Optimization of Signal Processing Algorithms

Raza Ahmed† and *Brian L. Evans*‡

† Measurement Business Division, Tektronix Inc.,

Redmond, OR 97756-0227

‡ Dept. of Electrical and Computer Engineering,

The University of Texas, Austin, TX 78712-1084

Asilomar Conference on Signals, Systems, and Computers

Pacific Grove, CA, November 6, 1996

Outline

Optimize Signal Processing Algorithms Using

1. Comprehensive collections of algebraic identities for signal processing algorithms

2. Search mechanisms to apply the identities in an intelligent manner

3. Accurate estimates of implementation cost

Facility	Environment	Extensions
Algebraic Identities	*Mathematica*	Signal Processing Packages
Search Mechanisms	*Mathematica*	Heuristic Search Packages
Cost Estimates	*Ptolemy*	Code Generation Cost Target

Motivation

Algorithm performance

1. Hardware: area, speed, power
2. Software: program memory, data memory, speed

Goals

1. Optimize a weighted combination of performance criteria subject to constraints
2. Improve performance to meet design constraints

Example: Design of touchtone decoder

Algebraic Representations of Signals

Signals as Functions

- Input signal $x[n]$ becomes `x[n]` without any definition given for the signal
- Impulse response of a digital FIR filter with filter taps 1, 2, and 1:

  `DigitalFIRFilter[ {1,2,1}, n ][ x[n] ]`
- Causal exponential sequence $a^n u[n]$ becomes

  `a^n DiscreteStep[n]`

Algebraic Representations of Systems

Systems as Operators

- Operators are represented in the form

  operator [*parameters*][*inputs*]

- Upsample by L

  `Upsample[L, n][ x[n] ]`

- Interpolation as an FIR following an upsampler

  `DigitalFIRFilter[{1,2,1}, n][Upsample[L, n][x[n]]]`

Algebraic Identities

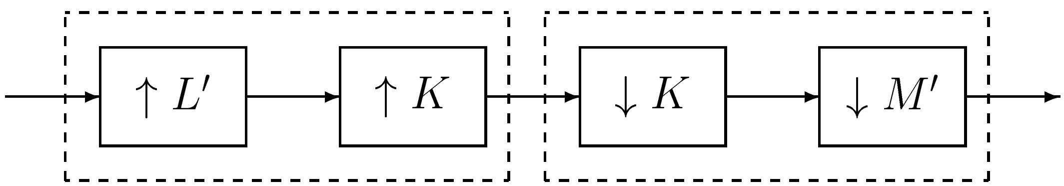
Based on System Properties

System Property	Meaning
`Associative`	can change grouping of inputs
`Additive`	distributes over addition
`Commutative`	can change order of inputs
`Continuous`	inputs are continuous signals
`Delay`	amount of delay before output is meaningful
`Discrete`	inputs are discrete signals
`Homogeneous`	scaled input gives scaled output
`Linear`	additive and homogeneous
`Linear Phase`	true if the frequency phase response is a linear function of the frequency variable
`Memoryless`	output does not depend on previous inputs or outputs; if a single-input system, then `Shift Invariant`
`Separable`	true if separable in all dimensions, false if completely non-separable, or a list of variables in which the operator is separable
`Shift Invariant`	shifted input gives shifted output

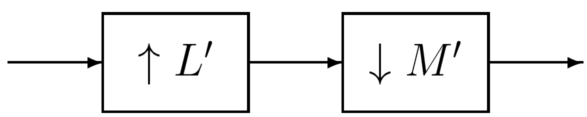
Algebraic Identities

Signal Processing Identities

- one-dimensional multirate rules collected by Myers and Covell
- multidimensional multirate rules reported by Evans *et al.*



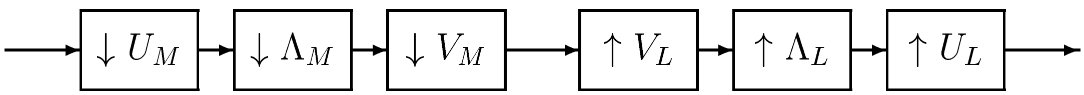
(a)



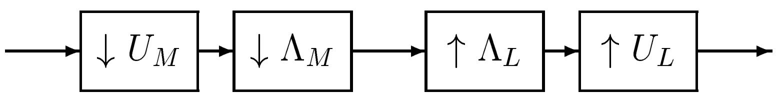
(b)

Algebraic Identities

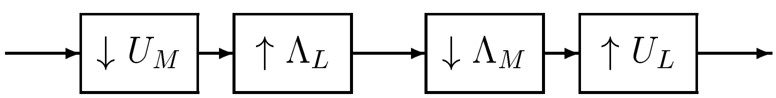
Signal Processing Identities



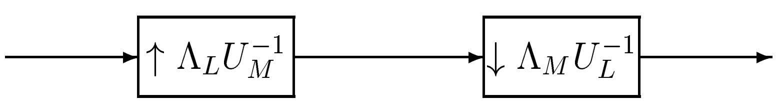
(c) Cascade in Smith Form



(d) Simplified cascade *if* $V_M = V_L$



(e) Reversing order of operations in (b) *if* Λ_M and Λ_L are coprime



(f) Combining operations in (c)

Heuristic Search

Inputs

Expression to optimize

Algebraic identities

Successor function

- takes an expression and algebraic identities
- returns a set of equivalent forms of the expression

Evaluation function

Heuristic Search

Algorithm Framework

- Initial state is the original equation
- Generate successor states by applying algebraic identities
- Choose a successor state
- Check stopping criteria, and repeat if not met

Heuristic Search

Uninformed Search

Cannot guess the location of the goal state

Exponential time and memory requirements in the worst case

Goal state is one that reduces cost by a certain amount

Search the tree of successor states

- from top to bottom in breadth-first searching
- from bottom to top in depth-first searching

Depth-first is better when

- many goal states exist
- goal state is in deepest layers of the tree

Heuristic Search

Informed Search

Use heuristic to navigate to the goal state

Exponential time but linear memory requirements in worst case

Hill Climbing

- Initial state is the original equation
- Generate successor states by applying algebraic identities
- Choose the successor state with the lowest cost
- Process continues until no better successor state can be found

Sensitive to local minima, flat valleys, crevices of solution space

Can restart hill climbing at a randomly chosen subexpression

Heuristic Search

Simulated Annealing

- Initial state is the original equation
- Generate successor states by applying algebraic identities
- Choose a successor state at random
  - if the state has a lower cost, take it
  - otherwise, take the state with a probability of inversely proportional to the number of iterations (cooling schedule)
- Process continues until no better successor state can be found

Becomes hill climbing as the number of iterations get large

Heuristic Search

Example

```
In[3]:= poly = A x + B x^2 + C x^3 + D x^4 + E x^5 + F x^6
                 2     3     4     5     6
Out[3]= A x + B x  + C x  + D x  + E x  + F x

In[4]:= {timing, optpoly} =
           Timing[HillClimbing[poly,
                    EvaluationFunction ->
                      PolynomialEvaluationCost,
                    SuccessorFunction ->
                      PolynomialSuccessorFunction]]
Out[4]= {1.41667 Second,
           x (A + x (B + x (C + x (D + x (E + F x)))))}

In[6]:= optcost = PolynomialEvaluationCost[optpoly]
Out[6]= 35

In[7]:= initcost = PolynomialEvaluationCost[poly]
Out[7]= 110

In[8]:= FactorReductionInCost = N[initcost/optcost]
Out[8]= 3.14286
```

Cost Estimates

System Design Tools

Describe how algorithms are computed

Measure implementation costs

Restrict algorithm rearrangements

Our Approach

Model computation in algorithms using Synchronous Dataflow (SDF)

Decide admissible rearrangements using SDF Composition Theorem

Extend Ptolemy code generation to report implementation costs

Cost Estimates

Synchronous Dataflow

Produces static schedules (easy to estimate implementation costs)

Every subsystem produces and consumes a fixed number of samples

Dependency between computation must be static

Examples

Conclusion

Optimize Signal Processing Algorithms Using

1. Comprehensive collections of algebraic identities for signal processing algorithms
2. Search mechanisms to apply the identities in an intelligent manner
3. Accurate estimates of implementation cost