LUT-Oriented Boolean Decomposition for Efficient Arithmetic on FPGAs and ASICs

Danila Gorodetsky and Leonel Sousa

INESC-ID, Instituto Superior Tecnico, Universidade de Lisboa, Portugal

high-level representation

logic design

“well-suited” algebraic formula

Boolean representations

splitting input and output operands into sub-words of smaller bit-width

representation of the formula as a superposition of “simple” arithmetic functions

properties of Boolean functions

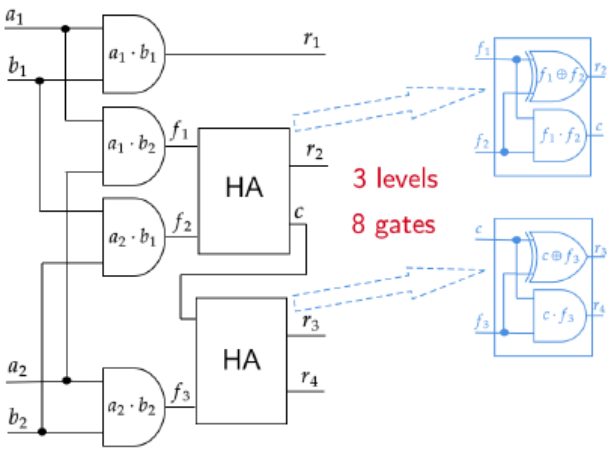
Boolean representations

Boolean minimizations

Efficient representation of arithmetic operation

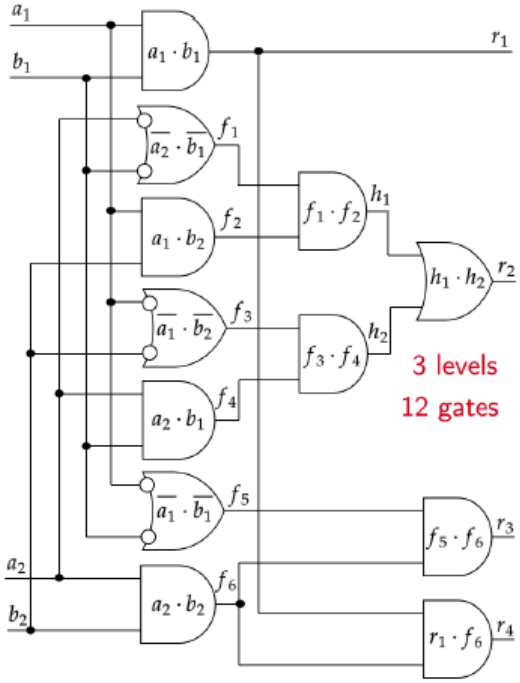
$$A \cdot B = R$$

$$A = (a_2, a_1), B = (b_2, b_1), R = (r_4, r_3, r_2, r_1)$$



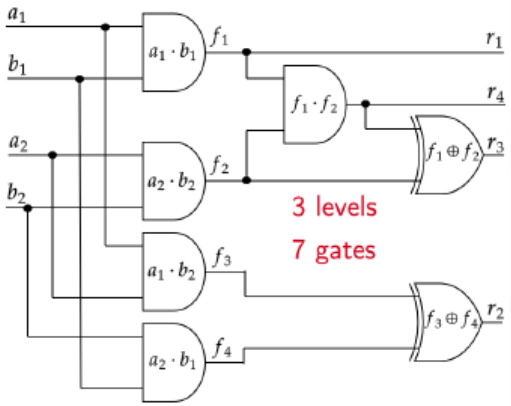
a) School-book multiplication

$r_1 = a_1 \cdot b_1$
$r_2 = f_1 \oplus f_2$
$r_3 = c \oplus f_3$
$r_4 = c \cdot f_3$
$c = f_1 \cdot f_2$
$f_1 = a_1 \cdot b_2$
$f_2 = a_2 \cdot b_1$
$f_3 = a_2 \cdot b_2$



b) DNF minimization multiplication

$r_1 = a_1 \cdot b_1$
$r_2 = h_1 \vee h_2$
$r_3 = f_5 \cdot f_6$
$r_4 = r_1 \cdot f_6$
$h_1 = f_1 \cdot f_2,$
$h_2 = f_3 \cdot f_4$
$f_1 = \bar{a}_2 \vee \bar{b}_1$
$f_2 = a_1 \cdot b_2$
$f_3 = \bar{a}_1 \vee \bar{b}_2$
$f_4 = a_2 \cdot b_1$
$f_5 = \bar{a}_1 \vee \bar{b}_1$
$f_6 = a_2 \cdot b_2$



c) Reed-Muller multiplication

$r_1 = f_1$
$r_2 = f_3 \oplus f_4$
$r_3 = r_4 \oplus f_2$
$r_4 = f_1 \cdot f_2$
$f_1 = a_1 \cdot b_1$
$f_2 = a_2 \cdot b_2$
$f_3 = a_1 \cdot b_2$
$f_4 = a_2 \cdot b_1$

6 – stage scalable methodology*

split operation into partial products of smaller sub-words

Arithmetic Decomposition

$$F = A_1 \cdot B_1 \cdot C_1 + A_2 \cdot B_2 \cdot C_2 + \ldots + A_w \cdot B_w \cdot C_w$$

represent each partial product as a Boolean function $g_i(X)$

Boolean Mapping

$$A_1 \cdot B_1 \cdot C_1 = g_1(X_1),\ A_2 \cdot B_2 \cdot C_2 = g_2(X_2),\ \ldots,\ A_w \cdot B_w \cdot C_w = g_w(X_w)$$

Boolean Decompos./Minimiz.

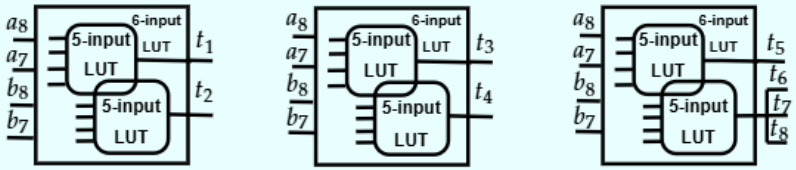
$$F = g_1(h_1(X_1), \ldots, h_v(X_v)) \circ g_2(h_1(X_1), \ldots, h_v(X_v)) \circ \ldots \circ g_w(h_1(X_1), \ldots, h_v(X_v))$$

$$\text{"}\circ\text{"} \in \{+, -, \cdot\}$$

minimize Boolean functions using Reed-Muller, Karnaugh, or technology-dependent methods

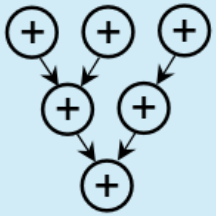
map minimized functions to FPGA LUTs or ASIC standard cells

LUT/Library Mapping



Adders Tree

ballanced parallel additions



combine partial results using balanced parallel addition

Result Integration

concatenation

$$R - P \quad if\ R \geq P$$

modular correction

when required

concatenate sub-results and apply modular correction if R ≥ P

*D. Gorodecky, L. Sousa, "Efficient Arithmetic on FPGA, DATE, 2026, Verona, Italy

Arithmetic Operations

- Modular operations (for an arbitrary P)
  - constant multiplication $((A \cdot const)(\mathrm{mod}\ P) = R)$*
  - modular multiplication $((A \cdot B)(\mathrm{mod}\ P) = R)$*,**,******
  - modular addition $((A + B)(\mathrm{mod}\ P) = S)$
  - modular reduction $(A(\mathrm{mod}\ P) = R)$*
  - Modular division $((A \cdot x)(\mathrm{mod}\ P) \equiv 1)$
  - Residue Number System (RNS)
- Standard operations
  - constant multiplication $(A \cdot const = R)$****
  - multiplication $(A \cdot B = R)$*****
  - addition $(A + B = S)$
  - division by constant $(A = constant \cdot Q + residue)$**
  - Miscellaneous operations ******: $\mathrm{A} \cdot B + C$, $\mathrm{A} \cdot B + C \cdot D$, etc.
- Floating-point computations
  - multiplication $(A \cdot B = R)$
  - addition $(A + B = S)$
  - Normalization
  - Multi-precision computations

*D. Gorodecky, L. Sousa, “Modular Arithmetic Based on Boolean Functions: A Divide and Conquer Approach” // IEEE Access, Vol. 13, Oct., 2025, pp. 186383-186396.
**D. Gorodecky, L. Sousa, "Scalable architecture of constant division on FPGA" // Proceedings of the 30th IEEE Symposium on Computer Arithmetic, Sep. 4-6, 2023, Portland, Oregon, USA.
***D. Gorodecky, L.Sousa, “Two-Operand Modular Multiplication to Small Bit Ranges” // Advanced Boolean Techniques, Springer Nature Switzerland, 2023, p. 111-122.
****D. Gorodecky, P. Bibilo, “Constant Multiplication Based on Boolean Minimization” // Proceedings of the 14th International Workshop on Boolean Problems, Bremen, Germany, Sept. 24-25, 2020.
*****D. Gorodecky “Design of Multipliers Using Fourier Transformations” // Further Improvements in the Boolean Domain, Cambridge Scholars Publishing, UK, 2018, Section 3.4, pp. 240-252.
******D. Gorodecky, "Two approaches of arithmetic units design" // *it - Information Technology* (De Gruyter), Vol. 66, Issue 4-5, 2025, pp-114-123.

Multiplication by power 2

Multiplication by $X \cdot 2^n$ is concatenation of n zeros, $X = (x_\delta x_{\delta-1} \dots x_1)$:

$$X \cdot 2^n = (x_\delta x_{\delta-1} \dots x_1 \underbrace{0 \dots 0}_{n})$$

Ex. $5 \cdot 2^3 = 101_2 \cdot 1000_2 = 101000_2 = 40$

Ex.

$$\begin{array}{r} 4\,0\,3\,0\,6\,0\,9\,0\,2\,0\,5 \\ \times \qquad\qquad\qquad\qquad\qquad 7 \\ \hline 2\,8\,2\,1\,4\,2\,6\,3\,1\,4\,3\,5 \end{array}$$

Ex.

$$\begin{array}{r} 4\,2\,3\,4\,6\,8\,9\,5\,2\,3\,5 \\ \times \qquad\qquad\qquad\qquad\qquad 7 \\ \hline 2\,8\,2\,1\,4\,2\,6\,3\,1\,4\,3\,5 \\ + \quad 1\,4\,2\,8\,5\,6\,3\,5\,2\,1 \\ \hline 2\,9\,6\,4\,2\,8\,2\,6\,6\,6\,4\,5 \end{array}$$

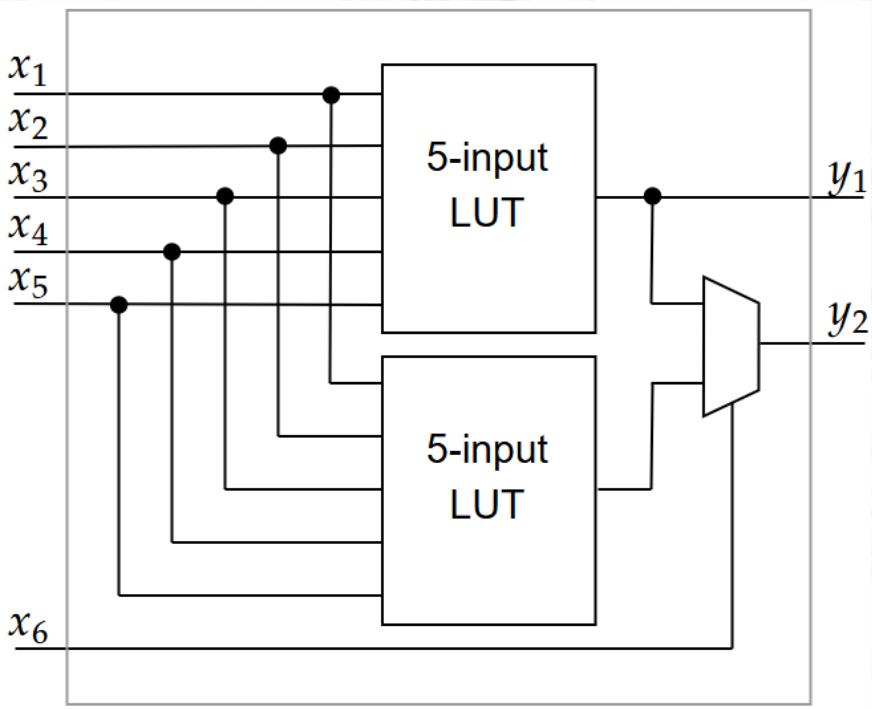
Ex.

$$\begin{array}{r} 10010101000101001010 \\ \times \quad 11010101001001010011 \end{array}$$

1) 10010101000101001010
   11010101001001010011
2) 10010101000101001010
   11010101001001010011
3) 10010101000101001010
   11010101001001010011
   etc.

```
module mult_ANY_NUMBERx2_01_exact (
   x0, x1, x2, x3, x4, x5, x6, x7,…
   z00, z01, z02, z03, z04, z05, z06, z07, z08, z09, z10, z11, …… );
 input  x0, x1, x2, x3, x4, x5, x6, x7, …….;
 output z00, z01, z02, z03, z04, z05, z06, z07, z08, z09, z10, z11…….;

 assign z00 = x0 & x1 & x2 & x3;
 assign z01 = x0 & x2 & (~x3 | (~x1 & x3));
 assign z02 = (x0 & x3 & (~x2 | (~x1 & x2))) | (x1 & x2 & (~x0 | ~x3));
 assign z03 = x1 & x3;

 assign z04 = x5 & x0 & x1 & x4;
 assign z05 = x0 & x4 & (~x5 | (~x1 & x5));
 assign z06 = (x0 & x5 & (~x4 | (~x1 & x4))) | (x1 & x4 & (~x0 | ~x5));
 assign z07 = x1 & x5;

 assign z08 = x7 & x0 & x1 & x6;
 assign z09 = x0 & x6 & (~x7 | (~x1 & x7));
 assign z10 = (x0 & x7 & (~x6 | (~x1 & x6))) | (x1 & x6 & (~x0 | ~x7));
 assign z11 = x1 & x7;
 etc.

endmodule
```

Boolean decomposition and LUT-Aware Mapping



- Decomposition operands to produce small products, where each product represented as Boolean function system (Espresso, ABC)
- ≤6 variables per function: direct mapping to single 6-input LUT
- Functions with >6 variables are decomposed into subfunctions of 5 or 6 variables each
- Architecture-aware exploitation of the Xilinx FPGA LUT structure: 1 LUT = one 6-variables function OR two 5-variables functions

$y_1(x_1, x_2, \dots, x_5)$

$y_2(x_1, x_2, \dots, x_5)$

vs. $y_1(z_1, z_2, \dots, z_6)$

$$A = (a_{12}a_{11} \dots a_1) \qquad B = (b_{12}b_{11} \dots b_1)$$

$$A_1 = (a_4a_3a_2a_1) \qquad B_1 = (b_4b_3b_2b_1)$$
$$A_2 = (a_8a_7a_6a_5) \qquad B_2 = (b_8b_7b_6b_5)$$
$$A_3 = (a_{12}a_{11}a_{10}a_9) \qquad B_3 = (b_{12}b_{11}b_{10}b_9)$$

$A \cdot B = R(\mathrm{mod}\ 3329) =$

$$A_1 \cdot B_1 + A_1 \cdot B_2 \cdot 2^4 + A_1 \cdot B_3 \cdot 2^8 +$$
$$A_2 \cdot B_1 \cdot 2^4 + A_2 \cdot B_2 \cdot 2^8 + A_2 \cdot B_2 \cdot 2^{12} +$$
$$A_3 \cdot B_1 \cdot 2^8 + A_3 \cdot B_2 \cdot 2^{12} + A_3 \cdot B_3 \cdot 2^{16} =$$

High – Level Representation

$$A_1 \cdot B_1 + A_1 \cdot B_2 \cdot 2^4 + A_1 \cdot B_3 \cdot 2^8 +$$
$$A_2 \cdot B_1 \cdot 2^4 + A_2 \cdot B_2 \cdot 2^8 + A_2 \cdot B_2 \cdot 767 +$$
$$A_3 \cdot B_1 \cdot 2^8 + A_3 \cdot B_2 \cdot 767 + A_3 \cdot B_3 \cdot 2285$$

Building blocks:
8 input Boolean functions

$A_1 \cdot B_1$ $\qquad$ $A_1 \cdot B_2 \cdot 2^4$

$A_1 \cdot B_3 \cdot 2^8$ $\qquad$ $A_2 \cdot B_2 \cdot 767$

$A_3 \cdot B_3 \cdot 2285$

$A = (a_8 a_7 \dots a_1) \quad B = (b_8 b_7 \dots b_1)$

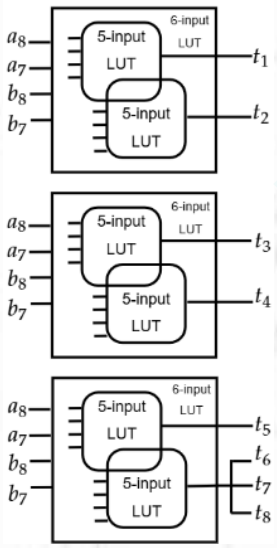
$$(A \cdot B)(\mathrm{mod}\ 241) =$$

$$((a_3a_2a_1) \cdot (b_3b_2b_1) + (a_3a_2a_1) \cdot (b_6b_5b_4) \cdot 2^3 + (a_3a_2a_1) \cdot (b_8b_7) \cdot 2^6 +$$

$$(a_6a_5a_4) \cdot (b_3b_2b_1) \cdot 2^3 + (a_6a_5a_4) \cdot (b_6b_5b_4) \cdot 2^6 + (a_6a_5a_4) \cdot (b_8b_7) \cdot 30 +$$

$$(a_8a_7) \cdot (b_3b_2b_1) \cdot 2^6 + (a_8a_7) \cdot (b_6b_5b_4) \cdot 30 + (a_8a_7) \cdot (b_8b_7) \cdot 240) \ (\mathrm{mod}\ 241) =$$

$$R = (r_8\, r_7 \dots r_1)$$



$$t_1 = a_8 \cdot \left(b_8 \cdot \overline{b_7} \vee \overline{a_7} \cdot b_7\right) \vee\ a_7 \cdot b_8 \cdot \overline{b_7} =$$
$$\overline{b_7} \oplus \overline{b_8} \cdot \overline{b_7} \oplus \overline{a_7} \oplus \overline{a_7} \cdot \overline{b_7} \oplus$$
$$\overline{a_8} \cdot \overline{a_7} \oplus \overline{a_8} \cdot \overline{a_7} \cdot \overline{b_8} \cdot \overline{b_7},$$

$$t_2 = a_8 \cdot b_7 \cdot \left(\overline{a_7} \vee \overline{b_8}\right) \vee a_7 \cdot b_8 \cdot \left(\overline{a_8} \vee \overline{b_7}\right) =$$
$$a_7 \cdot b_8 \oplus a_8 \cdot b_7,$$

$$t_3 = a_8 \cdot \left(\overline{b_8} \cdot\ b_7 \vee \overline{a_7} \cdot b_8 \cdot \overline{b_7}\right) \vee \overline{a_8} \cdot a_7 \cdot b_8 =$$
$$a_7 \cdot b_8 \oplus a_8 \cdot b_7 \oplus a_8 \cdot b_8 \oplus a_8 \cdot b_8 \cdot b_8 \cdot b_8,$$

$$t_4 = a_8 \cdot (b_8 \vee b_7) \vee a_7 \cdot b_8 =$$
$$a_7 \cdot b_8 \oplus a_8 \cdot b_7 \oplus a_8 \cdot b_8 \oplus a_8 \cdot b_8 \cdot b_7 \oplus a_8 \cdot a_7 \cdot b_8,$$

$$t_5 = \overline{a_8} \cdot a_7 \cdot \overline{b_8} \cdot b_7 =$$
$$a_7 \cdot b_7 \oplus a_7 \cdot b_8 \cdot b_7 \oplus a_8 \cdot a_7 \cdot b_7 \oplus a_8 \cdot a_7 \cdot b_8 \cdot b_7,$$

$$t_6 = t_7 = t_8 = (a_8 \vee a_7) \cdot (b_8 \vee b_7) =$$
$$1 \oplus \overline{b_8} \cdot \overline{b_7} \oplus \overline{a_8} \cdot \overline{a_7} \oplus \overline{a_8} \cdot \overline{a_7} \cdot \overline{b_8} \cdot \overline{b_7}$$

$$((a_8a_7) \cdot (b_8b_7) \cdot 240)(\mathrm{mod}\ 241) = T(t_8t_7 \dots t_1)$$

decimal $(a_8a_7) \cdot (b_8b_7) \cdot 240$	a_8	a_7	b_8	b_7	t_8	t_7	t_6	t_5	t_4	t_3	t_2	t_1	decimal T
$0 \cdot 0 \cdot 240$	0	0	0	0	0	0	0	0	0	0	0	0	0
$0 \cdot 1 \cdot 240$	0	0	0	1	0	0	0	0	0	0	0	0	0
$0 \cdot 2 \cdot 240$	0	0	1	0	0	0	0	0	0	0	0	0	0
$0 \cdot 3 \cdot 240$	0	0	1	1	0	0	0	0	0	0	0	0	0
$1 \cdot 0 \cdot 240$	0	1	0	0	0	0	0	0	0	0	0	0	0
$1 \cdot 1 \cdot 240$	0	1	0	1	1	1	1	1	0	0	0	0	240
$1 \cdot 2 \cdot 240$	0	1	1	0	1	1	1	0	1	1	1	1	239
$1 \cdot 3 \cdot 240$	0	1	1	1	1	1	1	0	1	1	1	0	238
$2 \cdot 0 \cdot 240$	1	0	0	0	0	0	0	0	0	0	0	0	0
$2 \cdot 1 \cdot 240$	1	0	0	1	1	1	1	0	1	1	1	1	239
$2 \cdot 2 \cdot 240$	1	0	1	0	1	1	1	0	1	1	0	1	237
$2 \cdot 3 \cdot 240$	1	0	1	1	1	1	1	0	1	0	1	1	235
$3 \cdot 0 \cdot 240$	1	1	0	0	0	0	0	0	0	0	0	0	0
$3 \cdot 1 \cdot 240$	1	1	0	1	1	1	1	0	1	1	1	0	238
$3 \cdot 2 \cdot 240$	1	1	1	0	1	1	1	0	1	0	1	1	235
$3 \cdot 3 \cdot 240$	1	1	1	1	1	1	1	0	1	0	0	0	232

*D. Gorodecky, L. Sousa, “Modular Arithmetic Based on Boolean Functions: A Divide and Conquer Approach” // IEEE Access, Vol. 13, Oct., 2025, pp. 186383-186396.

Division: high-level representation*

$$\frac{X}{d} = \{Quotient, Residue\} \quad \delta = \lceil log_2 d \rceil$$

1. $$X = (X_3 X_2 X_1) = (X_3 \underbrace{00\ldots0}_{2\cdot\delta}) + \left(X_2 \underbrace{00\ldots0}_{\delta}\right) + (X_1)$$

Ex. d=7, X=489=400+80+9

2. $$\frac{(X_3 \underbrace{00\ldots0}_{2\cdot\delta})}{d} = \{Q_3, R_3\}, \quad \frac{(X_2 \underbrace{00\ldots0}_{\delta})}{d} = \{Q_2, R_2\}, \quad \frac{X_1}{d} = \{Q_1, R_1\}$$

Ex. $\frac{400}{7} = \{57,1\}, \frac{80}{7} = \{11,3\}, \frac{9}{7} = \{1,2\}$

3. $$\frac{R_3+R_2+R_1}{d} = \{Q_t, R\}, \quad Q_3 + Q_2 + Q_1 + Q_t = Q$$

Ex. $\frac{1+3+2}{7} = \{0,6\}$, 57+11+1+0=69=*Q*, *R=6*

* D. Gorodecky, L. Sousa, " Scalable architecture of constant division on FPGA" // Proceedings of the 30th IEEE Symposium on Computer Arithmetic, Sep. 4-6, 2023, Portland, Oregon, USA.

Xilinx Vivado 2022.2

{xc7k70tfbg484-3 "Kintex-7"}
{xc7a100tcsg324-1 "Artix-7"}
{xcku5p-ffvb676-2-e "KintexUS+"}

	Synthesis	Implementation
{default_no_clk 0 "Default_NoClock"	**"Vivado Synthesis Defaults"**	**"Vivado Implementation Defaults"}**
{default_10ns 10 "Default_100MHz"	"Vivado Synthesis Defaults"	"Vivado Implementation Defaults"}
{default_1ns 1 "Default_1GHz"	"Vivado Synthesis Defaults"	"Vivado Implementation Defaults"}
{area_no_clk 0 "Area_NoClock"	"Flow_AreaOptimized_high"	"Vivado Implementation Defaults"}
{area_10ns 10 "Area_100MHz"	"Flow_AreaOptimized_high"	"Vivado Implementation Defaults"}
{area_1ns 1 "Area_1GHz"	"Flow_AreaOptimized_high"	"Vivado Implementation Defaults"}
{speed_no_clk 0 "Speed_NoClock"	"Flow_PerfOptimized_high"	"Performance_Explore"}
{speed_10ns 10 "Speed_100MHz"	"Flow_PerfOptimized_high"	"Performance_Explore"}
{speed_1ns 1 "Speed_1GHz"	"Flow_PerfOptimized_high"	"Performance_Explore"}
{area_max 0 "Area_Max_Optimization"	"Flow_AreaOptimized_high"	"Area_Explore"}
{area_seq 0 "Area_Sequential_Opt"	"Flow_AreaOptimized_high"	"Area_ExploreSequential"}
{speed_5_def 5 "Speed_200MHz_Default"	"Vivado Synthesis Defaults"	"Vivado Implementation Defaults"}
{speed_5_perf 5 "Speed_200MHz_Performance"	"Flow_PerfOptimized_high"	"Vivado Implementation Defaults"}
{speed_3_def 3 "Speed_333MHz_Default"	"Vivado Synthesis Defaults"	"Vivado Implementation Defaults"}
{speed_3_perf 3 "Speed_333MHz_Performance"	"Flow_PerfOptimized_high"	"Performance_Explore"}
{speed_1_def 1 "Speed_1GHz_Default"	"Vivado Synthesis Defaults"	"Vivado Implementation Defaults"}
{speed_1_perf 1 "Speed_1GHz_Performance"	"Flow_PerfOptimized_high"	"Performance_Explore"}

Synopsys 2022, Cadence 2022

2 libraries: **scl45nm (45nm)**, tsmc (28nm)

Time constraints synthesis:
- none (area optimization)
- 1ns (critical path optimization)
- 10ns (balanced optimization)

FloPoCo (state-of-the-art)

flopoco target=kintex7 IntConstDiv wIn=*bit_range* d=*divider* arch=*{0,3}* alpha={*5,6}*
flopoco target=kintex7 IntConstDiv computeQuotient=false wIn=*input_bit_range* d=*modulo* arch=3 alpha={5,6}
flopoco target=kintex7 IntConstMult wIn=*bit_range_input* c=*constant*

Results: Multiplication by a Constant

$constant \cdot A$

- 7-bit variable A and 29-bit prime $C = 536870909 = 2^{29} - 3$, as it titled 29×7;
- 8-bit variable A and 46-bit prime $C = 70368744177629$, as it titled 46×8;
- 9-bit variable A and 101-bit prime $C = 2535301200456458802993406409959$, as it titled 101×9;
- 9-bit variable A and 157-bit $C = 2^{157} - 7$, as it titled 157×9;
- 10-bit variable A and 183-bit $C = 2^{183} - 1$, as it titled 183×10

Vivado

Multiplication bits × bits	LUTs				critical path (ns)			
	Vivado	*lut_5*	*lut_6*	*fpc*	*Vivado*	*lut_5*	*lut_6*	*fpc*
7× 29	24	**17**	**17**	24	8.0	7.4	**7.2**	7.9
8× 46	71	**46**	49	78	9	**8.1**	8.3	9.2
9 × 101	126	**79**	85	134	13.8	**12.2**	**12.2**	13.1
9 × 157	179	**44**	49	244	**14.7**	16.3	15.8	16.1
10 × 183	204	**18**	**18**	257	15.9	16.3	17	**15.5**

Cadence

Multiplication bits × bits	cells				critical path (ns)			
	Cadence	*lut_5*	*lut_6*	*fpc*	*Cadence*	*lut_5*	*lut_6*	*fpc*
7× 29	**147**	228	161	161	0.8	**0.3**	0.4	0.6
8× 46	**306**	808	792	713	0.9	**0.7**	**0.7**	1.2
9 × 101	**387**	1063	1060	1660	**0.9**	**0.9**	**0.9**	1.7
9 × 157	**208**	483	464	964	0.8	0.6	**0.5**	1.4
10 × 183	138	**133**	**133**	795	**0.4**	0.5	0.5	1.2

Synopsys

Multiplication bits × bits	cells				critical path (ns)			
	Synopsys	*lut_5*	*lut_6*	*fpc*	*Synopsys*	*lut_5*	*lut_6*	*fpc*
7× 29	2741	**298**	312	361	1.0	**0.4**	**0.4**	0.9
8× 46	4753	1074	**1051**	2502	1.2	0.9	**0.7**	1.1
9 × 101	10804	1620	**1601**	5798	1.3	**0.9**	1.0	1.5
9 × 157	17816	617	**599**	8184	1.4	0.6	**0.5**	1.3
10 × 183	23348	246	**238**	8576	1.4	**0.4**	0.5	1.4

Results: Modular Multiplication

$$(A \cdot B) (\mathrm{mod}\ P)$$

Vivado

P	LUTs			critical path (ns)		
	Vivado	*lut_5*	*lut_6*	*Vivado*	*lut_5*	*lut_6*
241	145	128	**127**	13.6	10.9	**10.7**
491	162	**125**	129	13.9	**10.5**	10.9
997	**232**	244	245	14.8	**13.8**	13.9
2011	**219**	282	277	16.2	14.0	**14.3**
4051	298	**295**	310	17.1	**14.4**	14.9

Cadence

P	cells			critical path (ns)		
	Cadence	*lut_5*	*lut_6*	*Cadence*	*lut_5*	*lut_6*
241	**2083**	2279	2199	2.7	**2.0**	2.1
491	**2335**	2852	2891	3.1	**2.2**	**2.2**
997	**3288**	3624	3512	4.0	**2.6**	**2.6**
2011	**3613**	4225	4094	4.6	2.6	**2.3**
4051	**4518**	5322	5355	5.2	**2.8**	3.0

Synopsys

P	cells			critical path (ns)		
	Synopsys	*lut_5*	*lut_6*	*Synopsys*	*lut_5*	*lut_6*
241	**2653**	4409	4276	3.0	**1.5**	**1.5**
491	**3302**	5395	5103	3.5	**1.5**	**1.5**
997	**3912**	7739	7156	3.8	**1.7**	1.8
2011	**4705**	8603	8131	4.4	**1.7**	**1.7**
4051	**5748**	10447	9875	4.7	**1.6**	**1.6**

$A \pmod P$

Vivado

bit range of A	P	LUTs				critical path (ns)			
		Vivado	*lut_6*	*fpc_0*	*fpc_3*	*Vivado*	*lut_6*	*fpc_0*	*fpc_3*
168	241	4419	400	7979	**380**	25.7	17.2	238.4	**16.1**
	491	4330	**467**	12333	508	23.6	18.4	281.4	**17.9**
	997	4706	599	13198	**558**	23.6	17.0	298.2	**16.6**
	2011	4290	**534**	14847	610	25.9	17.7	308.8	**16.5**
	4051	4225	**484**	4691	529	25.3	17.5	263.4	**17.3**
270	241	10024	**509**	13847	604	33.6	20.5	382.6	**17.2**
	491	11088	**701**	17185	826	33.7	23.1	445.7	**17.3**
	997	11310	**760**	15061	905	30.7	22.4	444.9	**17.2**
	2011	11670	**834**	10454	1005	34.2	21.9	415.0	**18.0**
	4051	9926	**744**	7751	861	33.4	21.5	426.8	**18.6**

Cadence

bit range of A	P	cells				critical path (ns)			
		Cadence	*lut_6*	*fpc_0*	*fpc_3*	*Cadence*	*lut_6*	*fpc_0*	*fpc_3*
168	241	19474	**5274**	9447	8417	40.2	**2.3**	2.7	2.7
	491	21041	**13299**	14722	16519	49.1	**2.9**	3.2	3.1
	997	31824	**14280**	14868	16519	58.7	**2.8**	2.9	3.1
	2011	33253	**16388**	17588	28828	60.7	**2.5**	3.4	3.4
	4051	27766	**10840**	13782	14095	51.3	**2.7**	3.3	3.0
270	241	164957	**8178**	14489	14134	210.0	**2.6**	3.0	2.8
	491	225954	**21381**	24630	27730	280.0	**3.3**	3.9	3.7
	997	213834	**21801**	26337	29756	268.0	**3.2**	3.7	3.9
	2011	173843	**25866**	29421	34086	198.0	**3.3**	4.0	4.1
	4051	197444	**16374**	22502	24480	248.3	**2.9**	4.0	4.3

Synopsys

bit range of A	P	cells				critical path (ns)			
		Synopsys	*lut_6*	*fpc_0*	*fpc_3*	*Synopsys*	*lut_6*	*fpc_0*	*fpc_3*
168	241	23155	**10486**	19488	27827	33.8	2.13	**2.0**	**2.0**
	491	26315	**19275**	47380	69012	34.3	2.3	**2.1**	2.2
	997	28844	**18799**	49922	68247	35.1	**2.1**	2.2	2.3
	2011	32367	**22717**	44571	70130	37.3	**2.1**	2.3	2.3
	4051	35757	**16266**	30253	40081	38.4	**2.0**	2.2	2.2
270	241	36428	**15758**	32067	42038	56.4	**2.3**	**2.3**	**2.3**
	491	42121	**28570**	89559	150875	57.6	2.6	**2.5**	**2.5**
	997	44700	**29113**	89706	143962	60.1	2.5	**2.4**	2.5
	2011	49225	**33825**	88314	139098	64.8	**2.3**	2.5	2.6
	4051	52826	**24686**	51673	65493	68.2	**2.2**	2.5	2.5

Results: Division by a Constant

$$\frac{A}{d} = \{Quotient, Residue\}$$

Vivado

bit range of A	d	LUTs					critical path (ns)				
		Vivado	*[20]*	*lut_6*	*fpc_0*	*fpc_3*	*Vivado*	*[20]*	*lut_6*	*fpc_0*	*fpc_3*
16	5	**30**	51	64	71	59	8.0	8.7	**7.7**	8.4	9.0
	11	**37**	52	57	154	59	10.6	**8.5**	8.6	9.7	9.6
	13	**37**	70	66	184	63	10.3	8.7	**8.1**	9.2	9.0
32	5	718	125	**123**	158	155	15.6	15.8	**9.5**	12.9	10.7
	11	563	**158**	165	370	182	15.8	16.4	**10.1**	14.1	11.5
	23	445	**186**	187	979	197	14.7	15.3	**11.1**	18.7	11.7
48	47	1144	**394**	423	2849	410	18.4	22.8	**13.5**	29.5	13.7
	113	1382	501	523	4672	**408**	17.6	19.7	**13.4**	38.1	14.3
	241	1596	750	575	4713	**386**	18.4	17.2	**13.4**	35.5	14.3
64	5	2302	385	460	**341**	523	21.1	30.4	**12.8**	22.0	14.7
	11	2039	**435**	467	955	589	22.1	32.3	**12.5**	27.0	16.0
	23	1532	**493**	514	2454	542	19.2	33.0	15.5	35.7	**15.3**

Cadence

bit range of A	d	cells					critical path (ns)				
		Cadence	*[20]*	*lut_6*	*fpc_0*	*fpc_3*	*Cadence*	*[20]*	*lut_6*	*fpc_0*	*fpc_3*
16	5	**762**	1281	1131	1110	963	**1.1**	1.3	1.3	1.3	1.4
	11	**1335**	1413	1344	3225	1346	1.9	**1.4**	**1.4**	2.2	1.7
	13	1403	1097	**844**	4176	1409	1.8	**1.2**	**1.2**	3.2	1.4
32	5	**2197**	3371	2564	2347	2741	**1.3**	2.1	1.4	2.9	2.0
	11	**3108**	4132	3677	6774	3847	4.4	2.1	**1.7**	5.4	2.3
	23	3999	3840	**3520**	12277	4053	5.0	2.0	**1.4**	7.0	2.2
48	47	**7284**	9806	8336	29449	9781	9.1	2.9	**2.1**	21.1	3.0
	113	8134	8616	8288	47828	**8106**	9.6	3.0	**2.3**	31.2	2.8
	241	9500	7401	7393	36713	**6345**	9.7	2.5	**2.2**	20.7	2.8
64	5	5485	10193	7662	**4825**	9768	2.7	4.2	**1.9**	6.2	3.2
	11	**6033**	11311	9361	15324	12553	9.8	3.9	**2.2**	14.6	3.5
	23	**7903**	10467	9788	27861	11804	11.2	3.5	**2.4**	16.8	3.3

Synopsys

bit range of A	d	cells					critical path (ns)				
		Synopsys	*[20]*	*lut_6*	*fpc_0*	*fpc_3*	*Synopsys*	*[20]*	*lut_6*	*fpc_0*	*fpc_3*
16	5	**1445**	2110	1589	3199	2509	2.2	1.1	**1.0**	1.1	1.3
	11	1945	2221	**1939**	9980	2701	2.3	**1.1**	**1.1**	**1.1**	1.5
	13	1929	1927	**1250**	10861	3010	2.2	1.1	**1.0**	1.1	1.4
32	5	**3040**	5003	4189	7752	5768	4.2	2.2	**1.2**	1.5	1.5
	11	**4097**	5970	5430	21508	6503	5.0	2.1	**1.1**	2.2	1.6
	23	**4990**	6090	5398	36635	7138	5.2	2.1	**1.1**	2.4	1.7
48	47	**9414**	12709	12620	80152	15172	8.3	3.0	**1.1**	3.8	1.9
	113	**10315**	12212	12038	153050	14748	8.3	2.6	**1.1**	4.5	2.1
	241	11982	**10021**	**10021**	336409	111725	8.5	2.2	**1.2**	5.1	2.0
64	5	**5883**	15615	16068	14035	16030	8.8	4.5	**1.1**	2.9	2.0
	11	**8269**	17185	17121	38097	17102	9.9	4.3	**1.2**	4.4	2.1
	23	**10251**	15780	16090	62043	18741	11.1	4.0	**1.2**	4.8	2.0

- «Universal» methodology for arithmetic unit design via Boolean decomposition
- up to 85× shorter critical path and up to 28× fewer cells" / "up to 2× shorter critical path and up to 20× fewer LUTs
- Cadence gives better results than Synopsys
- Up to 2× critical path and up to 20× LUTs (Xilinx Vivado)
- Runtime: synthesis time is minutes vs. hours for large input bit-widths, compared with the built-in algorithms of Vivado, Cadence, Synopsys, and FloPoCo