Optimization of Arithmetic Datapaths with Finite Word-Length Operands

Sivaram Gopalakrishnan[1], Priyank Kalla[1] and Florian Enescu[2]

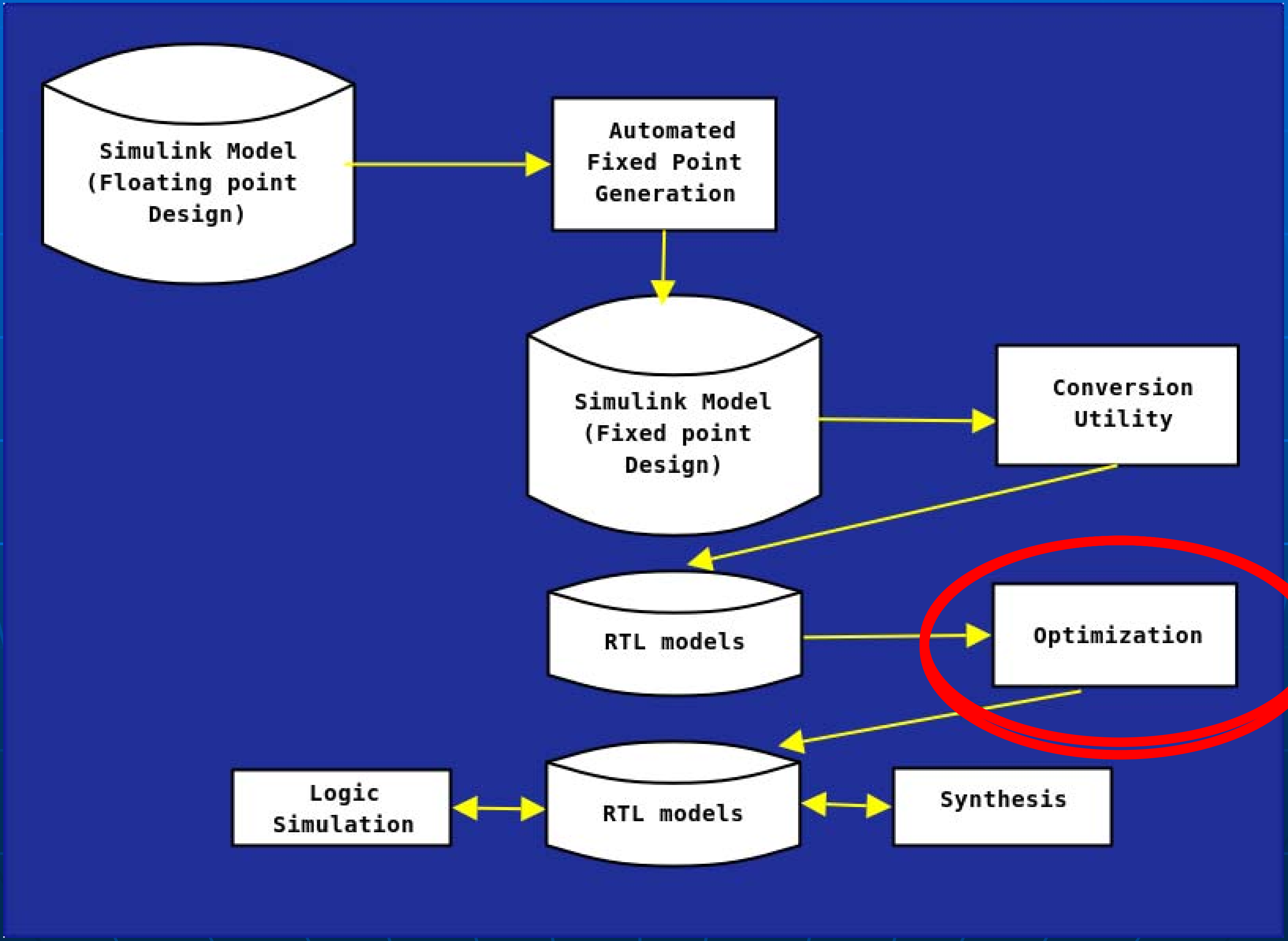
[1]Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT-84112

[2]Department of Mathematics and Statistics, Georgia State University, Atlanta, GA-30303

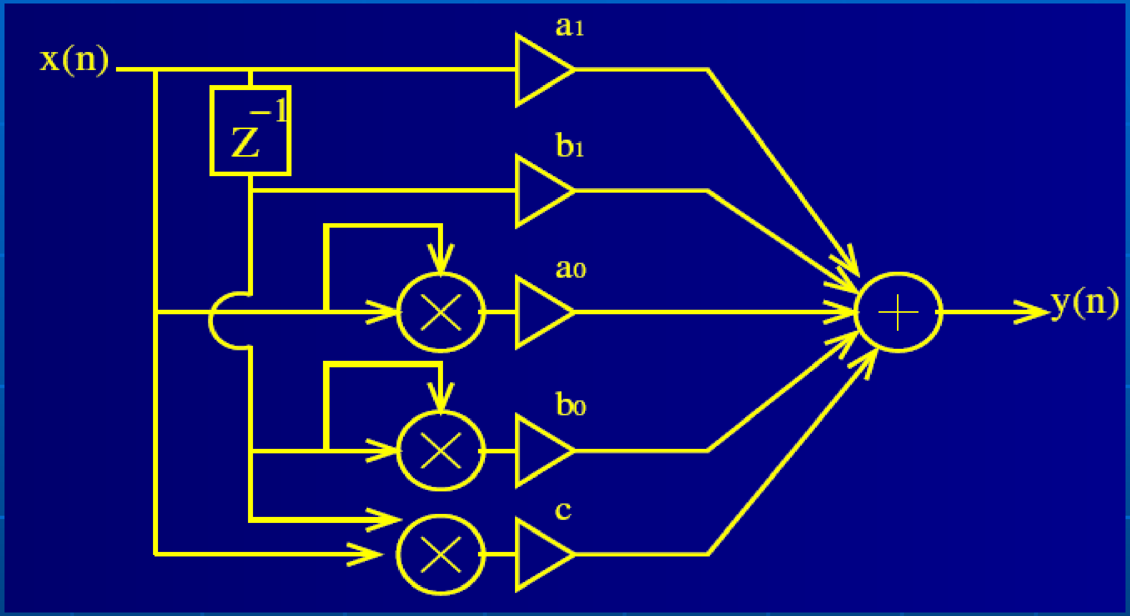
Outline

- Introduction to the datapath optimization problem
  - Our Focus: Arithmetic with Finite Word-Length Operands
- Problem Modeling
  - Polynomial Functions over Finite Integer Rings
- Previous Work and Limitations
- Approach and Contributions
  - Reducibility of Polynomials over Finite Rings
  - Cost Model
  - Integrated CAD Approach
- Results: Area optimization
- Conclusions & Future Work

The Optimization Problem



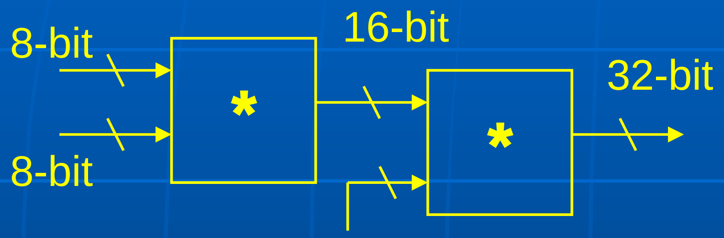
Polynomials over Bit-Vectors?

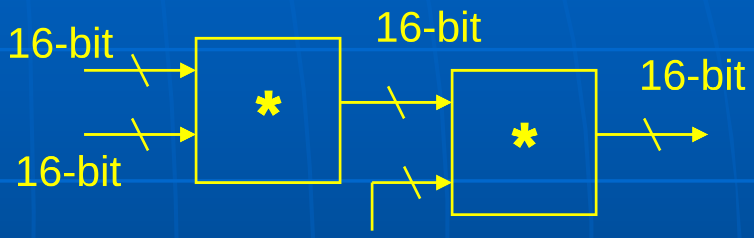


- Quadratic filter design for polynomial signal processing
- $y = a_0 . x_1^2 + a_1 . x_1 + b_0 . x_0^2 + b_1 . x_0 + c . x_0 . x_1$
- Coefficients/variables implemented with specific bit-vector sizes

Fixed-Size (m) Data-path: Modeling

- Control the datapath size: Fixed size bit-vectors (m)



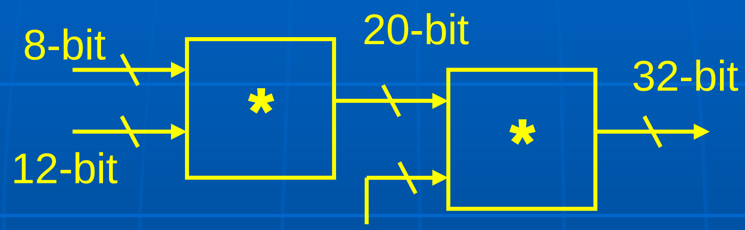


- Bit-vector of size m: integer values in $0,\ldots, 2^m-1$

Fixed-size (m) bit-vector arithmetic ⇒ Polynomials reduced $\%2^m$ ⇒ Algebra over the ring Z_{2^m}

Multiple Bit-Width Operands

- Bit-vector operands with different word-lengths



- Input variables : $\{x_1,\ldots, x_d\}$ Output variables: f, g
- Input bit-widths: $\{n_1,\ldots, n_d\}$ Output width: m

$$x_1 \in Z_{2^{n_1}}, x_2 \in Z_{2^{n_2}} \ldots f, g \in Z_{2^m}$$

- Model as polynomial function

$$Z_{2^{n_1}} \times Z_{2^{n_2}} \times \cdots \times Z_{2^{n_d}} \rightarrow Z_{2^m}$$

Arithmetic Datapath: Implementation

- Signal Truncation: Unsigned/Overflow Arithmetic
  - Keep lower order m-bits, ignore higher bits
  - $f \% 2^m$
- Fractional Arithmetic with rounding
  - Keep higher order m-bits, round lower order bits
- Saturation Arithmetic
  - Saturate at overflow
  - If(*x[7:0] > 255*) then *x[7:0] = 255*;
  - Used in image-processing applications

Conventional Methods

- Extracting control-dataflow graphs (CDFGs) from RTL
  - Scheduling
  - Resource sharing
  - Retiming
  - Control synthesis
- Algebraic Transforms
  - Factorization
  - Common Sub-expression Elimination
  - Term-rewriting
  - Tree-Height Reduction
- Overlook the effect of bit-vector size (m)

Previous Work

- Polynomial models for complex computational blocks
- Guiding Synthesis engines using Groebner's basis [Peymandoust and De Micheli, TCAD 02]
  - Given polynomial F and Library elements <I_1, …, I_n>
  - $F = h_1 I_1 + \ldots\ldots + h_n I_n$
- Computations over R, Q, Z, Z_p (Galois Fields)
  - Unique Factorization Domains (UFDs): *Uniquely* factorize into *irreducibles*
  - Polynomial approximation (do not account the effect of bit-vector size)
- Datapath allocation for multiple-wordlength operands [Constantinides et al, TVLSI 05]
  - Operates on the given expression

Why is the Problem Difficult?

- Z_{2^m} is a non-UFD
  - $f = x^2 + 6x$ in Z_8 can be factorized as

- Factorization in non-UFDs is therefore hard !!!
- Scope to explore multiple factorizations

Example: Polynomial Filter

- A Polynomial filter (f) over a uniform 16-bit datapath

$$f_1 = 16384x^5 + 19666x^4 + 38886x^3 + 16667x^2 + 52202x + 1$$

- Area: 42910 sq. units

- Alternatively, (f) can be implemented as

$$f_2 = 3282x^4 + 22502x^3 + 283x^2 + 52202x + 1$$

- Area: 28840 sq. units

$$f_1 \neq f_2, but \; f_1 \% 2^{16} \equiv f_2 \% 2^{16}, \; f_1[15:0] = f_2[15:0]$$

Digital Image Rejection Unit

```
input A[11:0], B[7:0];
output Y1[15:0], Y2[15:0];
```

$$Y_1 = 16384(A^4 + B^4) + 64767(A^2 - B^2) + A - B + 57344AB(A - B)$$

$$Y_2 = 24576A^2B + 15615A^2 + 8192AB^2 + 32768AB + A + 17153B^2 + 65535B$$

- $Y_1 \neq Y_2$
- $Y_1[15:0] = Y_2[15:0]$
- $Y_1 \% 2^{16} \equiv Y_2 \% 2^{16}$

Problem Modeling

- Polynomial Model:
  - **$Y_1(A_{12}, B_8)\%2^{16} \equiv Y_2(A_{12}, B_8)\%2^{16}$**
  - Y_1, Y_2: $Z_{2^{12}} \times Z_{2^8} \rightarrow Z_{2^{16}}$ are equal as functions

- Consider Y_1 - Y_2
  - $Y_1 - Y_2 \equiv 0 \% 2^{16}$

$$Y_1 - Y_2 = 16384(A^4 + B^4) + 32768AB(A+1) + 49152(A^2 + B^2)$$
$$\equiv 0\%2^{16}$$

- $Y_1 - Y_2$ *vanishes* as a function from $Z_{2^{12}} \times Z_{2^8} \rightarrow Z_{2^{16}}$
- $Y_1 - Y_2$ is known as the ***vanishing polynomial***

Vanishing Polynomials for Reducibility

- In Z_{2^3}, say $\boldsymbol{f(x) = 4x^2}$ and V(x) is a vanishing polynomial
  - $f(x) = f(x) - V(x)$
  - Generate V(x)
  - $V(x) = 4x^2 + 4x \equiv 0 \% 2^3$
- Reduce by subtraction:
  - $$\begin{array}{rrl} & 4x^2 & f(x) \\ - & 4x^2 + 4x & V(x) \\ = & -4x & = -4x \% 8 = 4x \end{array}$$
  - $4x^2$ can be reduced to $4x$
  - ***Degree reduction***

Vanishing Polynomials for Reducibility

- Degree is not always reducible

- In Z_{2^3}, $f(x) = 6x^2$

- Divide and subtract
  - $6x^2 = 2x^2 + 4x^2 \% 2^3$
  - $4x^2$ can be reduced to $4x$

- $f(x) = 2x^2 + 4x$: *Lower Coefficient*
  - ***Coefficient reduction***

Results From Number Theory

- *n*! divides a product of *n* consecutive numbers
  - 4! divides 99 X 100 X 101 X 102
- Find least *n* such that $2^m | n!$
  - *Smarandache Function (SF)*
  - $SF(2^3) = 4$, since $2^3 | 4!$
- 2^m divides the product of $n = SF(2^m)$ consecutive numbers
  - 2^3 divides the product of 4 consecutive numbers

Results From Number Theory

- $F \equiv 0 \% 2^3$
  - $2^3|F$ in Z_{2^3}
  - 2^3 divides the product of 4 consecutive numbers

If F is a product of 4 consecutive numbers
then $2^3|F$

- A polynomial as a product of 4 consecutive numbers?

$$(x)\,(x-1)\,(x-2)\,(x-3)$$

Basis for Factorization: One Variable

- $Y_0(x) = 1$
- $Y_1(x) = (x)$
- $Y_2(x) = (x)(x - 1)$ = Product of *2* consecutive numbers
- $Y_3(x) = (x)(x - 1)(x - 2)$ = Product of *3* consecutive numbers
- …
- …
- $Y_k(x) = (x – k + 1)\ Y_{k-1}(x)$ = Product of k consecutive numbers

Rule 1: Degree is k. If k $\geq$ n
where n = SF(2^m), use $Y_k(x)$ (degree reduction)

Straight forward extension to multiple variables with finite word-lengths

Constraints on the Coefficient

- $F(x) = 4x^2 - 4x = 4(x)(x-1) \% 2^3 = 0 \% 2^3$

  compensated by constant

- In Z_{2^3}
  - $Y_4(x) = (x)(x-1)(x-2)(x-3)$

    missing factor

Rule 2: if Coefficient $\geq b_{\mathbf{k}}$ where $b_k = 2^m/\gcd(\mathbf{k}!, 2^m)$, then use

$a_k.\ b_k.\ Y_k$ (for coefficient reduction)

- Here, Coefficient of $F(x) = 4$, Degree of $F(x) = 2$
- $b_{<2>} = 2^3/\gcd(2!, 2^3) = 4$ (coefficient's value!!!)

Example

- Consider x^4 in Z_8

x^4

k =4, SF(8)=4, So $V(x) = Y_4(x)$ (Rule 1)

$V(x) = x(x-1)(x-2)(x-3)$

Degree Reduction

$6x^3 + 5x^2 + 6x$

k=3 <SF(8), $b_k = 8/(8,6) = 4$, Coefficient = 6

$V(x) = 1.4.Y_3(x)$ (Rule 2)

$V(x) = 4.x(x-1)(x-2)$

Coefficient Reduction

$2x^3 + x^2 + 6x$ (Canonical Form)

Our Approach

- Say $f(x) = a_k x^k + a_{k-1} x^{k-1} + \ldots + a_0$
  - In decreasing total degree order
- Given *f(x)* and the input/output bit-vector sizes
  - Check if degree can be reduced
  - Check if coefficient can be reduced
  - Perform corresponding reductions to get an intermediate expression
  - Estimate the cost of the intermediate expression
  - Repeat for all monomials …
  - Finally, when *f* (*x*) **is** in the reduced, minimal, unique form, identify the expression with the least cost

Exploring more solutions

- Consider $f = x^6+8x^3+8x$ in Z_{16}
- Reduction of f leads to following intermediate forms

$f = x^6+8x^3+8x$ $\xrightarrow{Y_6(x)}$ $f_1 = 11x^5+x^4+9x^3+8x^2+4x$

$\downarrow$ $5.2.Y_5(x)$

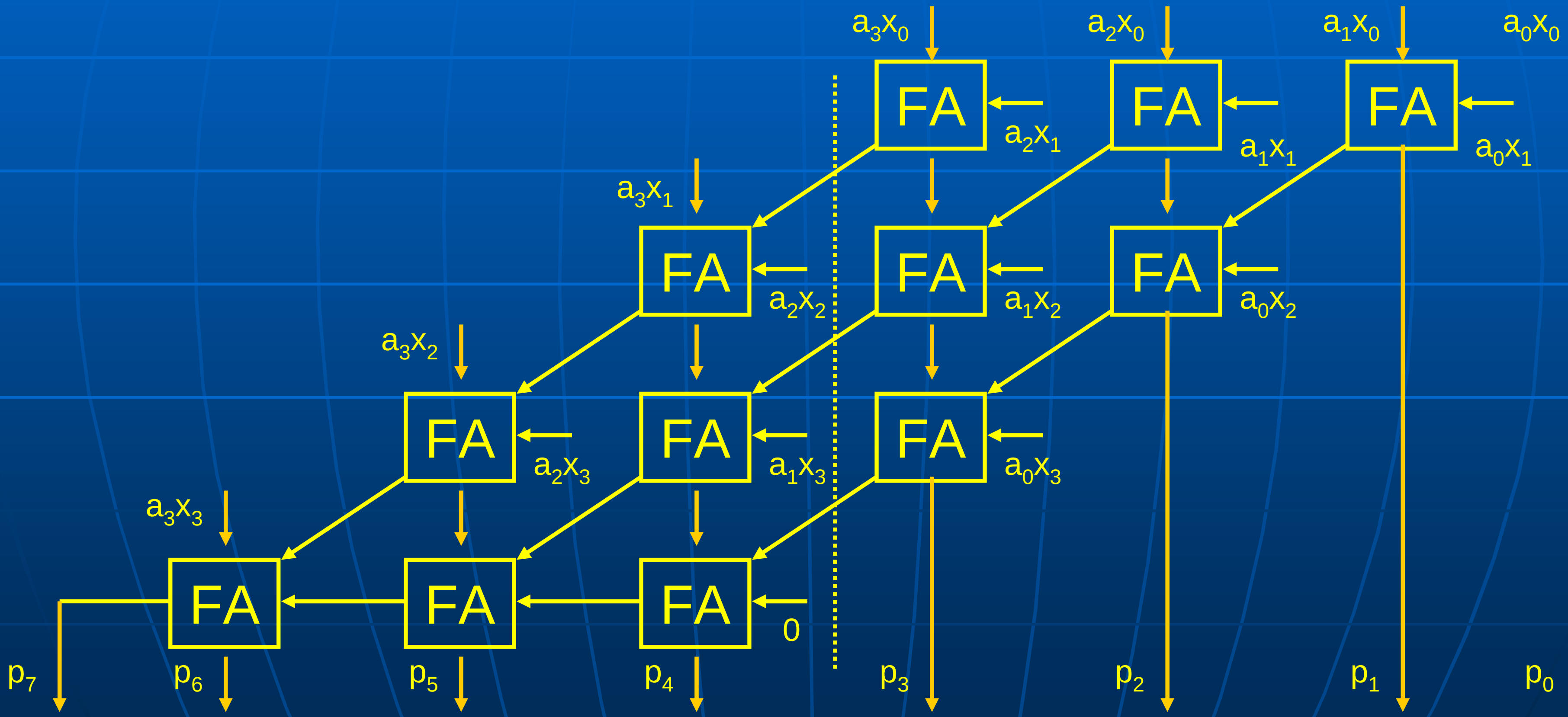
$f_3 = x^5+x^4+3x^3+12x$ (Canonical form) $\xleftarrow{5.2.Y_4(x)}$ $f_2 = x^5+11x^4+7x^3+14x^2+4x$

- Reducing only $8x^3+8x$ leads to 0 (vanishing polynomial!!!)
- f reduces from x^6+8x^3+8x to x^6
- x^6 is a better implementation!!!

Cost Model

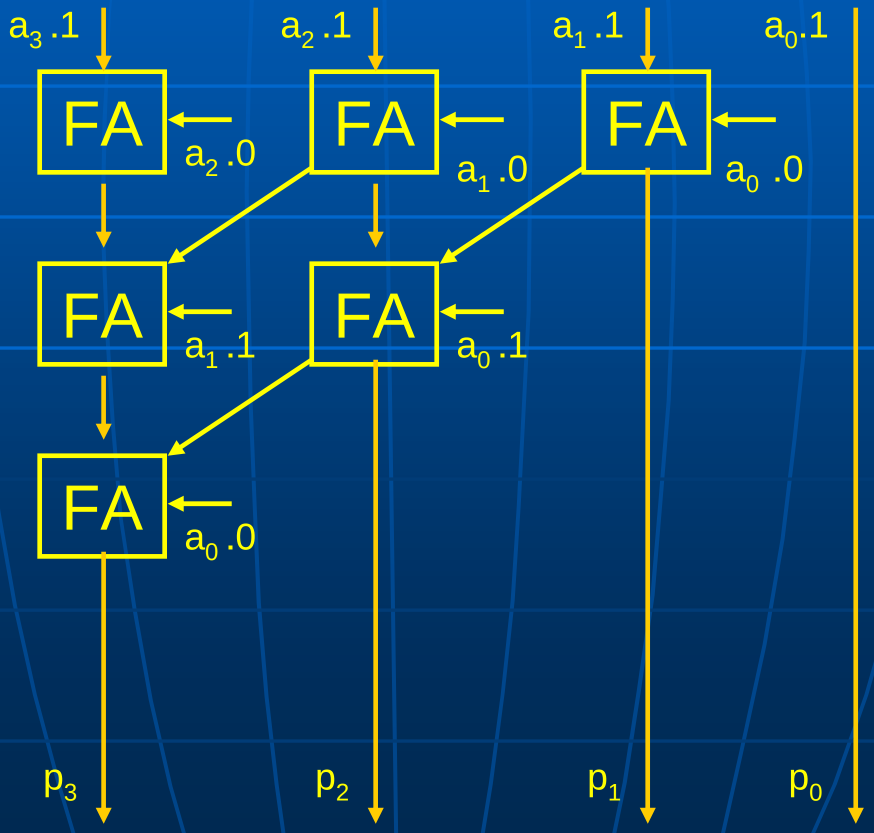
- Adder(m-bit)= m* Cost (Full Adder)

- MULT(m-bit)= Partial products + Array



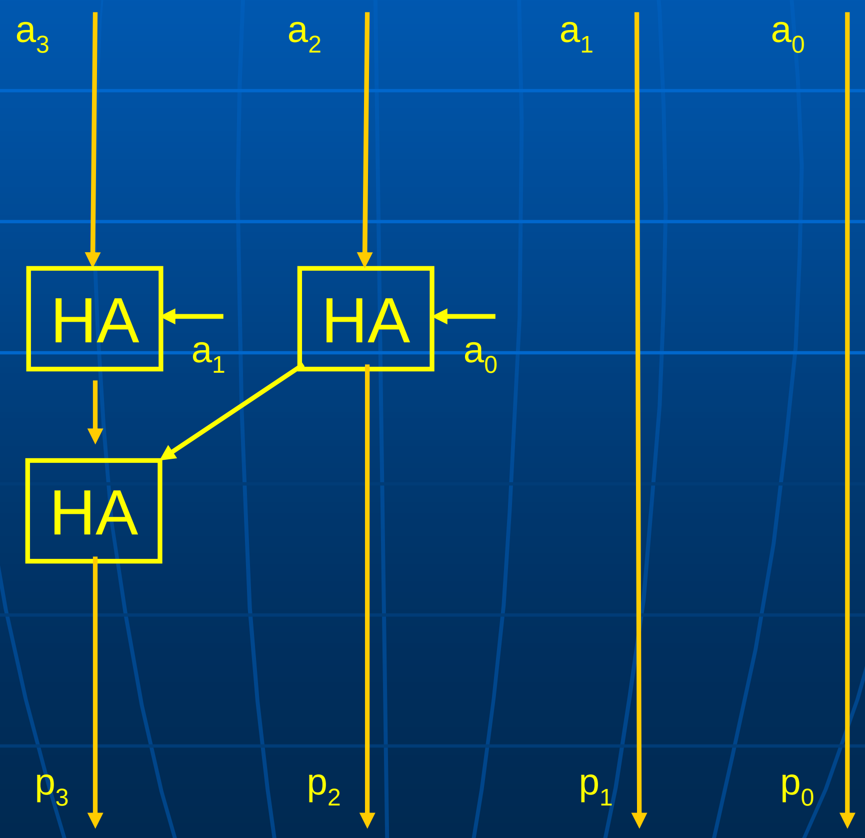
Cost Model

- Constant Multiplier: Simplification by constant propagation
  - Analyze the bit pattern of the constant
  - Propagate the bits using the array multiplier model
- Example 1: 5A, Bit pattern of 5 is {0101}



Cost Model

- Constant Multiplier: Simplification by constant propagation
  - Analyze the bit pattern of the constant
  - Propagate the bits using the array multiplier model
- Example 1: 5A, Bit pattern of 5 is {0101}



Cost is
3*Cost (HA)

Results

Benchmark	Est. Cost			Imp. Cost			Selection
	Orig	Min	Improv	Orig	Min	Improv	
Poly1	7581	3766	50.3%	37430	20628	44%	Minimal
Poly2	4820	2393	50.3%	28848	11684	59.49%	Minimal
Poly3	6227	5465	11.7%	28840	23006	20.2%	Minimal
Poly_unopt	5196	2994	42.3%	28836	14424	49.9%	Minimal
Deg4	22731	16361	28%	116684	82718	29.1%	Minimal
Janez	8907	6163	30.9%	42910	28840	32.7%	Minimal
Mibench	58510	48226	17.6%	249290	216772	13.04%	Intermed
IRR	10864	6943	37.3%	54594	37792	30.77%	Minimal
Antialias	15997	12011	24.9%	79254	59712	24.65%	Intermed
PSK	18140	18140	<1%	76876	-	-	Orig
Cubic	47595	47586	<1%	256388	-	-	Orig
IIR-4	49339	49333	<1%	213408	-	-	Orig

Average area improvement: 23%

Results

Benchmark	Est. Cost			Imp. Cost			Selection
	Orig	Min	Improv	Orig	Min	Improv	
Poly1	7581	3766	50.3%	37430	20628	44%	Minimal
Poly2	4820	2393	50.3%	28848	11684	59.49%	Minimal
Poly3	**6227**	**5465**	**11.7%**	**28840**	**23006**	**20.2%**	**Minimal**
Poly_unopt	5196	2994	42.3%	28836	14424	49.9%	Minimal
Deg4	22731	16361	28%	116684	82718	29.1%	Minimal
Janez	8907	6163	30.9%	42910	28840	32.7%	Minimal
Mibench	**58510**	**48226**	**17.6%**	**249290**	**216772**	**13.04%**	**Intermed**
IRR	10864	6943	37.3%	54594	37792	30.77%	Minimal
Antialias	15997	12011	24.9%	79254	59712	24.65%	Intermed
PSK	18140	18140	<1%	76876	-	-	Orig
Cubic	**47595**	**47586**	**<1%**	**256388**	**-**	**-**	**Orig**
IIR-4	49339	49333	<1%	213408	-	-	Orig

Average area improvement: 23%

Conclusions & Future Work

- Area optimization approach for polynomial datapaths implemented with finite word-length operands
- Arithmetic datapaths are modeled as a polynomial function from
  $$Z_{2^{n_1}} \times Z_{2^{n_2}} \times \cdots \times Z_{2^{n_d}} \rightarrow Z_{2^m}$$
- $f(x_1,\ldots, x_d) \% 2^m$ is reduced to its unique canonical form $g(x_1, \ldots, x_d) \% 2^m$
  - Exploiting the concept of polynomial reducibility over
    $$Z_{2^{n_1}} \times Z_{2^{n_2}} \times \cdots \times Z_{2^{n_d}} \rightarrow Z_{2^m}$$
- Cost Model to estimate area at polynomial level
- Reduction procedure + Cost model -> Least cost expression for implementation
- Future Work involves extensions for
  - Polynomial Decomposition over such arithmetic
  - Given n-bit ADD/MULTS, synthesize an m-bit datapath

Questions?