ESE532: System-on-a-Chip Architecture

Day 18: March 27, 2017

Representation and Precision

Today

- Fixed Point
- Errors from Limited Precision
- Precision Analysis / Interval Arithmetic
- Floating Point

Message

- We must always calculate with limited precision
- Precision costs area (and energy)
  - Can economize area (and energy) by judiciously using just the precision we need
  - Precision-cost tradeoff
- Can perform analysis on precision

Fixed Point

- Integer which interpret as a fraction
- Fixed-Point N.F
  - N bits
  - F bits below fraction (typically N<F)
  - Equivalently: meaning is Integer-value/2^F

$$A = \sum_{i=0}^{N-1} a_i 2^{i-F}$$

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 sizes from: https://www.xilinx.com/support/documentation/ip_documentation/ru/floating-point.html

Observe

- Floating-Point operators are large compared to Fixed-Point
  - For similar precision
    - 712 vs. 32 for addition
- Double-precision Floating point operators are large
  - 2229 Multiply, 712 Add
    - Can quickly fill 50,000 LUT programmable logic

Fixed-Point Economy

- Can fit more logic (more parallelism) using modest fixed-point
  - At 16b: Multiply 283, Add 16
  - Vs. Double: 2229, 712
- But
  - How much precision do we need?
  - How do we determine?

Perfect Representation

- Start with Fixed-Point 16.8
- What do we need to
  - represent the result of an addition? (3a)
  - represent the result of a multiplication? (3b)

Sequence

- Across a sequence of operations
  - A, B, C, D, E start Fixed-Point 16.8
- Y=((A+B)*C+D)*E;
- Perfect representation for partial results up to Y?

Looping: Bound loop

```
res=0;
for(i=0;i<4;i++)
    res=res*x+a[i]
```

- Assume a[i], x start Fixed-Point 16.8
- Final precision needed for res?

Looping: Unbound

```
res=0;
for(i=0;i<len;i++)
    res=res*x+a[i]
```

- Assume a[i], x start Fixed-Point 16.8, len starts Integer 16
- Final precision needed for res?

Perfect Representation

- Start with Fixed-Point 16.8
- What do we need to
  - represent the result of a division? (3e)
    - E.g. 00000001.00000000/00000011.00000000

Conclude

- Cannot generally keep perfect precision
- Will typically need to decide how much precision we need and where

Errors from Limited Precision

What error introduce?

- **Absolute Error** – what level of error do we have in approximated value or a result
- Might be all we care about
  - Get answer to 1mV accuracy
  - …or 1 pixel accuracy

- 4.13742 – assume full
- 4.1374 – 0.00002
  - 10^{-5}
- 4.137 – 0.00042
  - 10^{-4}
- 4.14 – 0.00256
  - 10^{-3}
- 4.1 – 0.03742
  - 10^{-2}

What error introduce?

- **Relative Error** – error as percentage of intended result
- May be more relevant, particularly if trying to identify small values

- 4.13742 – assume full
- 4.1374 – 4.8E-6
  - 10^{-6}
- 4.137 – 1.02E-4
  - 10^{-4}
- 4.14 – 6.2E-4
  - 10^{-4}
- 4.1 – 9.0E-3
  - 10^{-3}

Preclass 1

- Complete Table

Reduced Precision Calculation	Y	Y_1	Error (2 significant figures) Absolute $\|Y - Y_1\|$	Relative $\|(Y - Y_1)/Y\|$
$Y_1 = A_1 + B_1$				
$Y_1 = A_1 \times B_1$				
$Y_1 = A_1 / B_1$				
$Y_1 = A_1 / C_1$				
$Y_1 = A_1 / D_1$				

Observe

- Add/Multiply relatively well behaved
- Must be very careful when
  - Division involved
  - Divisors can be small
    - Get approximated near zero

Precision Allocation

- Full precision can be too expensive
  - Non-sensical
- Limited precision introduces errors
  - May be smaller than we care about
- Determine minimal precision needed
  - …or where to spend precision…

Empirical Analysis

- Make guess at precisions
- Set precisions in calculation
- Simulate on data
- Evaluate results (absolute, relative error) compared to gold standard
  - Unlimited precision…or, at least, higher precision
    - Often standard is double-precision float
      - …but, as we'll, even that's a compromise
- Update precision guess and repeat

Empirical Analysis

- Make guess at precisions
- Set precisions in calculation
- Simulate on data
- Evaluate results compared to gold standard

Care

- Adequate set of test data to trigger worst-case errors?
- Requires some understanding of calculation
- Shouldn't be entirely black box

Vivado HLS Support

- Has libraries to support
  - Arbitrary precision integers
  - Arbitrary precision fixed point
- For
  - Simulation
  - Synthesis
- UG902 – Vivado HLS User Guide
  - Chapter 2: Abritrary Precision Data Type Library

Precision Analysis

Precision Analysis

- Can analyze worst-case error impacts from limited precision
- Give results not sensitive to test set
- Give guidance on where to allocate precision
- …can be automated

Limit Precision Inputs

- Typically start with limited precision
  - A/D only sample to 12b
    - Real-world had more precise value, but didn't capture
  - Quantized data stored in representation
    - Sound samples, DCT frequency coefficients
- We start with error
  - What does that mean about values we calculate?

Interval Analysis

- Treat every value as an interval arrange
- Model effects of operations on range of results
- A=(A.H, A.L)
- A+B=(A.H+B.H,A.L+B.L)
- Positive A, B and B interval not cross 0
  - A*B=(A.H*B.H,A.L*B.L)
  - A/B=(A.H/B.L,A.L/B.H)

Interval Analysis

- A*B=(max(A.H*B.H,A.H*B.L,A.L*B.H,A.L*B.L), min(A.H*B.H,A.H*B.L,A.L*B.H,A.L*B.L))
- A/B … more complicated
  - If B.H positive, B.L negative, can become infinity

Limited Precision

- A=(A.H,A.L)=(A+ε,A-ε)=A±ε
- E.g. if rounded to Fixed Point 16.8
  - ε may be 2^{-9}

Preclass 2

- Complete Table

Calculation	Result Range (report like A)
$Y = 1 + A$	$A_8 + 1 \pm \epsilon$
$Y = A + B$	
$Y = 2A$	
$Y = A \times B$	

Multiplication

- (A±ε)(B±ε)
- A*B ± ((A+B) ε + ε^2)
- A and B can be MAXVAL, MINVAL
  - Assume symmetric (MAXVAL=-MINVAL)
- A*B ± (2*MAXVAL* ε + ε^2)
  - Probably reasonable to drop ε^2
- A*B ± 2*MAXVAL* ε

Preclass 3c

- Recall: Fixed-Point 16.8 multiply
  - Full precision result: 32.16
- What is error interval for result of 16.8 multiply?

Preclass 3c

- For 16.8, ε_0=1/512, maxval=256
- From answer multiply in preclass 2
  - A*B ± 2*MAXVAL* ε
  - So, get A*B ± 2*256*(1/512)
  - Or A*B±1

Observe

- Full precision may keep bits well below the error in the calculation

Rounding

- Rounding introduces an error
- Round Nearest A to Fixed-Point N.8

  $\varepsilon=2^{-9}$
- As does truncation, floor, ceil…
  - But asymetric interval

Preclass 3d

- Error range if round 32.16 result to 18.2?
  - (from 16.8 multiply)

Preclass 3d

- Rounding 32.16 to 18.2 introduces a quantization error of 2^{-3}
- Our 32.16 multiply result was ±1
  - Add an addition ±1/8
  - Total error: ±9/8

Result

- Dropping (rounding) bits may not increase error range (much)

Symbolic

- If work through computation symbolically, can generate equation for error
- Each rounding (precision drop) adds some ε_i precision

Symbolic Example (start)

- Start A, B, C, D, E with $maxval_0$, ε_0
- Y=((A+B)*C+D)*E;
- A+B $maxval_1=2maxval_0$, $\varepsilon_2=2\varepsilon_0+\varepsilon_1$
  - ε_1 for round after this operation
- (A+B)*C
  - $maxval2=maxval_1*maxval_0$
  - $\varepsilon_4=\varepsilon_2*maxval0+\varepsilon_0*maxval1+\varepsilon_3$
  - ε_3 for round at this operation

Result Precision

- After series of operations, may have expression like:
  - $Y\pm(3\varepsilon_2+\varepsilon_1+1024*\varepsilon_0)$
- If looking for result with precision $\pm\varepsilon_{res}$

  $\varepsilon_{res}\geq(3\varepsilon_2+\varepsilon_1+1024*\varepsilon_0)$

Result Precision

- Fixed Precision12.0 = Round(val)
  - E.g. A/D output
  - Only need to know val to $\varepsilon=1/2$
- Fixed Precision 12.0 = Round(val/4)
  - E.g. Quantized value stored in file
  - Only need to know val to $\varepsilon=2$
  - $\varepsilon_{res}\geq(3\varepsilon_2+\varepsilon_1+1024*\varepsilon_0)$
    - Maybe: $\varepsilon_0=1/1024$, $\varepsilon_1=1/2$, $\varepsilon_2=1/8$

Optimize Precision Allocation

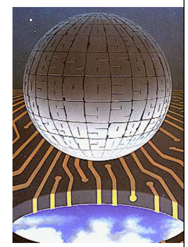
$\varepsilon_{res}\geq(3\varepsilon_2+\varepsilon_1+1024*\varepsilon_0)$
- Maybe: $\varepsilon_0=1/1024$, $\varepsilon_1=1/2$, $\varepsilon_2=1/8$

- More generally
  - Combine with area model and look at expense of providing each ε_i
- Try pick ε_i to meet ε_{res} while minimizing area

Tools

- Tools can automate interval calculations to verify precision
  - E.g. Gappa++
  - https://bitbucket.org/mlinderm/gappa

Floating Point



Robert Tinney
(Byte circa 1980)

Floating Point

- Leading 0s aren't that useful
- Can represent more compactly by counting them
  - Only need log bits to count
- Represent value as
  - Mantissa
  - Exponent

$$value = 1.mantissa \times 2^{\exp - offset}$$

- Like Scientific Notation

Floating Point

- Floating Point means
  - Move the datapath to the interesting/ significant part of computation
  - Don't represent leading zeros
  - Don't represent less significant bits
    - Even if they are well above 1
      - In the integer portion

Revisit Divisions Preclass 1

- C=0.14378 → 1.4×2^{-1}
- D=0.00192 → 1.9×2^{-3}
- Float A/C = 4.1/0.14
  - Absolute error 0.5
  - Relative error 0.018
- Float A/D = 4.1/0.0019
  - Absolute error 3.0
  - Relative error 0.0014

Standard Floating Point

- Double Precision (64b)
  - 1 bit sign
  - 11 bit exponent
    - Offset 1023 represents 1023 to -1022
  - 53 bit mantissa (52b + implicit 1)
- Single Precision (32b)
  - 1 bit sign
  - 8 bit exponent (offset 127)
  - 24 bit mantissa (23b + implicit 1)

Expensive Add

Recall Double=712, Single= 370

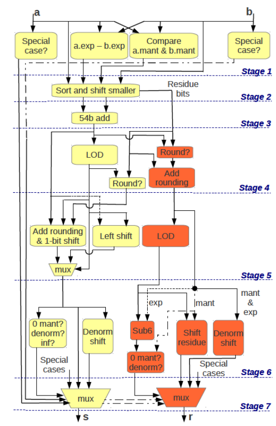
A=(A.m,A.e), B=(B.m,B.e)

- By e, sort to large, small
- Small.m>>(large.e-small.e)
- Perform mantissa addition
  - Large.m+small.m>>(large.e-small.e)
- Possibly shift to normalize result
  - Update exponent
  - Round

FP Add

- 54b add one stage of 7 in pipeline
- Done in 27 6-LUTs

[Kadric, Arith 2013]



Floating Point Multiply

Double 2229, Single 461

- Don't need to sort, pre-shift
- m=A.m*B.m (53x53 multiply dominates)
- e=A.e+B.e
- Still have shifting, rounding at end

Dynamic Range

- Floating Point has very wide dynamic range
- Can deal with significant piece being anywhere in 1023 to -1022
- For fixed-point to cover same range
  - Fixed Point 2046.1022
  - Add 1024 LUTs
  - Multiply ~ 4M LUTs
- When need dynamic range, FP economical

Customization

- Can customize on FPGA or custom silicon
  - Mantissa bits
  - Exponent bits
- Fewer bits when need less precision or range to save area
- More bits if need greater precision or range

Floating Point

Not free of precision problems

- $((1+2^{100})-2^{99})-2^{99}=0$
- $1+(2^{100}+(-2^{99}-2^{99}))=1$

Floating-Point Analysis

- Can do similar analysis on floating point
  - …and there are tools to help
  - Including Gappa++

Floating-Point vs. Fixed-Point

- Floating-Point (esp. double-precision) is a **big hammer** solution
  - Trades hardware/energy for programmer attention to needs
  - Standards have been well thought out so works over wide range
  - Most cases it is more than needed
- Cost/energy sensitive designs will ask what's necessary and tune accordingly

Economizing Precision

- FPGAs
  - Get fined-grained selection of precision
- GPUs
  - Often provide single-precision
    - Enough, allow pack more parallelism
- DSP
  - Fixed-Point DSPs cheaper, lower energy

Big Ideas

- We must always calculate with limited precision
- Precision costs area (and energy)
  - Can economize area (and energy) by judiciously using just the precision we need
  - Precision-cost tradeoff
- Can perform analysis on precision

Admin

- Project Design Space Milestone
  - Out
  - Due Friday