory-Time tradeoff
- Still have possibility of large buckets
  - …but probability is low

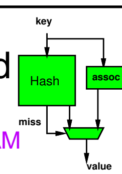
Idea

- Hash mostly works
- Engineer hash to hold most cases
  - Combination of
    - sparcity (entries>N)
    - Hold multiple entries per hash value
- Few cases that overflow
  - Store in small fully associative memory

Hybrid Hash+Assoc.



LZW 4K Chunk Hybrid



- 72 entry assoc. match
  - needs 3 match BRAMs + 1 data BRAM
  - Associative match 20b key
  - 72 entries (72/4096=1.7% for 4096)
- So, can hold ~1% conflicts in 4K hash
- Hash N=4096, C=16384, m=2, store 2
  - Prob 3+: <1% (see table 1024, 4096)
  - 20b key+12b value=4B per entry
  - 16384*2*4B=4*2*4 BRAMs
- 32+4=36 BRAMs

Further Optimization

- Previous example illustrative
  - Not necessarily optimal (explore parameters)
    - Expect not optimal
- May be able to do better with multiple hashes
  - See Dhawan reading paper
  - May need to use that design in hybrid configuration with assoc. memory like previous example

Allow Imperfect?

- **Question:** impact on compression if cannot store a few tree entries?
- Some encodings will find shorter matches than optimal
- **Q:** Impact on compression rate as a function of conflict rate?
- How compare to compression rate impact of chunk size?
  - Larger chunk with conflict rate vs. smaller chunk with smaller (or no) conflict rate
- → another tradeoff to explore

Hash Complexity

- Want to compute these lookup hashes for hardware fast
  - In a single cycle to keep II down for LZW
  - Can xor-together a set of bits quickly in hardware
    - Any 6-bits for one output bit in a single 6-LUT
    - Means capacity must be power-of-2

4K Chunk LZW Search

	BRAMs	Operations
Brute Search	1	4K
Tree with Dense RAM	512	1
Tree with Full Assoc	175	1
Tree with Tree	14	12
Tree with Hybrid	36	1

36Kb BRAMs on ZU3EG = 216

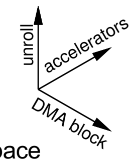
Part 3
Design-Space Exploration

Generic

Design Space

- Have many choices for implementation
  - Alternatives to try
  - Parameters to tune
  - Mapping options
- This is our freedom to impact implementation costs
  - Area, delay, energy

Design Space



- Ideally
  - Each choice orthogonal axis in high-dimensional space
  - Want to understand points in space
  - Find one that best meets constraints and goals
- Practice
  - Seldom completely orthogonal
  - Requires cleverness to identify dimensions
  - Messy, cannot fully explore
  - But…can understand, prioritize, guide

Preclass 3

- What choices (design-space axes) can we explore in mapping a task to an SoC?
- Hint: What showed up in homework so far?

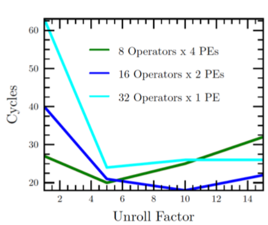
Design-Space Choices

- Type of parallelism
- How decompose / organize parallelism
- Area-time points (level exploited)
- What resources we provision for what parts of computation
- Where to map tasks
- How schedule/order computations
- How synchronize tasks
- How represent data
- Where place data; how manage and move
- What precision use in computations

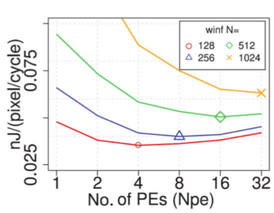
Generalize Continuum

- Encourage to think about parameters (axes) that capture continuum to explore
- Start from an idea
  - Maybe can compute with 8b values
  - Maybe can put matrix-mpy computation on FPGA fabric
  - Maybe 1 hash + 1 fully assoc.
  - Move data in 1KB chunks
- Identify general knob
  - Tune intermediate bits for computation
  - How much of computation go on FPGA fabric
  - How many hash/assoc levels?
  - What is optimal data transfer size?

Finding Optima



- Kapre, FPL 2009



- Kadric, TRETS 2016

Design Space Explore

- Think systematically about how might map the application
- Avoid overlooking options
- Understand tradeoffs
- The larger the design space
  →more opportunities to find good solutions
  Reduce bottlenecks

Elaborate Design Space

- Refine design space as you go
- Ideally identify up front
- Practice bottlenecks and challenges
  - will suggest new options / dimensions
    - If not initially expect memory bandwidth to be a bottleneck…
- Some options only make sense in particular sub-spaces
  - Bitwidth optimization not a big issue on the 64b processor
    - More interesting on vector, FPGA

Tools

- Sometimes tools will directly help you explore design space
  - Sometimes do it for you
    - Mimimize II
  - In your hands, make easy
    - Unrolling, pipelining, II
    - Array packing and partitioning
    - Some choices for data movement
    - DMA pipelining and transfer sizes
    - Some loop transforms
    - Granularity to place on FPGA
    - ap_fixed
    - Number of data parallel accelerators

Tools

- Often tools will not help you with design space options
  - Need to reshape functions and loops
  - Line buffers
  - Data representations and sizes
  - C-slow sharing
  - Communications overlap
  - Picking hash function parameters

Code for Exploration

- Can you write your code with parameters (#define) that can easily change to explore continuum?
  - Unroll factor?
  - Number of parallel tasks?
  - Size of data to move?
- Want to make it easy to explore different points in space

Part 4
Design-Space Exploration

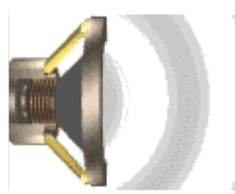
Example FFT

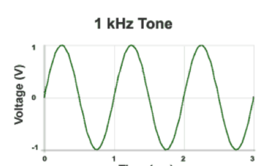
Skip Wrapup

Sound Waves

Hz = 1/s

1kHz = 1000 cycles/s

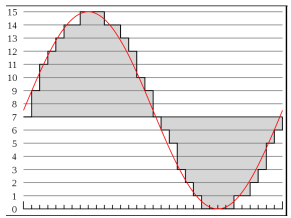




Source: http://www.mediacollege.com/audio/01/sound-waves.html

Discrete Sampling

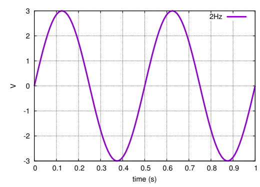
- Represent as time sequence
- Discretely sample in time
- What we can do directly with an Analog-to-Digital (A2D) converter



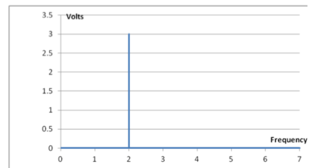
http://en.wikipedia.org/wiki/File:Pcm.svg

Time-Domain & Frequency-domain

- **example…have a pure tone**
  - If period: T = 1/2 and Amplitude = 3 Volts
  - $s(t) = A\,sin(2\pi f t) = A\,sin(2\pi 2 t)$



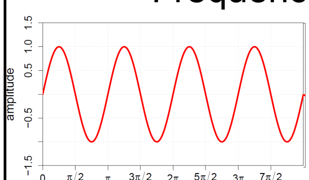
Time domain representation

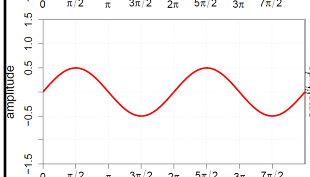


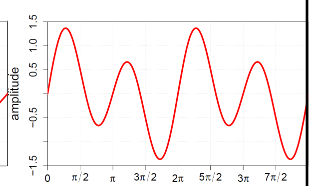
Frequency domain representation

Frequency-domain

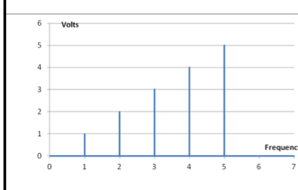
- Can represent sound wave as linear sum of frequencies

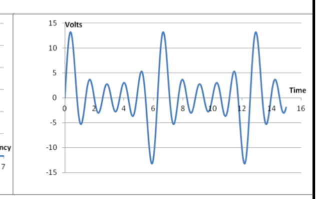




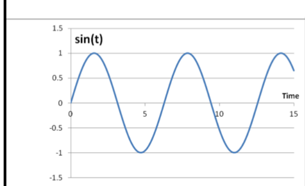


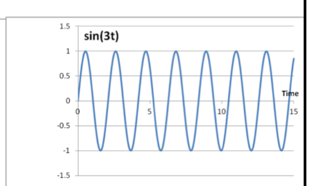
Time vs. Frequency

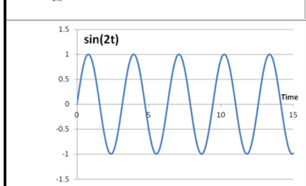




Fourier Series







- The cos(nx) and sin(nx) functions form an orthogonal basis: they allow us to represent any periodic signal by taking a linear combination of the basis components without interfering with one another

Fourier Transform

- Identify spectral components (frequencies)
- Convert between Time-domain to Frequency-domain
  - E.g. tones from data samples
  - Central to audio coding – e.g. MP3 audio

$$Y[k] = \sum_{j=0}^{n-1} \left(X[j] e^{-2i\pi \frac{k}{n}} \right)$$

FT as Matching

- Fourier Transform is essentially performing a dot product with a frequency
  - How much like a sine wave of freq. f is this?

$$Y[k] = \sum_{j=0}^{n-1} \left(X[j] e^{-2i\pi \frac{k}{n}} \right)$$

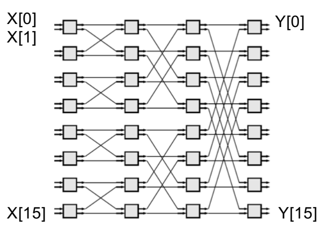
Fast-Fourier Transform (FFT)

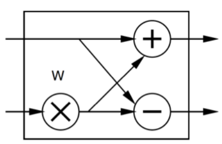
- Efficient way to compute FT
- O(N*log(N)) computation
- Contrast N^2 for direct computation
  - N dot products
    - Each dot product has N points (multiply-adds)

$$Y[k] = \sum_{j=0}^{n-1} \left(X[j] e^{-2i\pi \frac{k}{n}} \right)$$

FFT

- Large space of FFTs
- Radix-2 FFT Butterfly



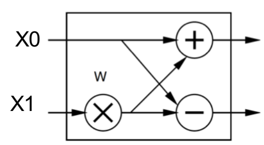


w – constant that
Differs for each instance
-- ``twiddle'' factor

$$Y[k] = \sum_{j=0}^{n-1} \left(X[j] e^{-2i\pi \frac{k}{n}} \right)$$

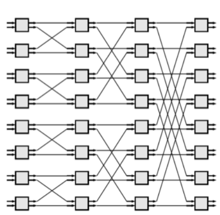
Basic FFT Butterfly

- Y0=X0+W(stage,butterfly)*X1
- Y1=X0-W(stage,butterfly)*X1
- Common sub expression, compute once: W(stage,butterfly)*X1

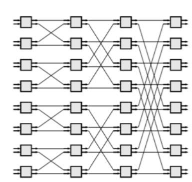


Preclass 4

- What parallelism options exist?
  - Single FFT
  - Sequence of FFTs

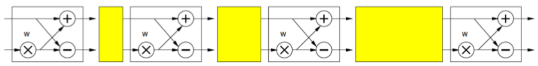


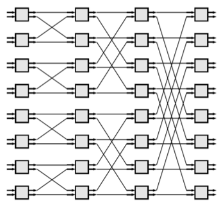
FFT Parallelism



- Spatial
- Pipeline
- Streaming
- By column
  - Choose how many Butterflies to serialize on a PE
- By subgraph
- Pipeline subgraphs

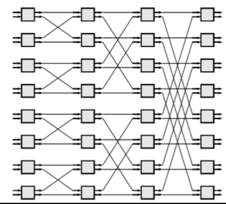
Streaming FFT





Preclass 5

- How large of a spatial FFT can implement with 360 multipliers?
  - 1 multiply per butterfly
  - (N/2) $\log_2(N)$ butterflies

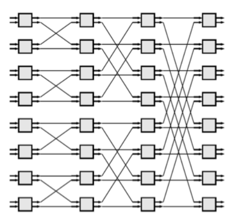


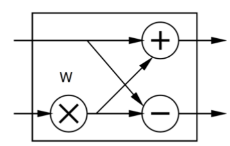
Bit Serial

- Could compute the add/multiply bit serially
  - One full adder per adder
  - W full adders per multiply
  - W=16, maybe 20—30 LUTs
  - 70,000 LUTs
    - ~= 70,000/30 ~= 2330 butterflies
      - 512-point FFT has 2304 butterflies
- Another dimension to design space:
  - How much serialize word-wide operators
  - Use LUTs vs. DSPs

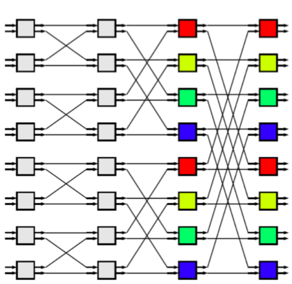
Accelerator Building Blocks

- What common subgraphs exist in the FFT?



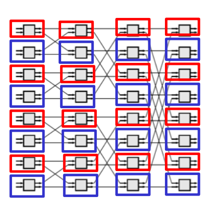


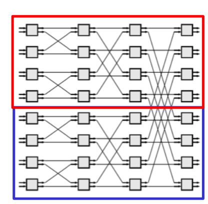
Common Subgraphs



Processor Mapping

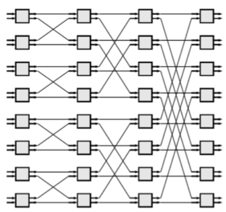
- How map butterfly operations to processors?
  - Implications for communications?





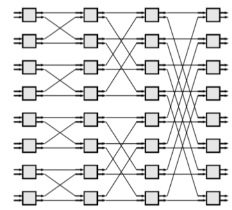
Storage

- How large local memory to communicate from stage to stage?



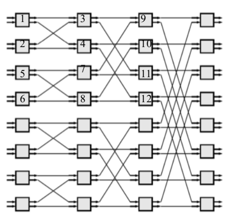
Evaluation Order

- How change evaluation order to reduce local storage memory?

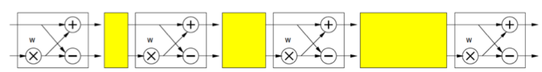


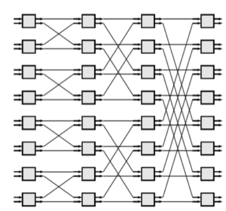
Preclass

- Evaluation order



Streaming FFT





Communication

- How implement the data shuffle between processors or accelerators?
  - Memories / interconnect ?
  - How serial / parallel ?
  - Network?

Data Precision

- Input data from A2D likely 12b
- Output data, may only want 16b
- What should internal precision and representation be?

Number Representation

- Floating-Point
  - IEEE standard single (32b), double (64b)
    - With mantissa and exponent
    - …half, quad ….
- Fixed-Point
  - Select total bits and fraction
    - E.g. 16.8 (16 total bits, 8 of which are fraction)
      - Represent 1/256 to 256-1/256
  - A(mpy) ~ W^2, A(add) ~ W

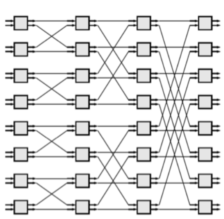
Operator Sizes

Operator	LUTs	LUTs + DSPs
Double FP Add	712	681+3 DSPs
Single FP Add	370	219+2 DSPs
Fixed-Point Add (32)	16	
Fixed-Point Add (n)	n/2	
Double FP Multiply	2229	223+10 DSPs
Single FP Multiply	511	461+3 DSPs
Fixed Multiply (32x32)	1099	
Fixed Multiply (16x16)	283	1 DSP
Fixed Multiply (18x25)		1 DSP
Fixed Multiply (n)	~ n^2	

FP (Floating Point) sizes from:
https://www.xilinx.com/support/documentation/ip_documentation/ru/floating-point.html

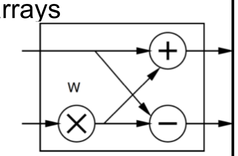
Heterogeneous Precision

- May not be same in every stage
  - W factors less than 1
  - Non-fraction grows at most 1b per stage



W Coefficients (w[stage][j])

- Precompute and store in arrays
- Compute as needed
  - How?
    - sin/cos hardware?
    - CORDIC?
    - Polynominal approximation?



- Specialize into computation
  - Many evaluate to 0, ±1, ±½, ….
  - Multiplication by 0, 1 not need multiplier…

FFT (partial) Design Space

- Parallelism
- Decompose
- Size/granularity of accelerator
  - Area-time
- Sequence/share
- Communicate
- Representation/precisions
- Twiddle

Big Ideas

- Near O(1) Map access → Hash Table
- Large design space for implementations
  - Including associative maps
- Worth elaborating and formulating systematically
  - Make sure don't miss opportunities
- Think about continuum for design axes
- Model effects for guidance and understanding

Admin

- Feedback
- Reading for Monday on web
- First project milestone due Friday
  - Including teaming
- P2 out