HIGH DENSITY AND PERFORMANCE MULTIPLICATION FOR FPGA

Martin Langhammer and Gregg Baeckler

June 2018

Overview

AI/DL/ML inference driving need for much higher FPGA arithmetic density

- And at a low precision

Known small multiplier architectures are not effective

- Logic mapping, routing mapping

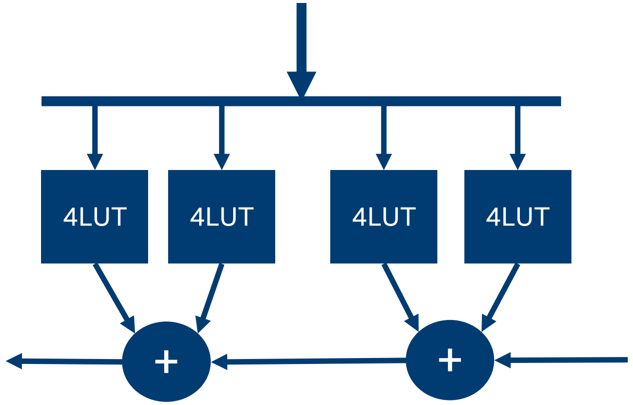
Routing micro-architectures and macro-architectures poorly understood

- Independent routing density a subset of redundant routing density

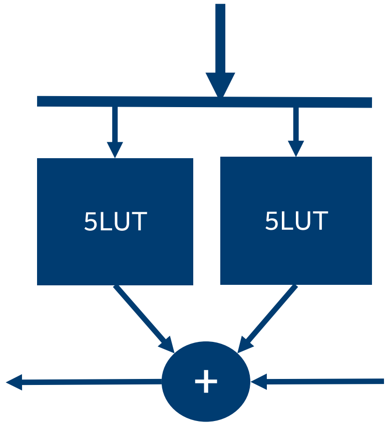
This paper shows how to balance logic and routing for maximum FPGA efficiency

FPGA Architecture

Intel



Xilinx



Recent Work

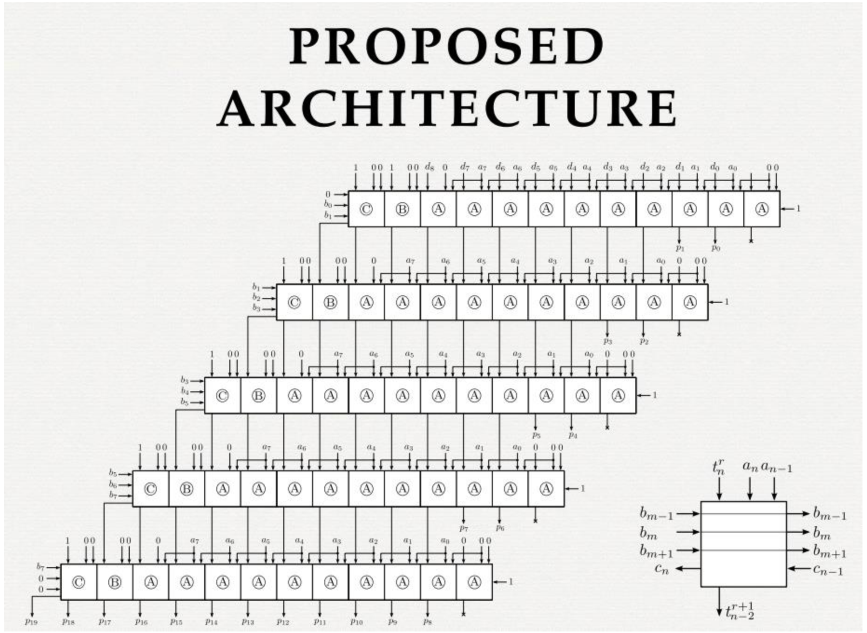
Kumm, et.al. ARITH22 (2015)

Similar work by Walters

Use Xilinx 6LUT to implement Booth's 4 level

Use embedded adder to sum previous level

Array structure has long latency



Regular and Irregular Multipliers

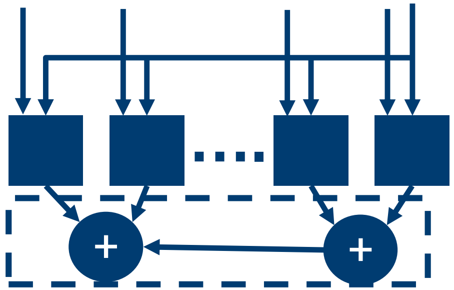
FPGA logic naturally more efficient with regular structures

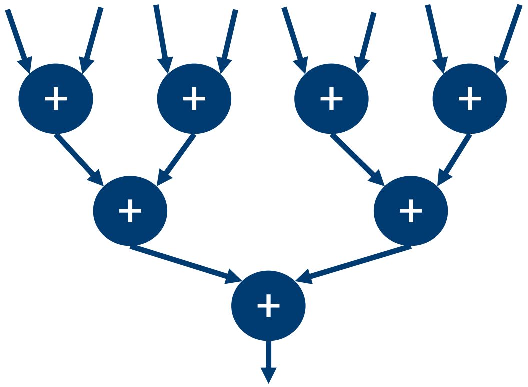
4LUT + adder can implement sum of two pencil and paper partial products

Adder tree (2 input) most efficient when powers of 2 number of inputs

Nx4, Nx8, Nx16, etc

Multiplier regularization makes irregular multipliers regular





MULTIPLIER REGULARIZATION

3x3 - A Trivial First Case

3x3 multiplier useful for AI inference

A trivial case – or is it?

Saving 1 LUT significant if you have 50K multipliers

Logic savings overshadowed by routing optimization and simplification

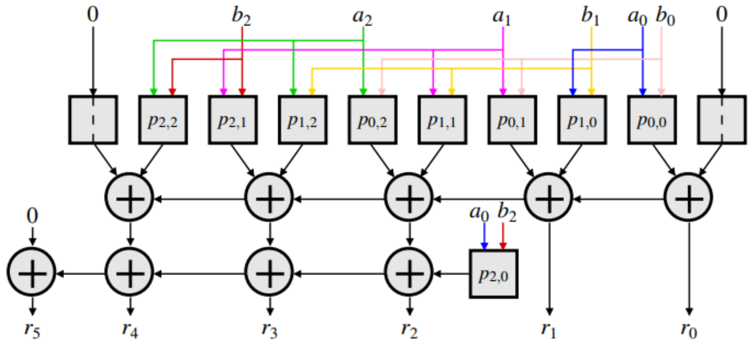
- Second level carry chain creates device placement restriction

Doesn't depend on carry chain

3 high column affects rest of multiplier

Column	5	4	3	2	1	0
PP0	0	0	0	$p_{0,2}$	$p_{0,1}$	$p_{0,0}$
PP1	0	0	$p_{1,2}$	$p_{1,1}$	$p_{1,0}$	0
PP2	0	$p_{2,2}$	$p_{2,1}$	$p_{2,0}$	0	0

Unused logic (and routing)



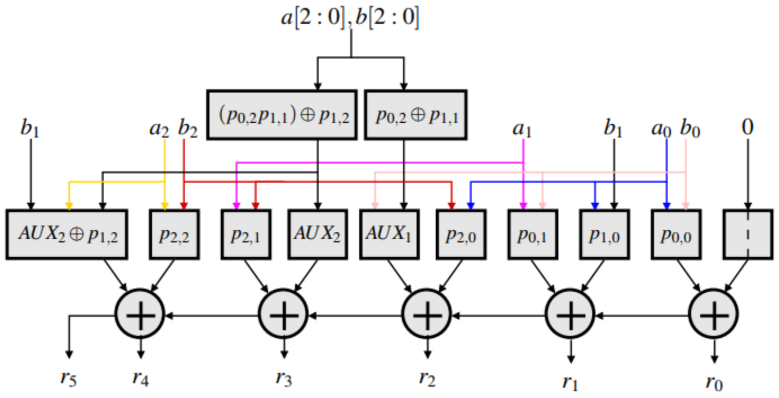
Regularized 3x3

Refactor logic in any column to more than 100% if required

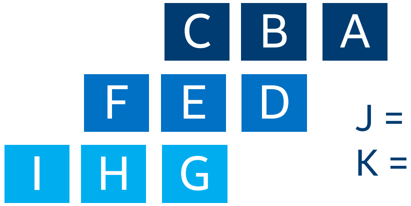
- Refactor any logic over 100% to a function of 1 bit + 100% of remaining capability

Use out of band functions for other partially used columns

Column	5	4	3	2	1	0
PP0	0	$p_{2,2}$	$p_{2,1}$	$p_{2,0}$	$p_{0,1}$	$p_{0,0}$
PP1	0	$AUX_2 \oplus p_{1,2}$	AUX_2	AUX_1	$p_{1,0}$	0



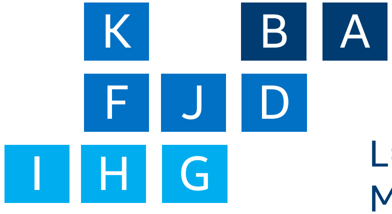
3x3 - Details



J = C XOR E
K = C AND E

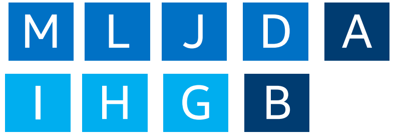
J = A2B0 XOR A1B1
K = A2B0 AND A1B1

{A2,A1,B1,B0}



L= K XOR F
M = K AND F

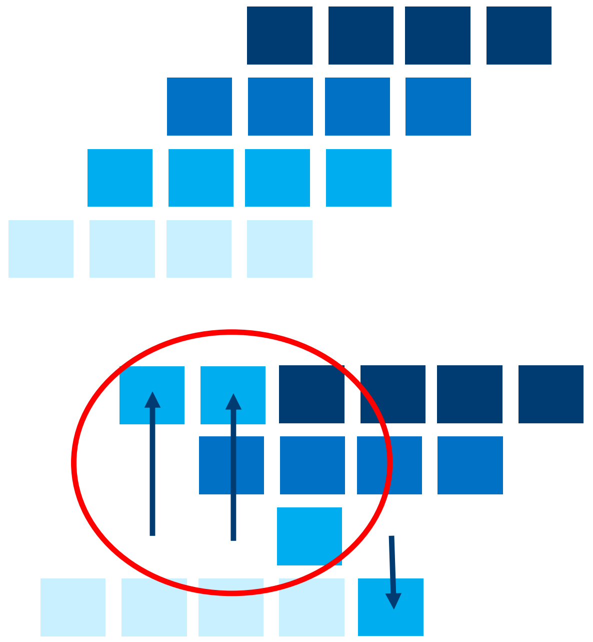
L = (A2B0 AND A1B1) XOR A2B1

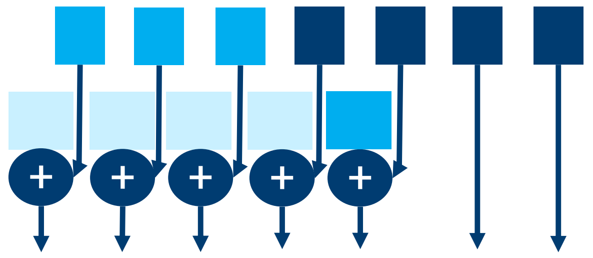


M = (A2B0 AND A1B1) AND A2B1
= (L XOR A2B1) AND A2B1

A2,B1 can be brought into ALM

4x4 Case – De-regularize then Regularize





5x5

Col.	8	7	6	5	4	3	2	1	0
PP0	0	0	0	0	$p_{0,4}$	$p_{0,3}$	$p_{0,2}$	$p_{0,1}$	$p_{0,0}$
PP1	0	0	0	$p_{1,4}$	$p_{1,3}$	$p_{1,2}$	$p_{1,1}$	$p_{1,0}$	0
PP2	0	0	$p_{2,4}$	$p_{2,3}$	$p_{2,2}$	$p_{2,1}$	$p_{2,0}$	0	0
PP3	0	$p_{3,4}$	$p_{3,3}$	$p_{3,2}$	$p_{3,1}$	$p_{3,0}$	0	0	0
PP4	$p_{4,4}$	$p_{4,3}$	$p_{4,2}$	$p_{4,1}$	$p_{4,0}$	0	0	0	0

Column	8	7	6	5	4	3	2	1	0
PP0	0	0	$p_{2,4}$	$p_{2,3}$	$p_{0,4}$	$p_{0,3}$	$p_{0,2}$	$p_{0,1}$	$p_{0,0}$
PP1	0	0	0	$p_{1,4}$	$p_{1,3}$	$p_{1,2}$	$p_{1,1}$	$p_{1,0}$	0
PP2	0	0	0	0	$p_{2,2}$	0	0	0	0

Column	8	7	6	5	4	3	2	1	0
PP3	0	$p_{3,4}$	$p_{3,3}$	$p_{3,2}$	$p_{3,1}$	$p_{3,0}$	0	0	0
PP4	$p_{4,4}$	$p_{4,3}$	$p_{4,2}$	$p_{4,1}$	$p_{4,0}$	0	0	0	0

Col.	8	7	6	5	4	3	2	1	0
PP0'	0	0	$p_{2,4}$	$p_{2,3}$	$p_{0,4}$	$p_{0,3}$	$p_{0,2}$	$p_{0,1}$	$p_{0,0}$
PP1'	0	0	$AUX_2 \oplus p_{2,3}$	AUX_2	AUX_1	$p_{1,2}$	$p_{1,1}$	$p_{1,0}$	0

Good

Column	8	7	6	5	4	3	2	1	0
PP0	0	0	$p_{2,4}$	$p_{2,3}$	$p_{0,4}$	$p_{0,3}$	$p_{0,2}$	$p_{0,1}$	$p_{0,0}$
PP1	0	0	0	$p_{1,4}$	$p_{1,3}$	$p_{1,2}$	$p_{1,1}$	$p_{1,0}$	0
PP2	0	0	0	0	$p_{2,2}$	0	0	0	0

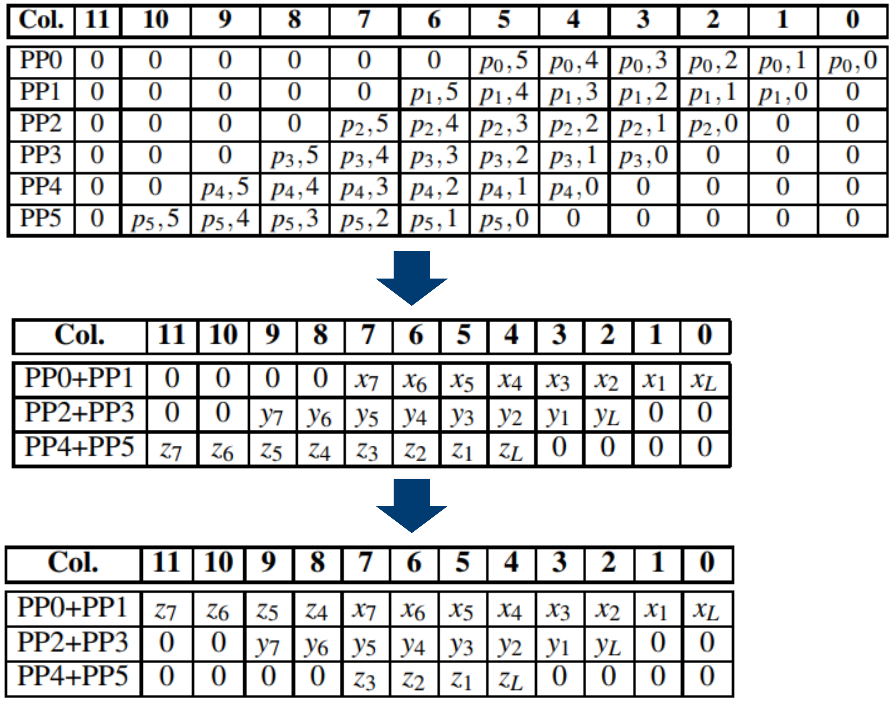
Bad

Column	8	7	6	5	4	3	2	1	0
PP0	0	0	$p_{2,4}$	$p_{2,3}$	$p_{0,4}$	$p_{0,3}$	$p_{0,2}$	$p_{0,1}$	$p_{0,0}$
PP1	0	0	0	$p_{1,4}$	$p_{1,3}$	$p_{1,2}$	$p_{1,1}$	$p_{1,0}$	0
PP2	0	0	0	0	$p_{2,2}$	0	0	0	0

TERNARY TO BINARY ADDER MAPPING

Adder Tree Simplification

- Ternary addition does not work in the general case
  - Routing density
- Ternary condition may occur for n*3 partial products
  - 6x6
  - 7x7 regularized

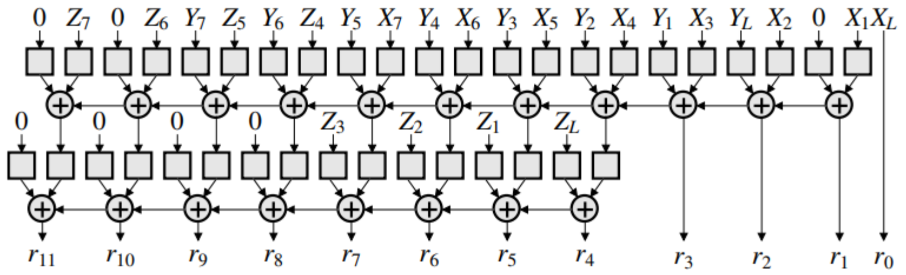


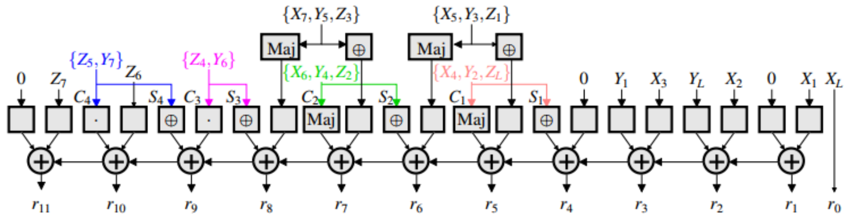
Col.	11	10	9	8	7	6	5	4	3	2	1	0
PP0	0	0	0	0	0	0	$p_0,5$	$p_0,4$	$p_0,3$	$p_0,2$	$p_0,1$	$p_0,0$
PP1	0	0	0	0	0	$p_1,5$	$p_1,4$	$p_1,3$	$p_1,2$	$p_1,1$	$p_1,0$	0
PP2	0	0	0	0	$p_2,5$	$p_2,4$	$p_2,3$	$p_2,2$	$p_2,1$	$p_2,0$	0	0
PP3	0	0	0	$p_3,5$	$p_3,4$	$p_3,3$	$p_3,2$	$p_3,1$	$p_3,0$	0	0	0
PP4	0	0	$p_4,5$	$p_4,4$	$p_4,3$	$p_4,2$	$p_4,1$	$p_4,0$	0	0	0	0
PP5	0	$p_5,5$	$p_5,4$	$p_5,3$	$p_5,2$	$p_5,1$	$p_5,0$	0	0	0	0	0

Col.	11	10	9	8	7	6	5	4	3	2	1	0
PP0+PP1	0	0	0	0	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_L
PP2+PP3	0	0	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_L	0	0
PP4+PP5	z_7	z_6	z_5	z_4	z_3	z_2	z_1	z_L	0	0	0	0

Col.	11	10	9	8	7	6	5	4	3	2	1	0
PP0+PP1	z_7	z_6	z_5	z_4	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_L
PP2+PP3	0	0	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_L	0	0
PP4+PP5	0	0	0	0	z_3	z_2	z_1	z_L	0	0	0	0

Mapping Details


0 Z7 0 Z6 Y7 Z5 Y6 Z4 Y5 X7 Y4 X6 Y3 X5 Y2 X4 Y1 X3 YL X2 0 X1 XL
0 0 0 0 Z3 Z2 Z1 ZL
r11 r10 r9 r8 r7 r6 r5 r4 r3 r2 r1 r0



$$
\begin{aligned}
s_1 &= x_4 \oplus y_2 \oplus z_L \\
c_1 &= \text{Majority}(x_4, y_2, z_L) \\
&= (x_4 \cdot y_2) + (x_4 \cdot z_L) + (y_2 \cdot z_L) \\
hs_1 &= x_5 \oplus y_3 \oplus z_1 (\text{auxiliary cell}) \\
hc_1 &= \text{Majority}(x_5, y_3, z_1)(\text{auxiliary cell}) \\
s_2 &= x_6 \oplus y_4 \oplus z_2 \\
c_1 &= \text{Majority}(x_6, y_4, z_2) \\
hs_2 &= x_7 \oplus y_5 \oplus z_3 (\text{auxiliary cell}) \\
hc_2 &= \text{Majority}(x_7, y_5, z_3)(\text{auxiliary cell})
\end{aligned}
$$

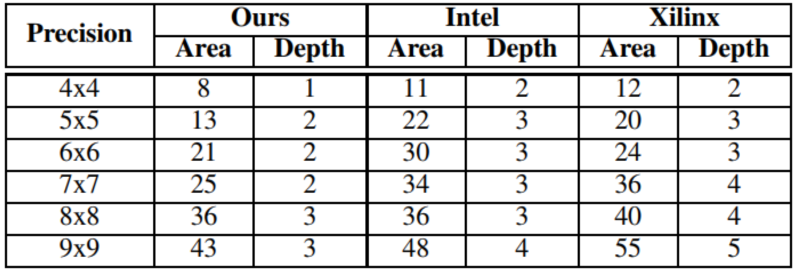
$$s_3 = z_4 \oplus y_6 \qquad c_3 = z_4 \cdot y_6$$

$$s_4 = z_5 \oplus y_7 \qquad c_4 = z_5 \cdot y_7$$

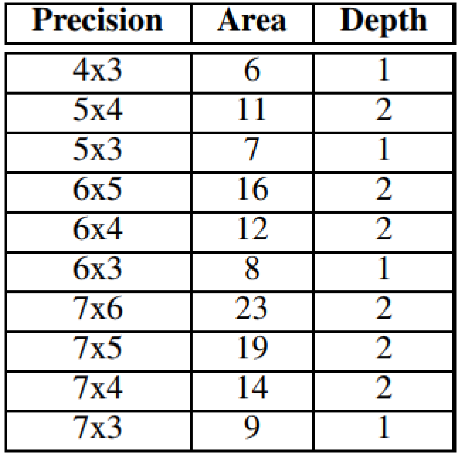
Col.	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Line0	z_7	z_6	s_4	s_3	hs_2	s_2	hs_1	s_1	x_3	x_2	x_1	x_L
Line1	0	c_4	c_3	hc_2	c_2	hc_1	c_1	0	y_1	y_L	0	0

RESULTS AND CONCLUSIONS

Results



Precision	Ours		Intel		Xilinx	
	Area	Depth	Area	Depth	Area	Depth
4x4	8	1	11	2	12	2
5x5	13	2	22	3	20	3
6x6	21	2	30	3	24	3
7x7	25	2	34	3	36	4
8x8	36	3	36	3	40	4
9x9	43	3	48	4	55	5



Precision	Area	Depth
4x3	6	1
5x4	11	2
5x3	7	1
6x5	16	2
6x4	12	2
6x3	8	1
7x6	23	2
7x5	19	2
7x4	14	2
7x3	9	1

Summary

Introduced out-of-band (Auxiliary) functions

Maximized use of independent routing

- Overall routing use low stress
  - Likely to support high system density

Multiple optimizations can be used together

Smallest – and lowest latency – low precision multiplier results known intel®