Low-Depth, Low-Size Circuits for Cryptographic Applications

Joan Boyar*[1] Magnus Gausdal Find[2] René Peralta[2]

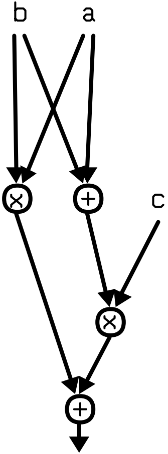
[1]University of Southern Denmark

[2]National Institute of Standards and Technology, USA

BFA 2017

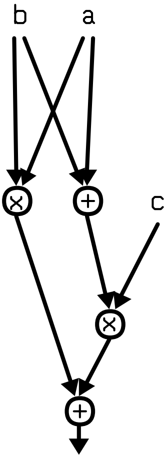
Circuits over $GF(2)$

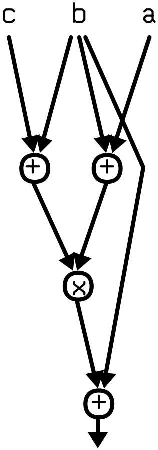
AND gates $\times / \wedge$ XOR gates $+$ XNOR gates $\#$



Circuits over $GF(2)$

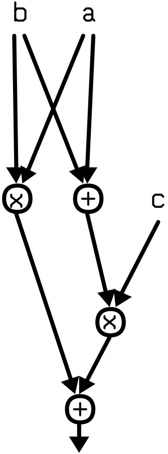
AND gates $\times/\wedge$ XOR gates $+$ XNOR gates $\#$

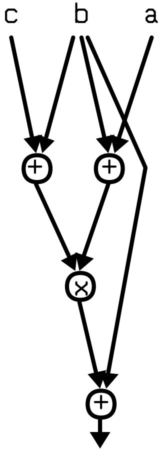




Circuits over $GF(2)$

AND gates $\times/\wedge$ XOR gates $+$ XNOR gates $\#$





Both circuits compute the *predicate* MAJ(a,b,c) in *size* 4 and *depth* 3.

Boolean Circuit Complexity

- The (Boolean) *circuit complexity* of a function f is the number of gates necessary and sufficient to compute f.

Boolean Circuit Complexity

SDU

- The (Boolean) *circuit complexity* of a function f is the number of gates necessary and sufficient to compute f.
- Shannon-Lupanov bound: the circuit complexity of a predicate on n bits is about $\frac{2^n}{n}$ almost everywhere.

Multiplicative Complexity

- The *multiplicative complexity* of a function f is the number of multiplications (ANDs) necessary and sufficient to compute f
  (over the basis AND, XOR, XNOR).

Multiplicative Complexity

- The *multiplicative complexity* of a function f is the number of multiplications (ANDs) necessary and sufficient to compute f
(over the basis AND, XOR, XNOR).
- Almost all Boolean predicates on n bits have multiplicative complexity close to $2^{\frac{n}{2}}$ (i.e. about the square root of the total number of gates needed).
[B., Peralta, Pochuev],[Nechiporuk]

Multiplicative Complexity

- The *multiplicative complexity* of a function f is the number of multiplications (ANDs) necessary and sufficient to compute f
(over the basis AND, XOR, XNOR).
- Almost all Boolean predicates on n bits have multiplicative complexity close to $2^{\frac{n}{2}}$ (i.e. about the square root of the total number of gates needed). [B., Peralta, Pochuev],[Nechiporuk]
- Our thesis is that this observation can be used for Boolean circuit optimization.

Motivation

Why do we care?

Motivation

Why do we care?

1. Smaller chip area, less power
   Lower depth, faster

Motivation

Why do we care?

1. Smaller chip area, less power
   Lower depth, faster
2. Multi-party computations:
   Communication complexity can depend (only) on the number of ANDs in the circuit.

Motivation

Why do we care?

1. Smaller chip area, less power
   Lower depth, faster
2. Multi-party computations:
   Communication complexity can depend (only) on the number of ANDs in the circuit.
3. Homomorphic computations:
   Performing computations on encrypted data, such as in the cloud.
   The multiplicative complexity can affect the number of bootstrappings.

An example function: AES S-Box

Advanced Encryption Standard (AES)

Block cipher - 128 bit blocks, 128 bit keys

An example function: AES S-Box

Advanced Encryption Standard (AES)

Block cipher - 128 bit blocks, 128 bit keys

10 rounds using 4 operations:

- SubBytes — Nonlinear substitution step (S-Box)
- ShiftRows
- MixColumns
- AddRoundKey

AES S-Box

The S-Box has 8 inputs and 8 outputs.
Inversion in $GF(2^8)$, followed by affine transformation (linear, followed by some negations).

AES S-Box

The S-Box has 8 inputs and 8 outputs.
Inversion in $GF(2^8)$, followed by affine transformation
(linear, followed by some negations).

Can be done by *table look-up*.

- 256 different inputs, each with 8 bits output
- 2048 bits
- large area — 16 S-Boxes in each round

AES S-Box

The S-Box has 8 inputs and 8 outputs.
Inversion in $GF(2^8)$, followed by affine transformation.

Tower of fields constructions:

- Concentration on size:
  - Wolkerstorfer, Oswald, Lamberger 2002
    — work over subfield $GF(2^4)$
  - Satoh, Morioka, Takano, Munetoh 2001
    — within $GF(2^4)$ use $GF(2^2)$
  - Canright 2005 — tried many different bases
  - B., Peralta 2010 — used Canright's base - 115 gates
    (improved to 113 gates by Calik; same technique, exploring all ties)

AES S-Box

The S-Box has 8 inputs and 8 outputs.
Inversion in $GF(2^8)$, followed by affine transformation.

Tower of fields constructions:

- Concentration on size:
  - Wolkerstorfer, Oswald, Lamberger 2002
    — work over subfield $GF(2^4)$
  - Satoh, Morioka, Takano, Munetoh 2001
    — within $GF(2^4)$ use $GF(2^2)$
  - Canright 2005 — tried many different bases
  - B., Peralta 2010 — used Canright's base - 115 gates
    (improved to 113 gates by Calik; same technique, exploring all ties)
    depth 28

AES S-Box

The S-Box has 8 inputs and 8 outputs.
Inversion in $GF(2^8)$, followed by affine transformation.

Tower of fields constructions:

- Depth:
  - Canright 2005 — depth 25 ($\geq$ 125 gates)
  - Nogami, Nekado, Toyota, Hongo, Morikawa 2010
    - choose mixed bases so $\leq$ 4 ones for top and bottom transformations, so depth 2 for each
    - depth 22, size 148

AES S-Box

SDU

The S-Box has 8 inputs and 8 outputs.
Inversion in $GF(2^8)$, followed by affine transformation.

Tower of fields constructions:

- Depth:
  - Canright 2005 — depth 25 ($\geq$ 125 gates)
  - Nogami, Nekado, Toyota, Hongo, Morikawa 2010
    - choose mixed bases so $\leq$ 4 ones for top and bottom transformations, so depth 2 for each
    - depth 22, size 148
  - B., Peralta 2012 — depth 16, size 128
  - this presentation — depth 16, size 125, more automated

AES S-Box

Goal: minimize size (number of gates) and depth

Technique:

1. Start with a circuit with small size
   (using previous techniques, for example [B.,Matthews,Peralta 2013])

AES S-Box

Goal: minimize size (number of gates) and depth

Technique:

1. Start with a circuit with small size
   (using previous techniques, for example [B.,Matthews,Peralta 2013])
2. Use techniques from automatic theorem proving to re-synthesize non-linear components into lower-depth constructions
   (reused from [B., Peralta 2012])

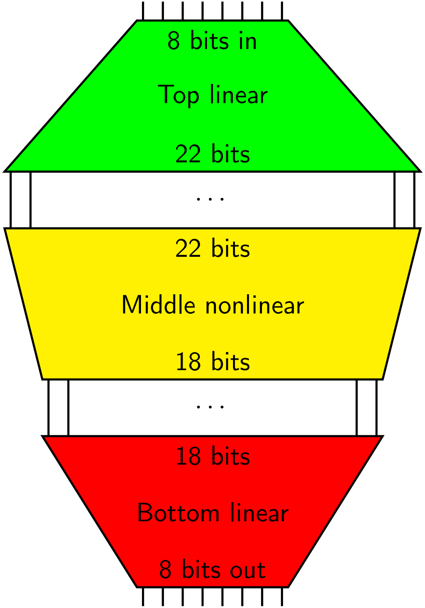
AES S-Box

Goal: minimize size (number of gates) and depth

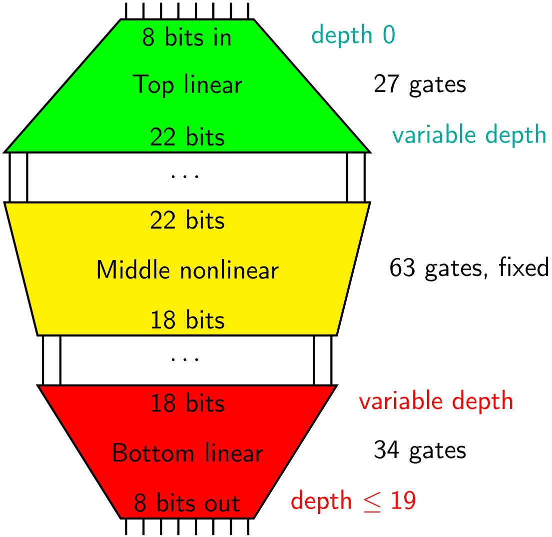
Technique:

1. Start with a circuit with small size
   (using previous techniques, for example [B.,Matthews,Peralta 2013])
2. Use techniques from automatic theorem proving to re-synthesize non-linear components into lower-depth constructions
   (reused from [B., Peralta 2012])
3. Apply a randomized, greedy heuristic to re-synthesize linear components into lower-depth constructions, using a new
   See-Saw Method

Circuit for the S-Box of AES

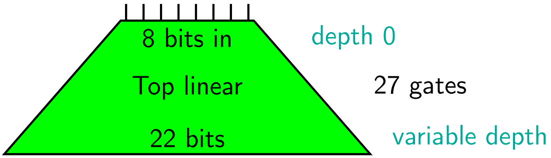


See-Saw Method

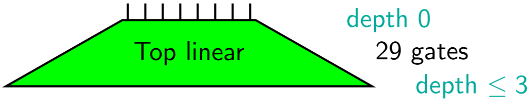


Start: Total depth 19, size 124 gates.

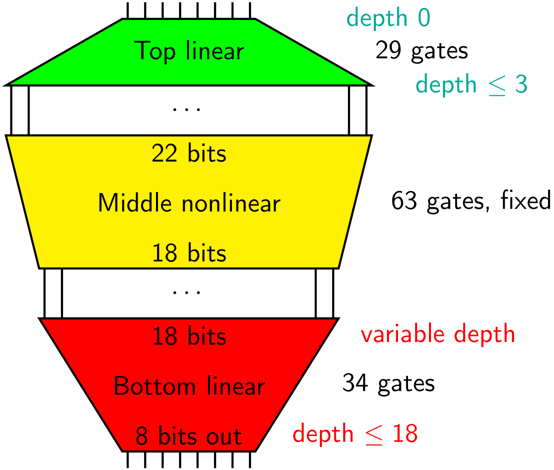
See-Saw Method



After processing....

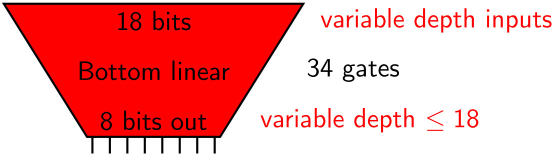


See-Saw Method



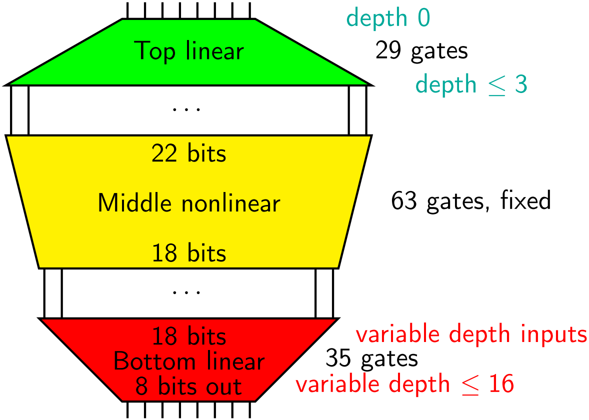
Start: Total depth 19, size 124 gates.
Now: Total depth 18, size 126 gates.

See-Saw Method



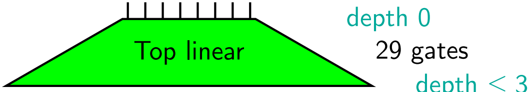
After processing....

See-Saw Method

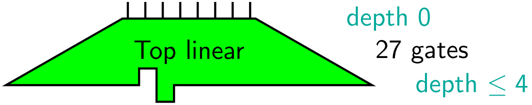


Previous: Total depth 18, size 126 gates.
Now: Total depth 16, size 127 gates.

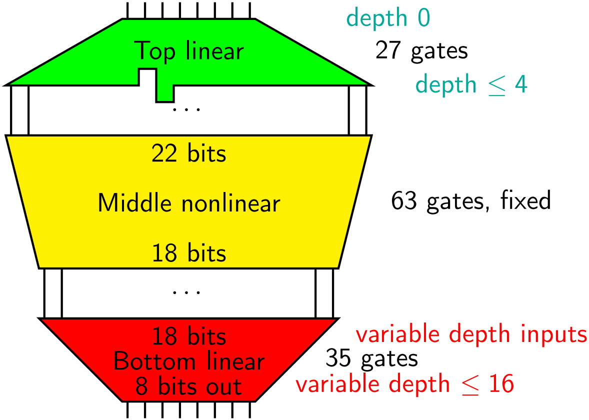
See-Saw Method



After processing....

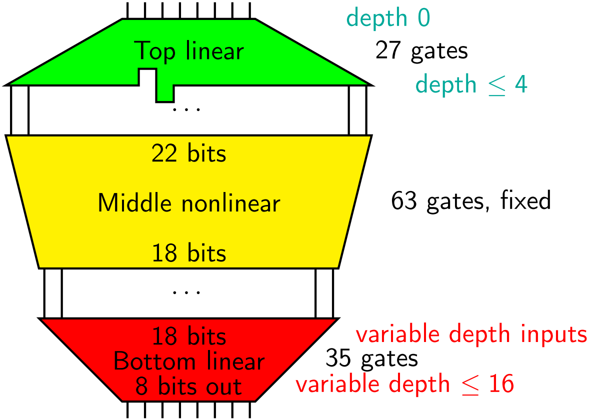


See-Saw Method



Previous: Total depth 16, size 127 gates.
Now: Total depth 16, size 125 gates.

See-Saw Method



Previous: Total depth 16, size 127 gates.
Now: Total depth 16, size 125 gates.

Work on bottom linear to get all outputs at depth 16.

Optimizing the linear components

[B.,Matthews,Peralta 2013]

It is NP-hard to find the optimal linear program (circuit).

Optimizing the linear components

[B.,Matthews,Peralta 2013]

It is NP-hard to find the optimal linear program (circuit).

Unless P $=$ NP there exists no ϵ-approximation scheme.

Optimizing the linear components

SDU

[B.,Matthews,Peralta 2013]

It is NP-hard to find the optimal linear program (circuit).

Unless P $=$ NP there exists no ϵ-approximation scheme.

So our problem is intractable.

Optimizing the linear components

[B.,Matthews,Peralta 2013]

It is NP-hard to find the optimal linear program (circuit).

Unless P $=$ NP there exists no ϵ-approximation scheme.

So our problem is intractable.

Use heuristics.

Optimizing the linear components

[B.,Matthews,Peralta 2013]

It is NP-hard to find the optimal linear program (circuit).

Unless P = NP there exists no ϵ-approximation scheme.

So our problem is intractable.

Use heuristics.

Modify Paar's greedy heuristic to maintain feasibility for required max depth (given input depths).

Optimizing the linear components

[B.,Matthews,Peralta 2013]

It is NP-hard to find the optimal linear program (circuit).

Unless P $=$ NP there exists no ϵ-approximation scheme.

So our problem is intractable.

Use heuristics.

Modify Paar's greedy heuristic to maintain feasibility for required max depth (given input depths).

Allow some cancellation, using preprocessing.

Other results (polynomial multiplication)

- Multiplication of degree 9 polynomials over $GF(2)$:
  Starting from Bernstein's result, obtained same size, 155, but reduced depth from 9 to 6.

Other results (polynomial multiplication)

- Multiplication of degree 9 polynomials over $GF(2)$:
  Starting from Bernstein's result, obtained same size, 155, but reduced depth from 9 to 6.
  Cenk, Hasan 2015 — 155 gates, but depth 8.

Other results (polynomial multiplication)

- Multiplication of degree 9 polynomials over $GF(2)$:
  Starting from Bernstein's result, obtained same size, 155, but reduced depth from 9 to 6.
  Cenk, Hasan 2015 — 155 gates, but depth 8.
  Find, Peralta 2016 — 154 gates, and depth 9.
  Using the Find-Peralta nonlinear component, we achieved 154 gates in depth 7.

Other results (polynomial multiplication)

SDU

- Multiplication of degree 9 polynomials over $GF(2)$:
  Starting from Bernstein's result, obtained same size, 155, but reduced depth from 9 to 6.
  Cenk, Hasan 2015 — 155 gates, but depth 8.
  Find, Peralta 2016 — 154 gates, and depth 9.
  Using the Find-Peralta nonlinear component, we achieved 154 gates in depth 7.
- Multiplication of degree 12 polynomials over $GF(2)$:
  Starting from Bernstein's result, improved from 256 gates and depth 9 to 255 gates and depth 8.

Other results (polynomial multiplication)

- Multiplication of degree 9 polynomials over $GF(2)$:
  Starting from Bernstein's result, obtained same size, 155, but reduced depth from 9 to 6.
  Cenk, Hasan 2015 — 155 gates, but depth 8.
  Find, Peralta 2016 — 154 gates, and depth 9.
  Using the Find-Peralta nonlinear component, we achieved 154 gates in depth 7.
- Multiplication of degree 12 polynomials over $GF(2)$:
  Starting from Bernstein's result, improved from 256 gates and depth 9 to 255 gates and depth 8.
  Cenk, Hasan 2015 — Also 255 gates and depth 8.

Other results (multiplication in $GF(2^n)$)

- Multiplication in $GF(2^8)$:
  Improved a result with 117 gates and depth 7 to 106 gates and depth 6.

Other results (multiplication in $GF(2^n)$)

- Multiplication in $GF(2^8)$:
  Improved a result with 117 gates and depth 7 to 106 gates and depth 6.
  Former result from Circuit Minimization Work:
  http://cs-www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html

Other results (multiplication in $GF(2^n)$)

- Multiplication in $GF(2^8)$:
  Improved a result with 117 gates and depth 7 to 106 gates and depth 6.
  Former result from Circuit Minimization Work:
  http://cs-www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html
- Multiplication in $GF(2^{16})$:
  374 gates and depth 8

Other results (multiplication in $GF(2^n)$)

- Multiplication in $GF(2^8)$:
  Improved a result with 117 gates and depth 7 to 106 gates and depth 6.
  Former result from Circuit Minimization Work:
  http://cs-www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html
- Multiplication in $GF(2^{16})$:
  374 gates and depth 8
  Used in a 16-bit S-box from [Kelly,Kaminsky,Kurdziel,Lukowiak,Radziszowski 2015]
  "Customizable spone-based authenticated encryption using 16-bit S-boxes"
  Reduced 1382 gates to 462.

Thank you for your attention.