Algorithms and Arithmetic: Choose Wisely

George A. Constantinides[1]

Circuits and Systems Group
Imperial College London

July 13, 2017

[1]Thanks to: Dr Juan Jerez

Outline

- Designing an algorithm (and architecture) for numerical problems
- Design for CPU versus design for custom hardware / FPGA
- Case study: iterative methods for linear equations
- Automation: the state of play

Implement this algorithm

What's the starting point for design? Often: "Please implement this algorithm."

sum - algorithm

```
double sum( double x[], int n ) {
  double s = 0.0;
  for( int i = 0; i < n; i++ )
    s += x[i];
  return s;
}
```

Implement this algorithm

Can be seen as *one instance* $\mathcal{A}(\mathbb{F})$ of a family of algorithms $\mathcal{A}(T)$

sum - algorithm

```
T sum( T x[], int n ) {
  T s = 0.0;
  for( int i = 0; i <= n; i++ )
    s += x[i];
  return s;
}
```

Initial algorithm designer intent is probably to obtain a reasonable approximation to $\mathcal{A}(\mathbb{R})$.

Implement this behaviour

Sometimes, "implement this behaviour".

sum - behaviour

Precondition: $x \in \mathbb{R}^n$

Postcondition: $s \approx \sum_{i=0}^{n-1} x_i$

... for some appropriate definition of $\approx$ ...

- In general there may be many acceptable algorithms. Identify the problem with the set of acceptable algorithms, $\mathcal{P}$. So, for example, $\mathcal{A}(\mathbb{F}) \in \mathcal{P}$.
- $\mathcal{P}$ may be defined by: numerical requirements, execution time, energy consumption, silicon area, ...
- Let's imagine we know $\texttt{Quality}_{\mathcal{P}}(\mathcal{A}(T))$

Two Design Problems

General Purpose Computing

$$T_1^* = \text{argmax}_T : \mathbb{E}_{\mathcal{P},\mathcal{A}(T)\in\mathcal{P}} \left\{ \texttt{Quality}_{\mathcal{P}}(\mathcal{A}(T)) \right\} \qquad (1)$$

Choose a datatype that generally works well across a range of algorithms solving a range of problems of interest.

Custom / Accelerator Design

$$A_2^*(T_2^*)\,[\mathcal{P}] = \text{argmax}_{\mathcal{A}(T)\in\mathcal{P}} : \texttt{Quality}_{\mathcal{P}}(\mathcal{A}(T)) \qquad (2)$$

Chose an algorithm / datatype combination that works best for your specific problem.

Custom Computing Wins

$$\forall\mathcal{P}.\forall\mathcal{A}.\texttt{Quality}_{\mathcal{P}}(\mathcal{A}_2^*(T_2^*)\,[\mathcal{P}]) \succeq \texttt{Quality}_{\mathcal{P}}(\mathcal{A}(T_1^*)) \qquad (3)$$

Yet many designs import algorithm and types from GPC.

Iterative Methods

The Problem

Solve $Ax = b$

The Lanczos Process

Given q_1 such that $\|q_1\|_2 = 1$ and an initial value $\beta_0 := 1$
for $i = 1$ to ... i_{max} **do**
$q_i \leftarrow \frac{q_i}{\beta_{i-1}}$
$z_i \leftarrow Aq_i$
$\alpha_i \leftarrow q_i^T z_i$
$q_{i+1} \leftarrow z_i - \alpha q_i - \beta_{i-1} q_{i-1}$
$\beta_i \leftarrow \|q_{i+1}\|_2$
end for

Floating-point arithmetic seems to be a natural fit.

Algorithm / Datatype Codesign

An Alternative

1. Compute M diagonal with $M_{kk} = \sum_{j=1}^{n} |A_{kj}|$ in floating-point
2. Solve $M^{-\frac{1}{2}} A M^{-\frac{1}{2}} y = M^{-\frac{1}{2}} b$ with fixed-point Lanczos
3. Recover $x = M^{-\frac{1}{2}} y$ in floating-point

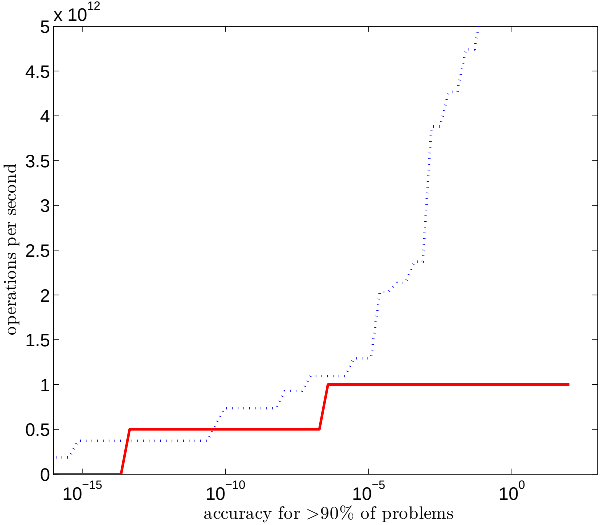
Theorem

Given preconditioner M, the symmetric Lanczos algorithm applied to $\widehat{A} := M^{-\frac{1}{2}} A M^{-\frac{1}{2}}$, for any symmetric matrix A, has intermediate variables with the following bounds:

- $q_i \in (-1, 1)$
- $\widehat{A} \in (-1, 1)$
- $\widehat{A} q_i \in (-1, 1)$
- $\alpha_i \in (-1, 1)$
- $\beta_i q_{i-1} \in (-1, 1)$
- $\alpha_i q_i \in (-1, 1)$
- $\widehat{A} q_i - \beta_{i-1} q_{i-1} \in (-2, 2)$
- $q_{i+1} \in (-1, 1)$
- $q_{i+1}^T q_{i+1} \in (-1, 1)$
- $\beta_i \in (\epsilon, 1)$
- $\frac{1}{\beta_i} \in (0, 1/\epsilon)$

where ϵ is determined by termination condition.

FPGA hybrid versus peak GPU float



Automation - State of Play

Custom Formulation

$$\text{maximise}_{\mathcal{A}(T)\in\mathcal{P}} : \texttt{Quality}_{\mathcal{P}}(\mathcal{A}(T)) \qquad (4)$$

- Most research limited to fixed $\mathcal{A}$ and typically to parametric T, *e.g.* $T = \mathbb{F}_m$.
  - Numerical aspects of `Quality` generally worst-case error instantaneous error or SNR
  - $\mathcal{A}$ limited in analytic cases, especially lacking in rich control structures
- Some research looking at the converse problem of fixed T
  - https://admk.github.io/soap/plot.html
- Lots of exciting work still to do.

Conclusion

- Custom design gives freedom - use it!
- Data type and algorithm selection are interrelated problems
- Floating point on FPGAs allows us a road into customisation for new customers
- Lots of research just waiting to be done... do some with me!