

Title: Design and Synthesis of Small and Fast Finite Field Multipliers for FPGAs
Authors: Gregory C Ahlquist, Brent E. Nelson, and Michael D. Rice
Department of Electrical and Computer Engineering
Brigham Young University, Provo, UT 84602

We present a novel method for designing and synthesizing finite field multipliers for FPGAs. Our method leverages characteristics of both FPGA architecture and finite field multiplication to significantly decrease multiplier area while increasing performance. First, our method exploits the ability of K-input Look-Up-Tables to compute any logical combination of inputs in a fixed circuit area. This capability, coupled with the exclusive-or identity $b + b = 0$ (+ here equals the logical XOR operation), makes it possible to significantly reduce the size of both pipelined and combinatorial finite field multiplier circuits while increasing throughput - by as much as 67 percent compared to traditional synthesis methods.

Finite field multipliers are the key circuits enabling certain widely used error correction codes and encryption schemes. These capabilities, in turn, play significant roles in emerging technologies of military interest such as software defined radios, dynamic error correction, and dynamic encryption. The dynamic nature of these technologies demands the use of FPGAs and other programmable devices. Thus, the efficient marriage of finite field multipliers and FPGAs stands to pay huge dividends in these fields.

In recent work, we have extensively compared and contrasted the performance of combinatorial and pipelined versions of standard and state-of-the-art finite field multiplier designs in FPGAs. We have learned that finite field multipliers designed and optimized for VLSI implementation and synthesized with current commercially available synthesis engines produce sub-optimal results in the FPGA environment. As a result, we developed an FPGA-based, finite field multiplier design and synthesis method that significantly reduces the number of FPGA resources needed to implement a multiplier circuit while increasing computational throughput. Depending on the finite field, our method can improve the functional density (product of area and clock cycle time) of a given design by as much as 67 percent over traditionally synthesized multipliers.

In essence, our design approach is a concurrent logic minimization and technology-mapping technique specifically developed for Look-Up Table (LUT)-based FPGAs. The hallmark of our method is a highly counter-intuitive algorithm that **increases** the size and complexity of the governing multiplier equations in order to **reduce** design area and clock cycle time. We find that adding terms to the governing equations leads to efficient technology mappings that we can exploit with FPGA LUTs and certain characteristics of finite field mathematics. Specifically, we use the XOR identity $b \oplus b = 0$, associated with finite fields with characteristic 2 (binary extension fields), to leverage the ability of any LUT to compute any logical combination of its inputs. The result allows us to map the governing finite field multiplier equations to FPGAs in a manner that reduces overall circuit area and decreases clock cycle time.

We have developed a complete suite of software tools that implement our design and synthesis ideas. These tools accept a finite field generator polynomial as input and

produce a VHDL file fully describing our multiplier design as output. We compare our results to combinatorial and pipelined versions of Mastrovito's standard bit-parallel finite field multiplier synthesized with current commercially available tools. We show that, in all cases, our pipeline designs significantly outperform traditionally synthesized circuits and our combinatorial designs are as good and often better than their traditionally synthesized counterparts. A sample of our data for certain select finite fields appears in the tables below. The headings LB, reg, CT, and FD stand for Logic Block, Registers, Clock cycle Time, and Functional Density respectively. A Logic Block is the common pairing of a LUT and a register found in most FPGA architectures. As the data shows, the biggest advantage gained by our design method for pipelined designs is a significant reduction in the number of external registers. For combinatorial designs, our method both reduces the number of required LUTs and decreases the length of the critical path resulting in decreased clock cycle time.

Pipelined Finite Field Multipliers										
Finite Field	Generator Polynomial	Mastrovito/Synplify				Our Method				% Imp.
		LB	reg	CT	FD	LB	reg	CT	FD	
GF(8)	1101	8	3	4.312	47.432	8	0	4.312	34.496	27.3
GF(16)	10011	12	11	4.312	99.176	12	0	4.312	51.744	47.8
GF(32)	100101	21	16	4.746	175.602	18	0	4.422	79.596	54.7
GF(32)	111101	20	30	4.318	215.900	21	3	4.422	106.128	50.8
GF(64)	1000011	28	17	4.795	215.775	27	4	4.312	133.672	38.1
GF(64)	1110011	30	31	4.827	294.447	29	4	4.312	142.296	51.7
GF(128)	10000011	37	22	4.795	282.905	39	3	4.727	198.534	29.8
GF(128)	10111111	50	61	5.129	569.319	42	1	4.627	198.961	65.1
GF(128)	11001011	57	42	5.398	534.402	41	1	4.627	194.334	63.6
GF(256)	100011101	54	34	4.896	430.848	52	1	4.312	228.536	47.0
GF(256)	110001101	71	56	5.656	718.312	54	1	4.312	237.160	67.0
GF(256)	111110101	52	50	4.727	482.154	53	1	4.312	232.848	48.3

Combinatorial Finite Field Multipliers								
Finite Field	Generator Polynomial	Mastrovito/Synplify			Our Method			Percent Improvement
		LB	CT	FD	LB	CT	FD	
GF(8)	1101	8	7.698	61.584	7	7.698	53.886	12.5
GF(16)	10011	12	9.316	111.790	12	6.210	74.528	33.3
GF(32)	100101	20	9.520	190.400	18	6.350	114.240	40.0
GF(32)	111101	20	11.569	231.380	20	8.677	173.535	25.0
GF(64)	1000011	28	9.520	266.560	26	9.520	247.520	7.1
GF(64)	1110011	30	11.569	347.070	28	8.673	242.949	30.0
GF(128)	10000011	37	9.621	355.977	36	9.621	346.356	2.7
GF(128)	10111111	50	13.659	682.950	39	8.195	319.621	53.2
GF(128)	11001011	57	12.181	694.317	39	9.136	356.294	48.7
GF(256)	100011101	54	12.124	654.696	51	9.093	463.743	29.2
GF(256)	110001101	71	13.775	978.025	54	8.265	446.310	54.4
GF(256)	110101001	61	14.204	866.444	50	8.522	426.120	50.8
GF(256)	111110101	52	13.659	710.268	51	8.195	417.965	41.2

To simplify our presentation, we have scoped the results presented here to consider only finite fields of $GF(256)$ and smaller, finite fields using the canonical basis, and FPGA architectures that incorporate 4 input LUTs. We justify these limitations by observing that finite field $GF(256)$ and smaller are widely used in error correction codes and some forms of encryption and that the standard basis for fielded finite field systems is the canonical basis. Also, most FPGA manufactures offer FPGA architectures built around 4-input LUTs. Nevertheless, we have shown in the lab that we can extend our algorithms to larger finite fields, fields using different bases, and to LUTs of varying input size. These are all topics of future research.