ARITH 2024

On the Systematic Creation of Faithfully Rounded Commutative Truncated Booth Multipliers

June 12, 2024

Theo Drane, Samuel Coward
Mertcan Temel
Joe Leslie-Hurd

Intel Graphics Numerical Hardware Group
Advanced Architecture Design Group
Design Engineering Group

Motivation

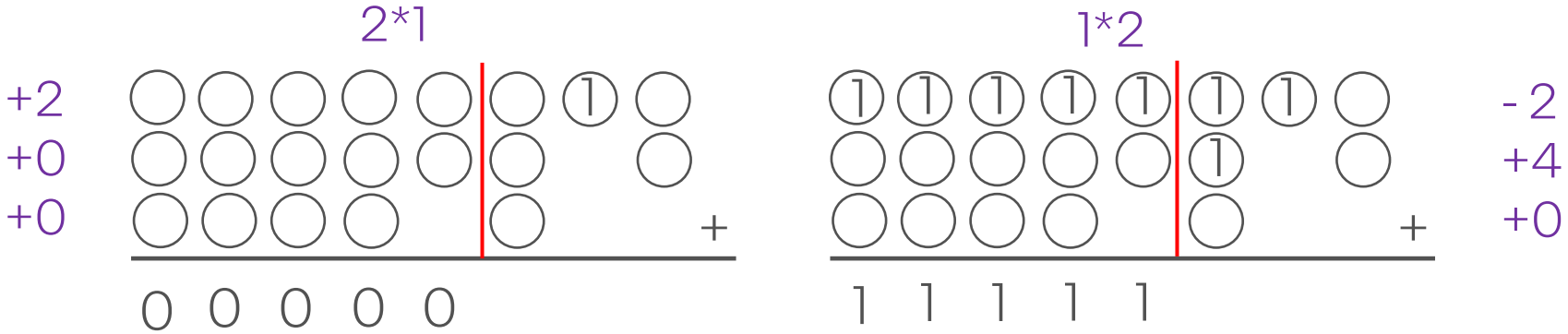
Floating-Point Dot Product Hardware: $a \times b + c \times d + e \times f + g$

Designed for low area and low accuracy

Months later a compiler optimisation discovered:

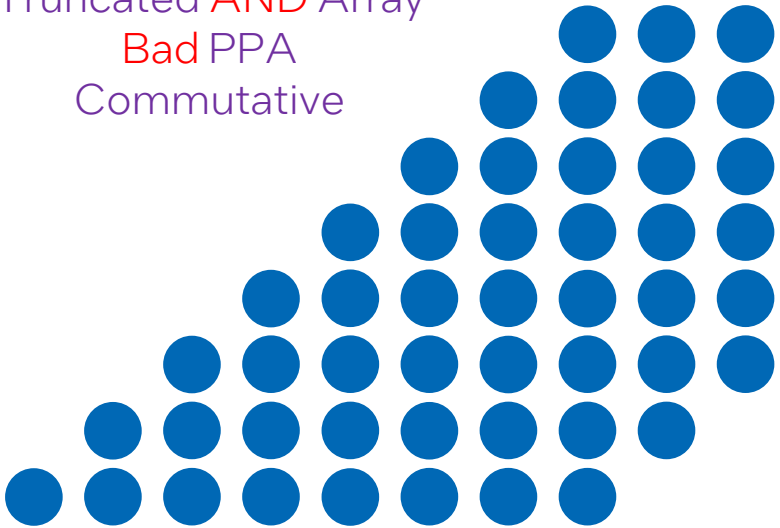
$a \times b + c \times d + e \times f + g \neq \mathrm{b} \times a + c \times d + e \times f + g$

Cause: Truncated Booth arrays



Motivation

Truncated AND Array
Bad PPA
Commutative



Truncated Booth Array
Good PPA
Not Commutative

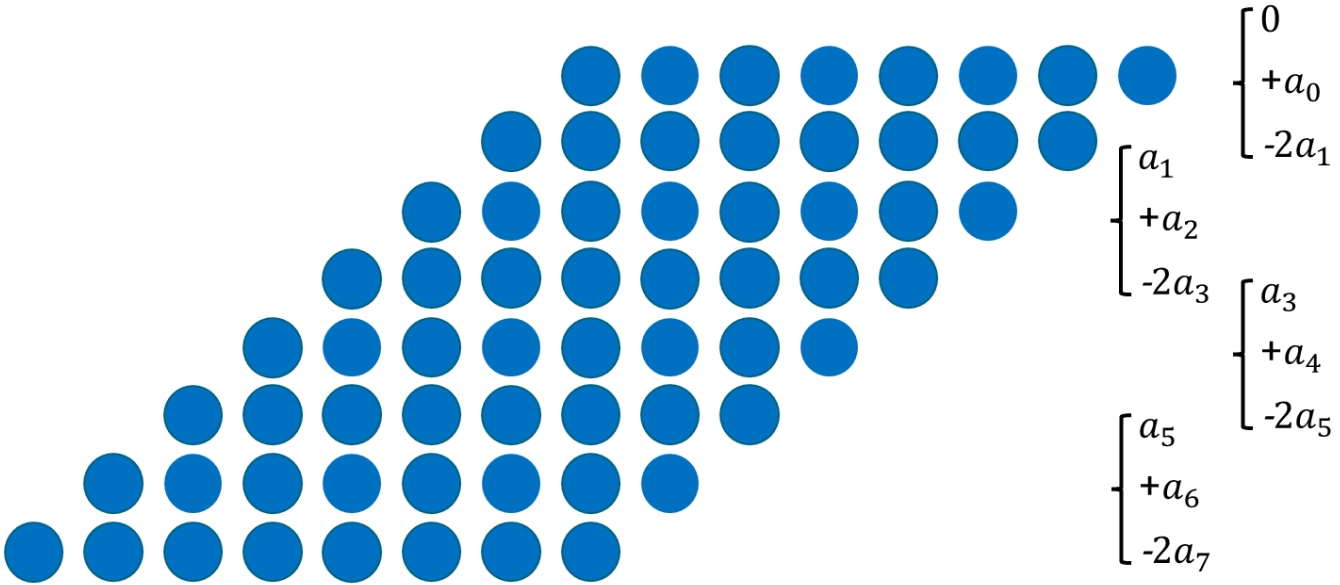
We want the best of both worlds:
Commutative
Truncated
Booth

Error:
Controlled => Faithfully Rounded
Exploited => Maximal HW Benefit

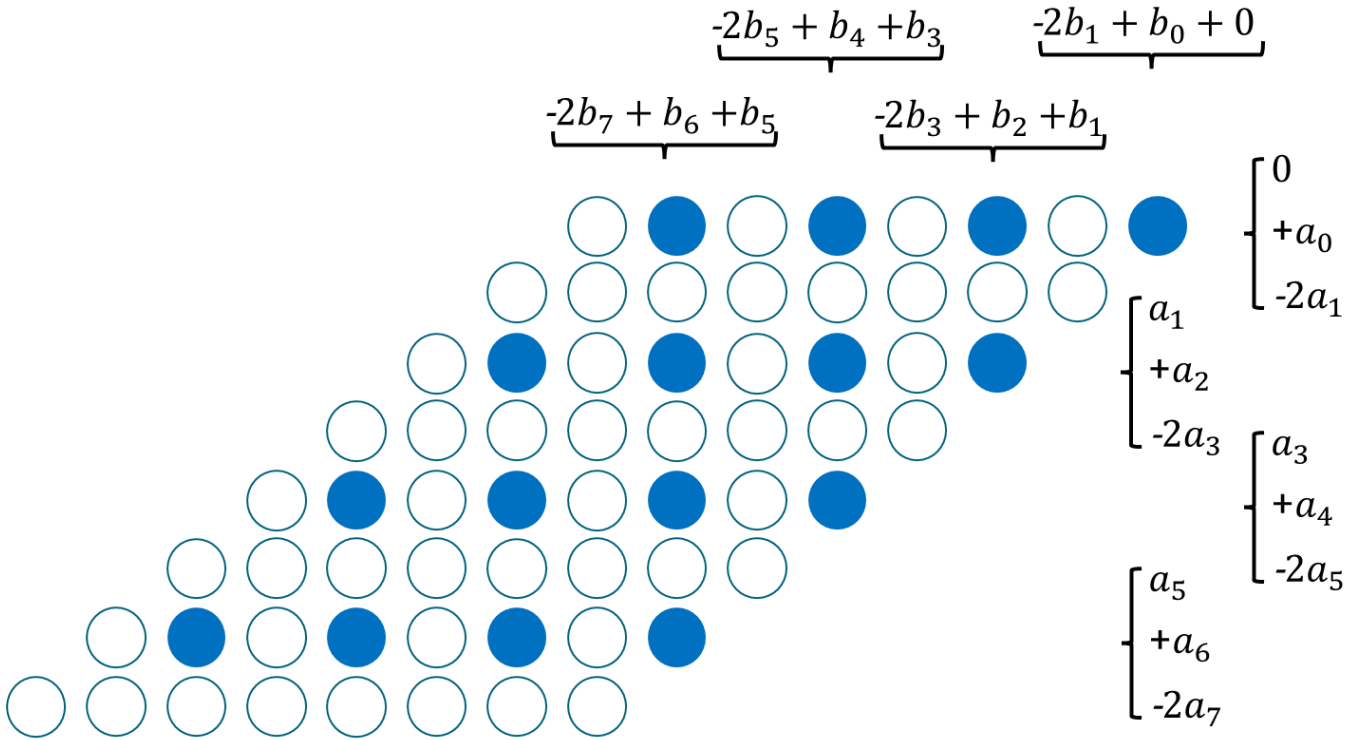
Delivery:
Formally Verified RTL

Approach

Commutativity Solution: Booth Encode Both Inputs

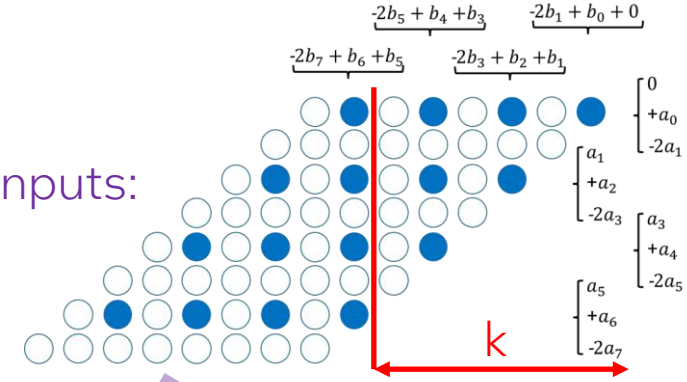


Commutativity Solution: Booth Encode Both Inputs

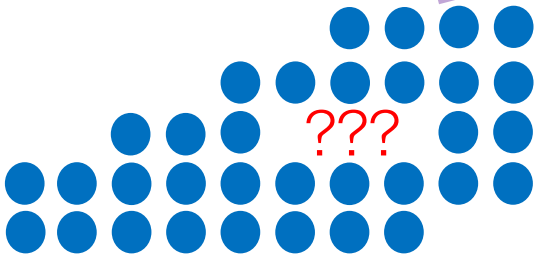


Commutativity Solution: Booth Encode Both Inputs

Truncate after Booth encoding both inputs:



Turn into binary array



A) Commutative Truncated Array Design

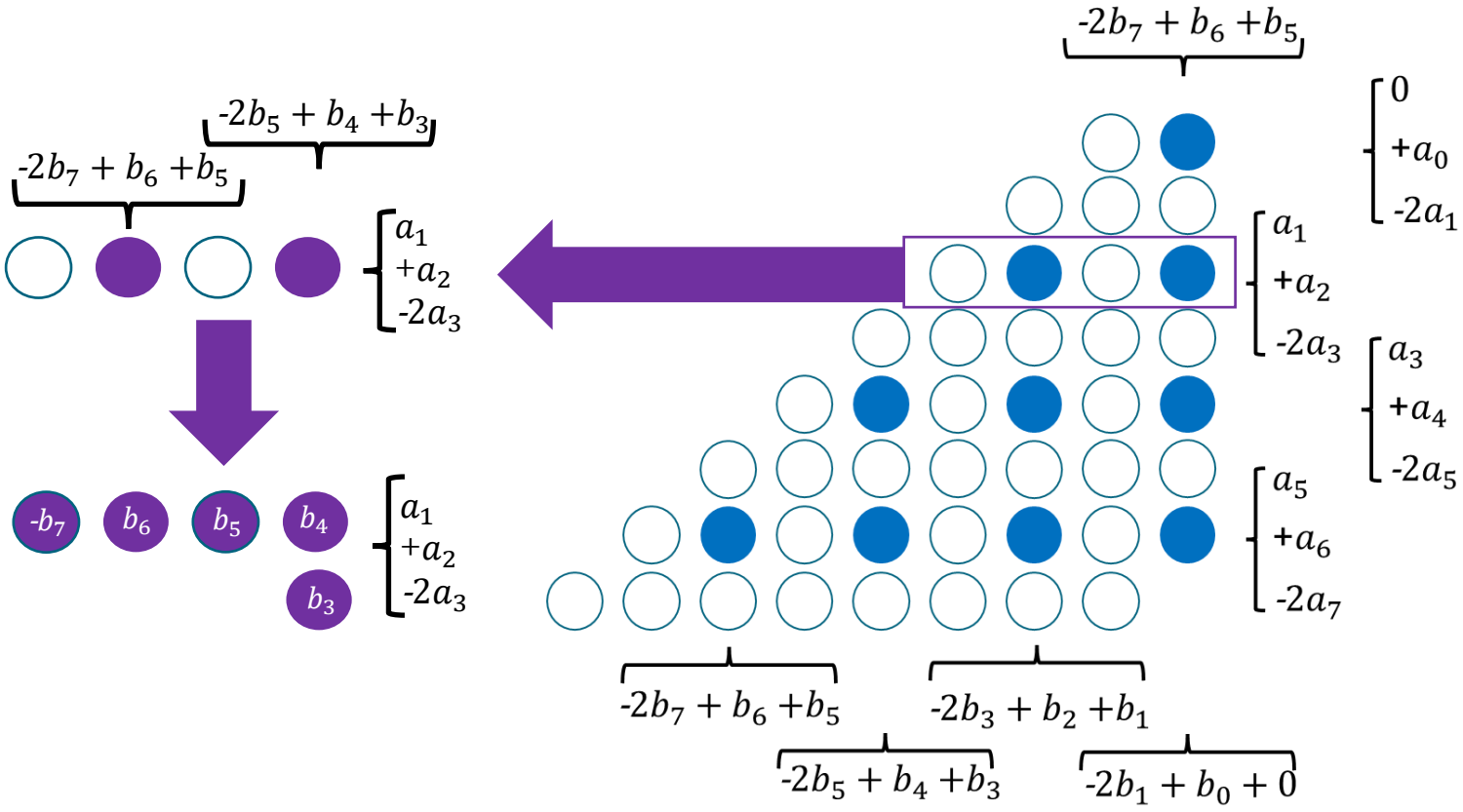
Find extremal values

???

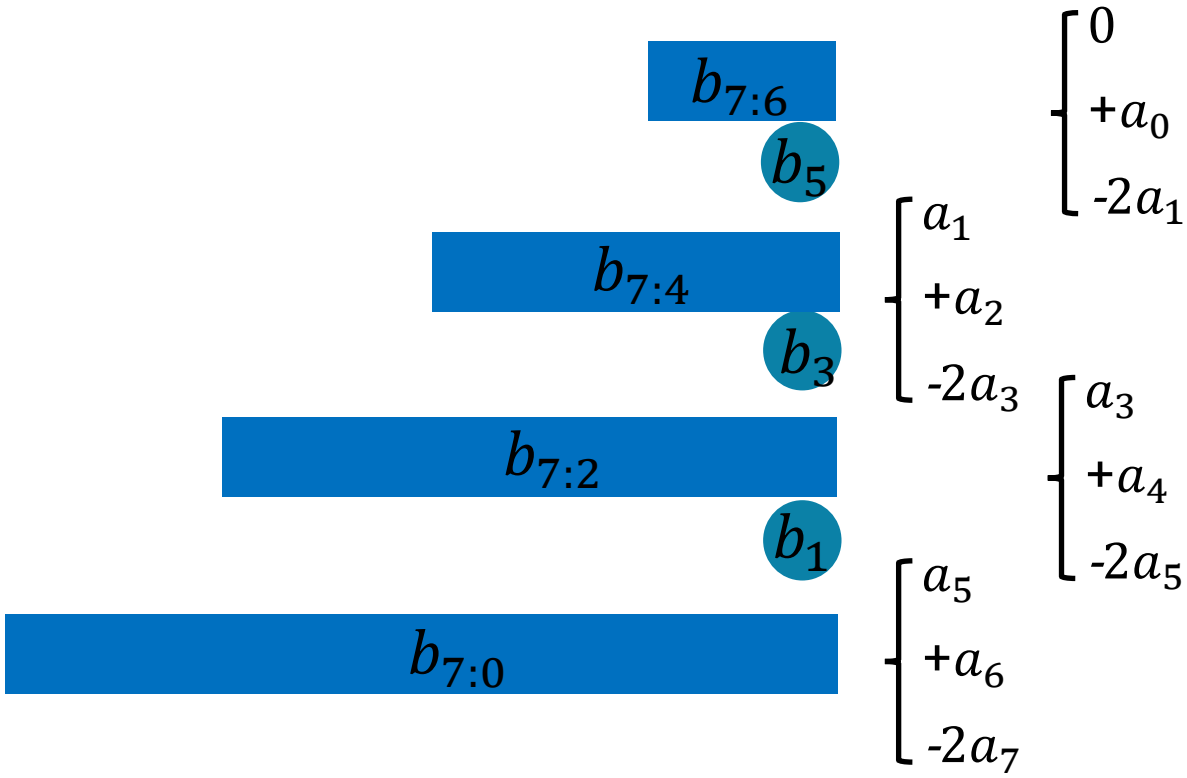
B) Commutative Truncated Array Error

A) Commutative Truncated Array Design

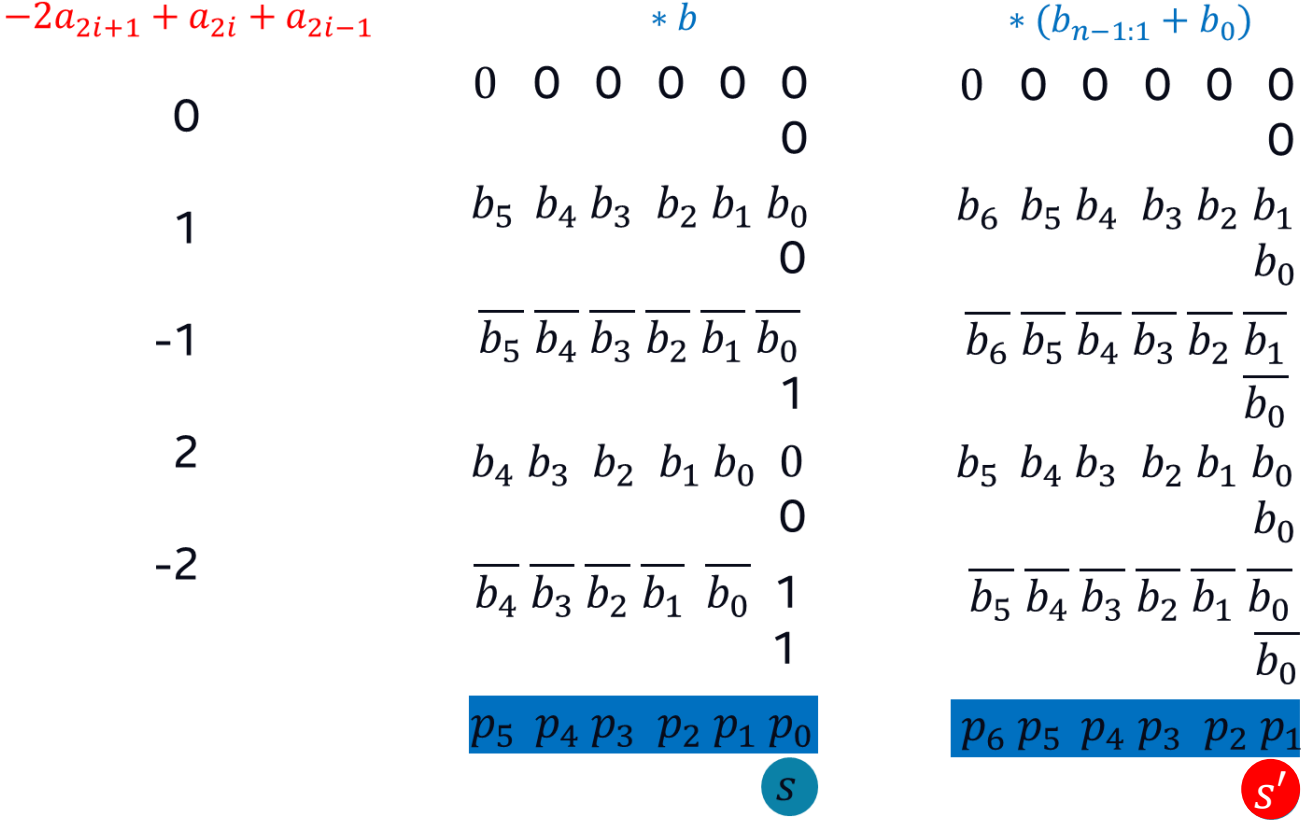
Commutativity Truncated Array Design



Commutativity Truncated Array Design



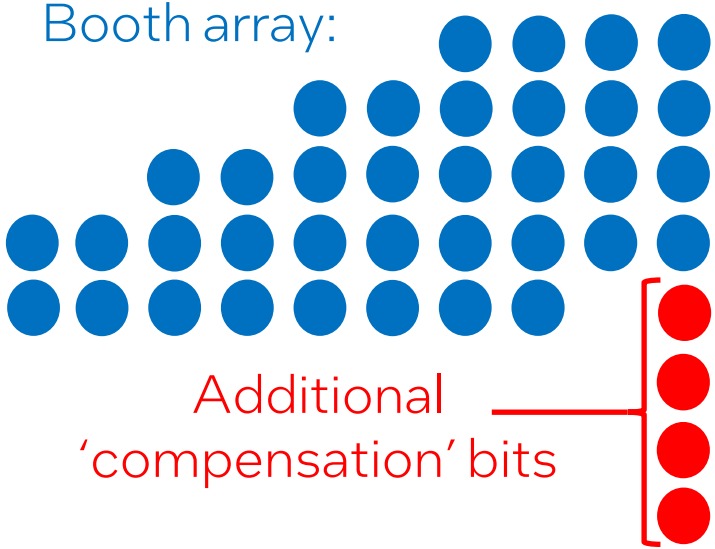
Commutativity Truncated Array Design



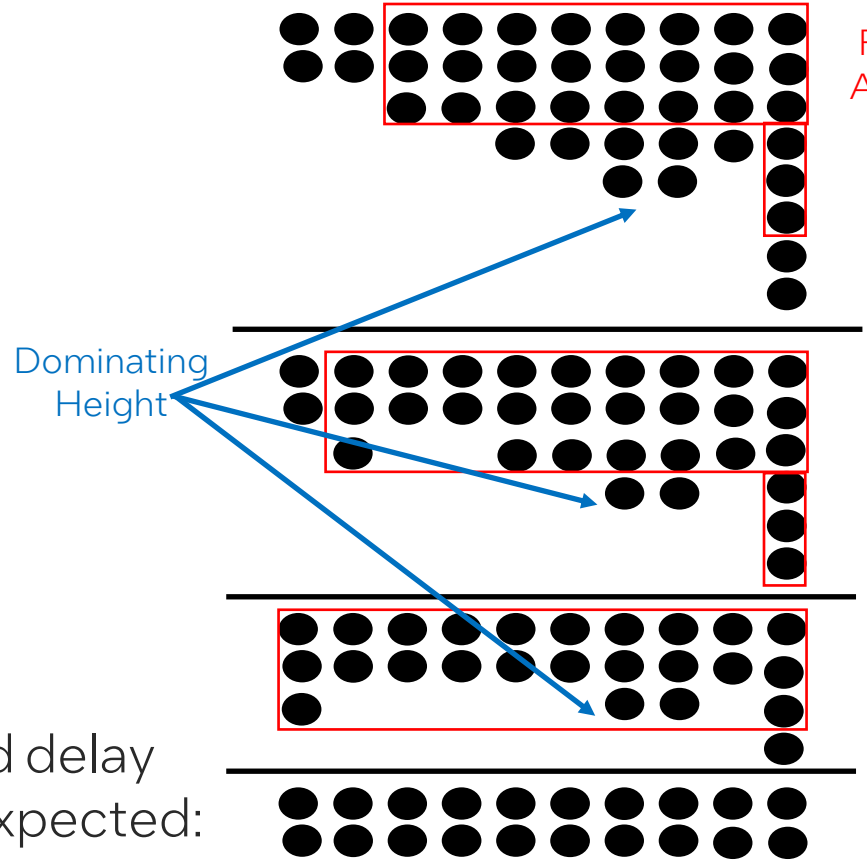
$$s' = (a_{2i+1} \oplus b_0)((a_{2i+1} \oplus a_{2i}) \vee (a_{2i} \oplus a_{2i-1}))$$

Commutativity Truncated Array Design

Standard truncated Booth array:



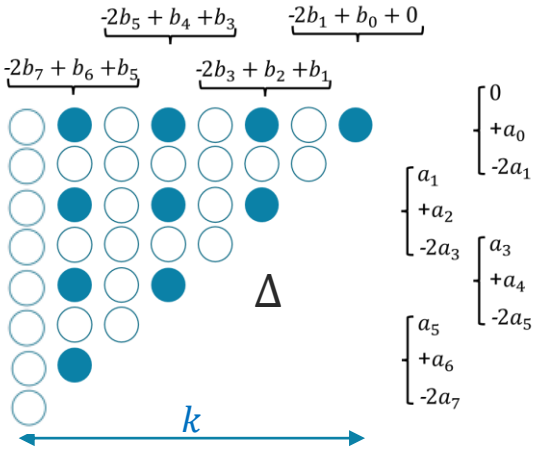
Additional ‘compensation’ bits



Limited delay impact expected:

B) Commutative Truncated Array Error

Error Analysis



Hexadecimal Summation Helper Functions:

$X_n =$	$Y_n =$	$Z_n =$	$W_n =$
$\overbrace{222..222}^{n}$	$\overbrace{222..222}^{n}$	$\overbrace{000..000}^{n}$	$\overbrace{444..444}^{n}$
	+444..444	+444..444	+444..444
+444..440	+444..440	+444..440	+444..440
+444..440	+444..440	+444..440	+444..440
+444..400	+444..400	+444..400	+444..400
+444..400	+444..400	+444..400	+444..400
...	...	...	...
+440..000	+440..000	+440..000	+440..000
+440..000	+440..000	+440..000	+440..000
+400..000	+400..000	+400..000	+400..000
+400..000	+400..000	+400..000	+400..000

$$\Delta = -2^{k-1}(b_{k-1}a_0 + b_{k-3}a_2) + 2^{k-2}(b_{k-2}a_0 + b_{k-4}a_2) + 4(b_2a_0 + b_0a_2) \ldots$$
$$= -2^{k-1}(a_{k-1}b_0 + a_{k-3}b_2) + 2^{k-2}(a_{k-2}b_0 + a_{k-4}b_2) + 4(a_2b_0 + a_0b_2) \ldots$$

Extremal values occur when either:

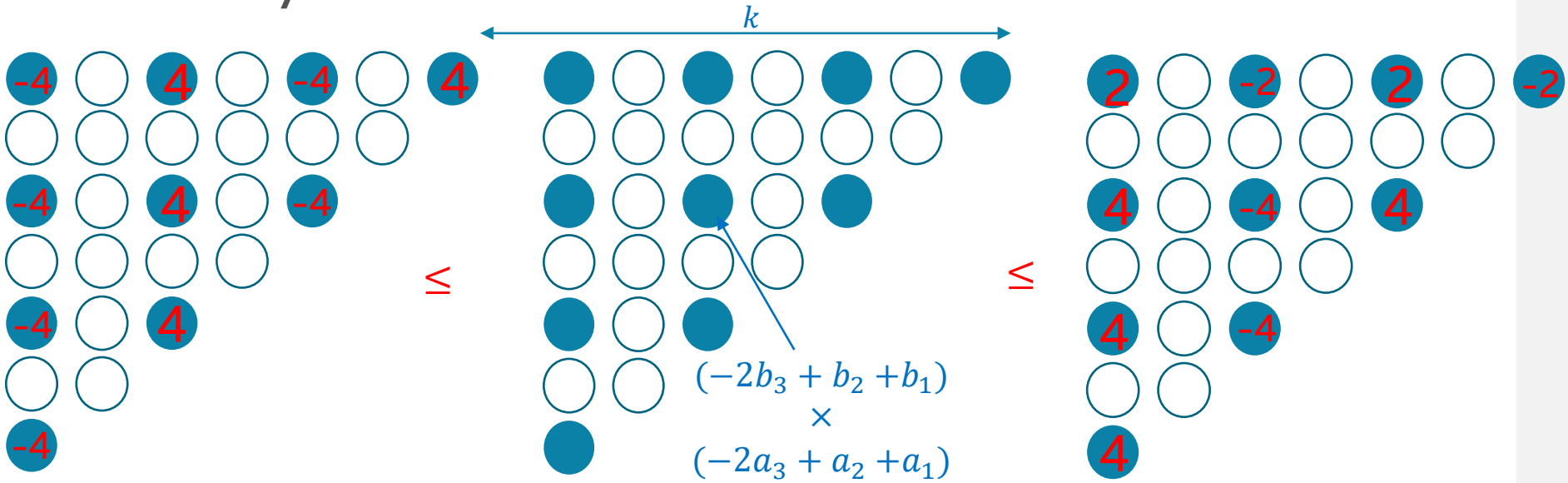
$a_i = a_{i-2}$ and $b_j = b_{j-2}$

a	b	Δ
111...111	111...111	1
111...111	101...010	$\frac{2^k+2}{3}$
111...111	010...101	$-\frac{2^k-1}{3}$
101...010	101...010	$\frac{1}{18}((3k+16)2^k+16)$
101...010	010...101	$-k2^{k-2} - \frac{1}{3}(2^k-1)$
010...101	010...101	$\frac{1}{18}((3k-2)2^k+2)$
000...000	000...000	0

$a_i \neq a_{i-2}$ and $b_j \neq b_{j-2}$

$k/2$	a	b	Δ
even	10011001...1001	01100110...0110	$4Y_{\frac{k}{4}} - X_{\frac{k}{4}}$
odd	10011001...10	10011001...10	$Z_{\frac{k+2}{4}} - 4W_{\frac{k-2}{4}}$
even	01100110...0110	01100110...0110	$Z_{\frac{k}{4}} - 4W_{\frac{k}{4}}$
odd	10011001...10	01100110...01	$-X_{\frac{k+2}{4}} + 4Y_{\frac{k-2}{4}}$

Error Analysis



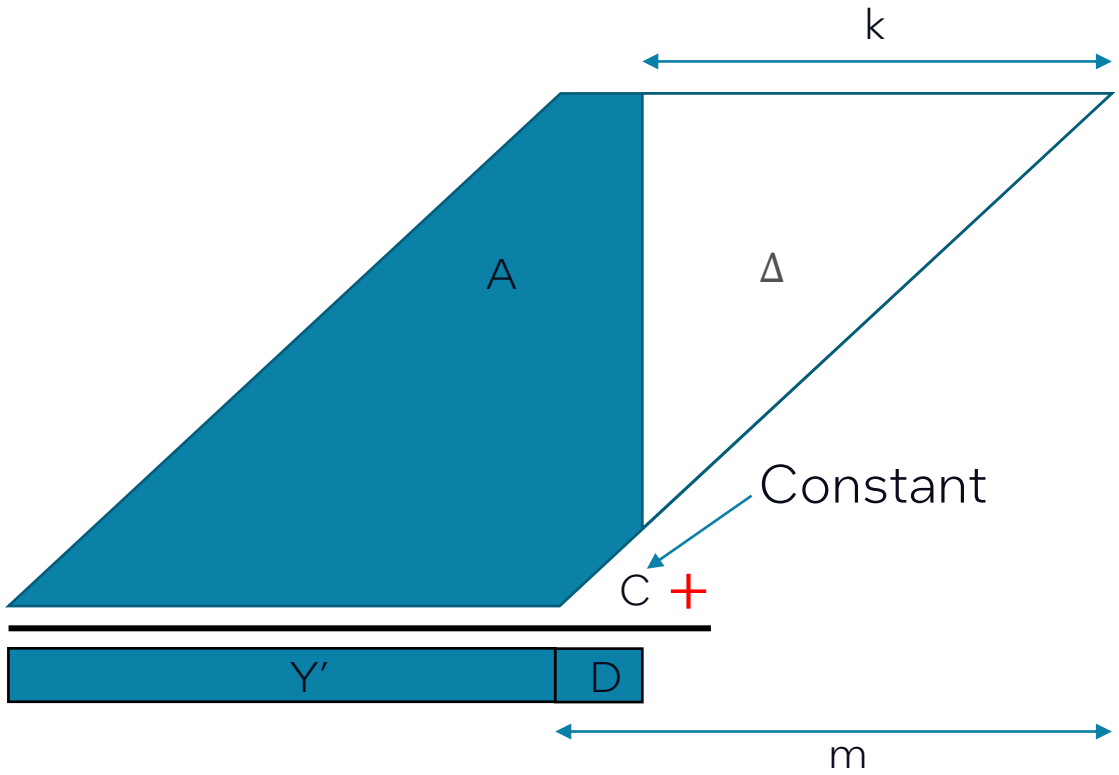
$$-\frac{1}{25}\left(2^k\left(10k + 5(-1)^{k/2} - 1\right) - 5 + (-1)^{k/2}\right) \leq \Delta \leq \frac{1}{25}\left(2^k\left(10k - 5(-1)^{k/2} - 1\right) + 5 + (-1)^{k/2}\right)$$

$\min(\Delta)$ $\max(\Delta)$

(These bounds are tight, and we have worst case inputs.)

Faithfully Rounded Commutative Truncated Array

Faithful Rounding



Faithful rounding:

$$-2^m < (A+\Delta) - Y' < 2^m$$

$$-2^m < (A+\Delta) - (A+C-D) < 2^m$$

$$-2^m < C - \Delta - D < 2^m$$

D can take any value in $[0, 2^m - 2^k]$, hence

$$C - 2^m < \Delta < C + 2^k$$

$$\max(\Delta) - 2^k < C < \min(\Delta) + 2^m$$

C must be a multiple of 2^k:

$$\left\lfloor \frac{\max(\Delta)}{2^k} \right\rfloor \leq \left\lfloor \frac{\min(\Delta)}{2^k} \right\rfloor + 2^{m-k}$$

Maximal Error Exploitation

Optimal (Maximal) Truncation:

$$k^* = \max_{even\ k} \left(k \leq 5 * 2^{\mathrm{m-k-2}}\right)$$

Associated constant C range:

$$\frac{2k^* - 5 - (-1)^{k^*/2}}{5} \leq C^* \leq \frac{5 * 2^{m-k^*} - 2k^* - (-1)^{k^*/2}}{5}$$

m	k*	min C*	max C*	C*
8	4	1	14	1
16	12	4	11	4
24	20	7	7	7
32	26	10	53	16
53	46	18	109	32
64	58	23	41	32

Conclusion

Commutative
Truncated
Booth
Faithfully Rounded
Maximal HW Benefit
Formally Verified
RTL

Unsigned n bit multiplication Faithful Rounding to column m

```
module gmd_int_mult_1ulp
 #(parameter n = 24,
   parameter m = 23)
  (
   input  [  n  -1:0] a,
   input  [  n  -1:0] b,
   output [2*n-m-1:0] y
   );
```

Array Creation

```
always_comb
   begin
      aext = {1'b0,a,1'b0};
      for (int i = 0; i < ((n+1)/2); i++)
         begin
            A[i] = aext[2*i+2];
            B[i] = aext[2*i+1];
            C[i] = aext[2*i  ];
         end
   end

always_comb s              = A & ~(B & C);
always_comb bxorc          =     B ^ C ;
always_comb axorb          = A ^   B   ;
always_comb x2             = axorb & ~bxorc;
always_comb axorb_or_bxorc = axorb |  bxorc;

always_comb
   begin
      bext = {1'b0,b,1'b0};
      for (int i = 0; i < ((n+1)/2); i++)
         bxors[i] = bext ^ (n+2{s[i]});
   end

always_comb
   begin
      for (int i = 0; i < ((n+1)/2); i++)
         pp[i] = ((n+1{x2[i]}) & bxors[i][n:0]) | ((n+1{bxorc[i]}) & bxors[i][n+1:1]);
   end

always_comb pp_final = (n{a[n-1]}) & b;

always_comb
   begin
      for (int i = 0; i < (k/2); i++)
         comp[i] = bxors[i][k-2*i] & (axorb[i] | bxorc[i]);
   end

always_comb
   begin
      for (int i = 0; i < ((n+1)/2); i++)
         signext[i] = axorb_or_bxorc[i] & bxors[i][n+1];
   end

always_comb array[0] = {~signext[0],signext[0],signext[0],pp[0]};

generate
   for (gi=1; gi<(n/2); gi++)
      assign array[gi] = {1'b1,~signext[gi],pp[gi],1'b0,s[gi-1],{2*(gi-1){1'b0}}};
endgenerate

generate
   if (n%2==0)
      assign array[(n/2)] = {                      pp_final,1'b0,s[(n/2)-1],{2*((n/2)-1){1'b0}}};
   else
      assign array[(n/2)] = {1'b1,~signext[(n/2)],pp[(n/2)],1'b0,s[(n/2)-1],{2*((n/2)-1){1'b0}}};
endgenerate

generate
   for (gi=0; gi<(k/2); gi++)
      assign array[(n/2)+1+gi] = comp[gi]<<k;
endgenerate

always_comb array[(n/2)+(k/2)+1] = constant<<k;
```

Create compensation bits

Add compensation bits & constant To array

Compute k and C

```
function int computek(input int m);
   int k = 0;
   for (int i = 0; i < m; i = i+2)
         if (i-1<(5*(1<<(m-i-2))))
            k=i;
   return k;
endfunction

localparam k        = computek(m);
localparam minconst = (2*k-2+2*((k/2)%2)) / 5;
localparam maxconst = ((5*(1<<(n-k)))-2*k-1+2*((k/2)%2)) / 5;

localparam AxorB       = minconst ^ maxconst;
localparam difflength  = $clog2(AxorB+1);
localparam Alsbs       = minconst % (1<<difflength);
localparam msbs        = minconst - Alsbs;
localparam Alsbslength = $clog2(Alsbs+1);
localparam lsbs_pre    = 1 << (Alsbslength-1);
localparam lsbs        = (Alsbs==lsbs_pre)? lsbs_pre : lsbs_pre<<1;
localparam constant    = msbs + lsbs;
```

Array Reduction

```
generate
   for (gi=0; gi<(n/2)+(k/2)+2; gi++)
      begin: array_i
         always_comb
            begin
               if (gi==0)
                  sum[ 0] =              array[ 0][2*n-1:k];
               else
                  sum[gi] = sum[gi-1] + array[gi][2*n-1:k];
            end
      end
endgenerate

assign y = sum[(n/2)+(k/2)+1][2*n-k-1:m-k];
```

Commutative
Truncated
Booth
Faithfully Rounded
Maximal HW Benefit
Formally Verified
RTL

```
mult = a[n-1:0]*b[n-1:0]
lsbs = mult[  m-1:0]
msbs = mult[2*n-1:m]

lemma (lsbs==0)? y==msbs : 0<=y-msbs<=1
```

Proofs are performed using a multiplier rewriter implemented as an ACL2 clause processor with some help from a SAT Solver

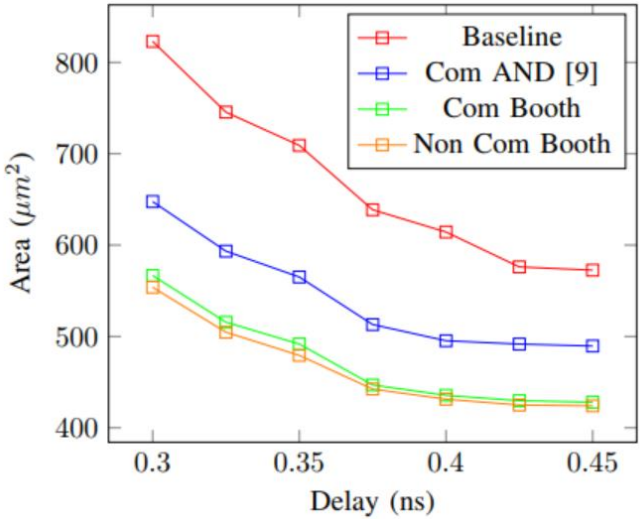
[Sound and Automated Verification of Real-World RTL Multipliers]

n	m	Faithful Rounding Proof (seconds)	Commutativity Proof (seconds)
4	4	3	0.6
6	6	4	0.6
8	7	4	0.6
16	5	4	0.7
16	16	7	0.7
24	16	11	0.7
32	16	22	0.9
32	32	45	0.9
36	32	52	1.0
36	36	120	1.0
42	42	728	1.1
64	64	--	2.0
128	128	--	8.0

Synthesis Results

n	Delay (ns)	Baseline Area (μm^2)	Baseline Power (μW)	Truncated AND [9] Area (μm^2)	Truncated AND [9] Power (μW)	Commutative Truncated Booth Area (μm^2)	Commutative Truncated Booth Power (μW)
16	0.175	74.0	450	**64.4** (-13.0%)	**319** (-29.2%)	69.3 (- 6.4%)	389 (-13.7%)
	0.2	56.1	346	**46.4** (-17.4%)	**252** (-27.3%)	48.3 (-13.9%)	286 (-17.5%)
	0.225	51.7	309	**40.9** (-20.8%)	**215** (-30.4%)	43.6 (-15.8%)	252 (-18.5%)
	0.25	47.5	283	**37.1** (-21.9%)	**198** (-30.1%)	39.7 (-16.3%)	226 (-20.1%)
24	0.225	127.7	815	108.6 (-14.9%)	622 (-23.7%)	**99.9** (-21.8%)	**600** (-26.4%)
	0.25	113.4	705	101.5 (-10.4%)	575 (-18.4%)	**88.7** (-21.8%)	**516** (-26.8%)
	0.275	108.0	653	93.5 (-13.4%)	502 (-23.2%)	**84.9** (-21.3%)	**498** (-23.8%)
	0.3	99.2	595	83.8 (-15.6%)	466 (-21.8%)	**77.3** (-22.1%)	**447** (-25.0%)
32	0.275	193.7	1225	161.3 (-16.7%)	897 (-26.7%)	**156.2** (-19.4%)	**914** (-25.4%)
	0.3	171.5	1097	156.3 (- 8.8%)	861 (-21.5%)	**148.0** (-13.7%)	**862** (-21.4%)
	0.325	164.3	1024	139.7 (-15.0%)	787 (-23.1%)	**134.2** (-18.3%)	**768** (-25.0%)
	0.35	152.9	945	137.2 (-10.3%)	772 (-18.3%)	**130.7** (-14.5%)	**748** (-20.8%)
64	0.3	823.0	5548	647.6 (-21.3%)	3808 (-31.4%)	**566.8** (-31.1%)	**3527** (-36.4%)
	0.325	745.5	4886	593.3 (-20.4%)	3403 (-30.4%)	**515.6** (-30.8%)	**3073** (-37.1%)
	0.35	709.2	4558	565.1 (-20.3%)	3165 (-30.6%)	**491.7** (-30.7%)	**2846** (-37.6%)
	0.375	638.6	4187	513.1 (-19.7%)	2936 (-29.9%)	**446.9** (-30.0%)	**2593** (-38.1%)

n = 64



Round towards zero

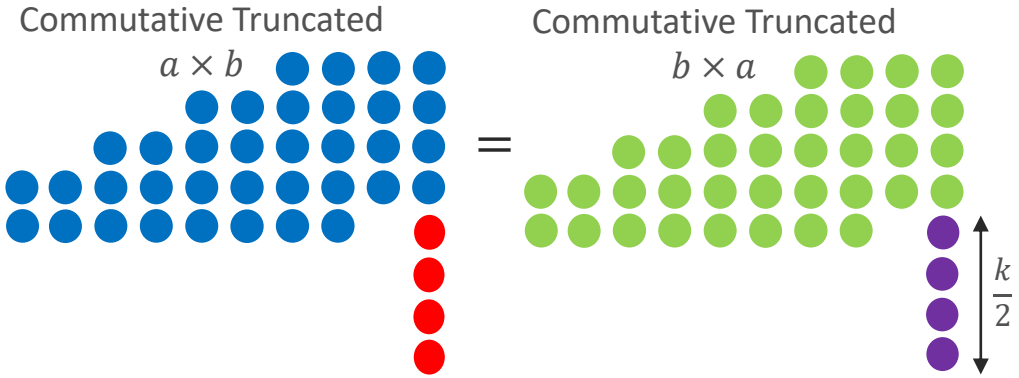
Commutative Truncated AND Array – up to 21% smaller

Commutative Truncated Booth Array – up to 31% smaller

Non-Commutative Truncated Booth Array – up to 33% smaller

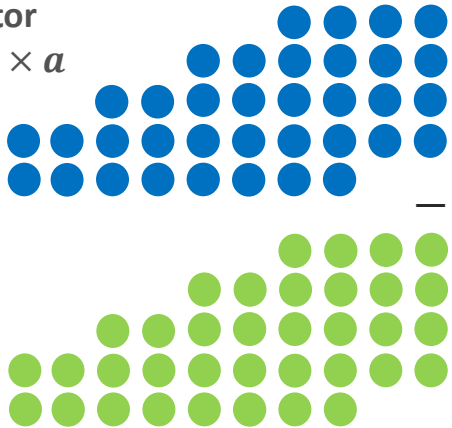
Epilogue

Commutator



Commutator

$= a \times b - b \times a$



$=$

$(a_1 \oplus b_{k-1})((a_1 \oplus a_0) \vee (a_0 \oplus 0))$

$+(a_3 \oplus b_{k-3})((a_3 \oplus a_2) \vee (a_2 \oplus a_1))$

$\dots$

$+(a_{k-1} \oplus b_1)((a_{k-1} \oplus a_{k-2}) \vee (a_{k-2} \oplus a_{k-3}))$

$-(b_1 \oplus a_{k-1})((b_1 \oplus b_0) \vee (b_0 \oplus 0))$

$-(b_3 \oplus a_{k-3})((b_3 \oplus b_2) \vee (b_2 \oplus b_1))$

$\dots$

$-(b_{k-1} \oplus a_1)((b_{k-1} \oplus b_{k-2}) \vee (b_{k-2} \oplus b_{k-3}))$

$0 \times 1010 \dots 1010$

$\in \left[-\frac{k}{2}, \frac{k}{2}\right]$

$1010 \dots 1010 \times 0$

ARITH 2024

On the Systematic Creation of Faithfully Rounded Commutative Truncated Booth Multipliers

June 12, 2024

Theo Drane, Samuel Coward
Mertcan Temel
Joe Leslie-Hurd

Intel Graphics Numerical Hardware Group
Advanced Architecture Design Group
Design Engineering Group