A Novel Modular Adder for One Thousand Bits and More Using Fast Carry Chains of Modern FPGAs

Marcin Rogawski, Ekawat Homsirikamol, and Kris Gaj
Volgenau School of Engineering
George Mason University
Fairfax, Virginia 22030
email: {mrogawsk, ehomsiri, kgaj}@gmu.edu

***Abstract*—In this paper a novel, low-latency family of high-radix Parallel Prefix Network adders and modular adders has been proposed. This family efficiently takes advantage of fast carry chains of modern FPGAs. The implementation results reveal that these adders have great potential for efficient implementation of modular addition with the long integers used in various public key cryptography schemes.**

***Keywords*—FPGA, parallel prefix network, Kogge-Stone, Brent-Kung, ripple carry adder**

I. Introduction

Adders are one of the most important kinds of digital circuits. They are used extensively in various branches of science and engineering, such as digital signal processing [1], computer graphics, and cryptography [2], [3]. Addition can also serve as a basic building block of higher level arithmetic operations: modular addition, multiplication, modular reduction, modular multiplication, and Montgomery multiplication [4], [5], [6], [3].

Many hardware architectures for adders have been investigated and optimized for various applications and implementation platforms [4], [5], [7], [8], [9], [10], [11], [12]. Of particular interest to us is the application of adders in public key cryptography, and their implementation using modern families of FPGAs.

Public key cryptography, which emerged in the mid-1970s, covers several families of cryptographic algorithms, in which communicating parties do not need to share a common key before exchanging confidential and authenticated messages with each other. Examples of public key algorithms include RSA, Diffie-Hellman, DSA, Elliptic Curve Cryptosystems [2], and Pairing Based Cryptosystems [13]. These transformations are typically used for digital signatures, key agreement, key exchange, identity-based encryption, and many other applications.

The common feature of these algorithms is that they operate on long operands of between 160 and 15424 bits. For such long operands, even relatively simple operations, such as addition and modular addition become very challenging to implement efficiently, especially in FPGAs. In order to fully optimize these operations, we need to employ all relevant embedded (hardwired) resources of modern FPGAs, as well as to adapt the best hardware architectures of adders, originally developed for ASICs.

In this paper, we describe and analyze several novel hardware architectures for addition and modular addition that are highly applicable for operations on very long operands in the range of one thousand bits and beyond.

II. Previous Work

The state of the art in the design of fast adders targeting standard cell ASICs is described in [4], [5]. These books and the papers they cite extensively cover the design of Parallel Prefix Network (PPN) adders, which are believed to be the fastest and most suitable for pipelining among the traditional fast adder designs. Somewhat surprisingly, none of these resources (or any other publication we are aware of) mentions the possibility of generalizing these adders to high-radix PPN adders, which is a subject of this paper.

Novel FPGA-specific two-operand adder architectures, targeting similar operand sizes and optimization goals as our work, are described in [10], [11] and [12]. Unlike our approach, which generalizes and optimizes Parallel Prefix Network adders, all these papers use as a starting point an alternative design – Carry Select Adder. This adder is then transformed into several related architectures optimized for operation using the fast carry chains of modern FPGAs.

Based on a detailed analysis of the aforementioned papers, we believe that the best design to date has been proposed in the most recent work from FPL 2011 [12]. In this paper, the authors present three related architectures based on Carry Select Adder, which they refer to as the AAM (Add-Add-Multiplex), the CAI (Compare-Add-Increment), and the CCA (Compare-Compare-Add) architectures. In Section VIII, we provide a detailed comparison of our design with the most efficient architecture from [12]. Additionally, we extend the selected design from [12] to a modular adder by applying a standard construction from [6], and demonstrate that this design is inferior to our design of a modular high-radix PPN adder described in Section V.

Alternative architectures of Parallel Prefix Network adders, different than the classical designs of Kogge-Stone and Brent-Kung, have been described in [14] and [7]. In both cases, these architectures targeted CMOS standard-cell technology, and were not optimized with a FPGA implementation in mind. An FPGA-based analysis of different architectures of PPN adders was conducted in [15].

Apart from the resources related to two-operand addition, there exists a very rich literature regarding the generalization

and optimization of multi-operand addition using carry save arithmetic in FPGA technology. Significant papers covering this topic include [16] and [17]. Since these papers address a substantially different problem, a direct comparison with them is outside the scope of this work. Similarly, previous work on pipelined adders, presented in [8], [9], [18], is beyond the scope of this paper.

In Section IX, we propose an extension of modern FPGAs with additional hardwired resources supporting fast addition. This topic has been extensively explored in [19]. Similarly to our proposal, the most efficient embedded structure proposed in [19] is based on a Parallel Prefix Network. However, unlike us, the authors of [19] do not demonstrate the applicability of the proposed extension to fast addition and modular addition of very long operands of the size of one thousands bits and more.

III. Background

In the following discussion, we will assume that the long-operand adders of interest to us do not contain the carry in input. This assumption is justified by the fact that these adders already have long operands, and therefore are not intended to be connected in series.

A. Fast Carry Chains of Modern FPGAs

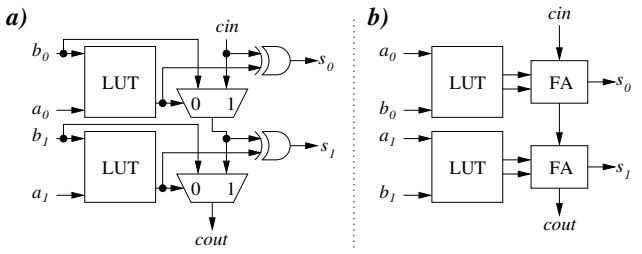

Fig. 1. Fast Carry Chains in a) Xilinx FPGAs, b) Altera FPGAs

Modern FPGAs from all major vendors contain hardwired support for fast addition. These special hardwired components, referred to as fast carry chains, are shown in Fig. 1. In case of Xilinx FPGAs (Fig. 1a), each hardwired stage of an adder includes a 2-to-1 multiplexer (used to calculate a carry out bit) and an XOR gate (used to calculate a sum bit). The full stage, implementing a Full Adder (FA), must also incorporate a corresponding look-up table (LUT), which implements an XOR of two corresponding bits of operands A and B (denoted as a_i, b_i). In case of Altera FPGAs (Fig. 1b), the entire Full Adder (FA) is implemented using hardwired logic. In both cases, neighboring stages can be connected in series, and the hardwired portion of the circuit is optimized for the minimum delay from carry in to carry out.

B. Parallel Prefix Network Adders

Some of the fastest known two-operand adders, designed originally for ASIC technology, are called Parallel Prefix Network (PPN) adders. These adders have low latency and a very regular structure, suitable for pipelining. The general structure of a Parallel Prefix Network adder is shown in Fig. 2a. In the top layer, the generate and propagate signals, g_i and p_i, are calculated in parallel, separately for each bit. In the middle layer, a Parallel Prefix Network is used to calculate the generate-propagate pairs, $g[0,i]$ and $p[0,i]$, for each block of bits starting from position 0 and ending at position i. Since carry in to the entire circuit is equal to zero, the generate signal $g[0,i]$ is equivalent to the projected carry at position $i+1$, denoted as pc_{i+1}. As a result, the generate outputs from a PPN correspond to projected carries at positions from 1 to n, where n is the size of operands in bits. In the bottom layer, these projected carries are XOR-ed with the corresponding propagate bits in order to calculate the final sum bits r_i.

The Parallel Prefix Network is always built of basic blocks, called carry operators, shown in Fig. 2c,d. These networks can have different structures, depending on the exact optimization criteria. The two most commonly used PPNs are shown in Fig. 2c,d for the case of n=8. The Kogge-Stone PPN (Fig. 2c) is optimized for minimum latency [20]. The Brent-Kung PPN (Fig. 2d) is optimized for the minimum product of latency and area [21].

Parallel Prefix Network adders are quite fast in FPGAs, but cannot take advantage of fast carry chains present in modern FPGAs. As a result, these adders must be implemented entirely using look-up tables (LUTs), which negatively affects their latency and resource utilization.

IV. Architecture of a Novel Long-Operand Adder

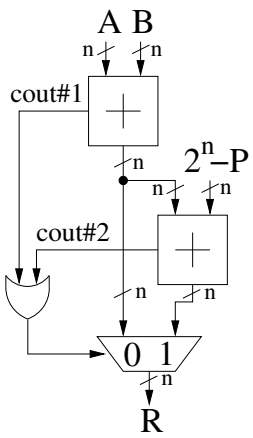

Fig. 4. Modular adder performing operation R=A+B mod P. Notation: n - number of bits of P.

A novel high-radix Parallel Prefix Network adder, taking advantage of fast carry chains of modern FPGAs, is shown in Fig. 2b. This adder is quite similar to the traditional PPN adders with the following exceptions. Arguments A and B are processed in words, instead of bits. Each word has a width of w bits. The GP (generate-propagate) units of a traditional PPN adder are replaced by the GPS (generate-propagate-sum) units. Each of these units takes the corresponding words of operands A and B, denoted as A_i and B_i, and produces three outputs: the generate signal for a block of w bits - g_i, the propagate signal for the same block of w bits - p_i, and the sum word - S_i. The generate and the sum signals can be obtained using a simple w-bit adder. In this adder, the generate signal is equivalent to carry out. The propagate signal for a block of

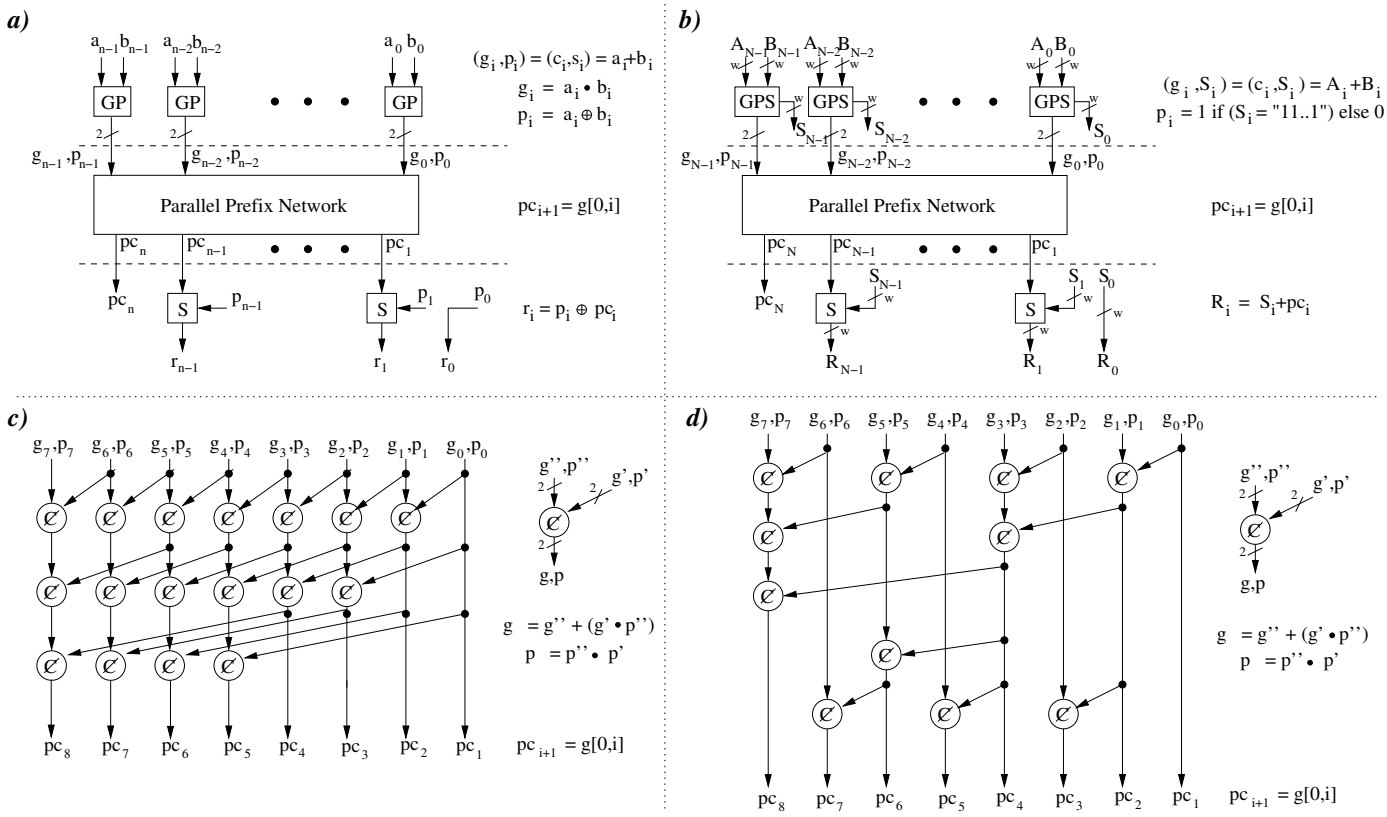

Fig. 2. a) Traditional Parallel Prefix Network adder; + represents an arithmetic addition; b) Proposed novel high-radix Parallel Prefix Network adder; + represents an arithmetic addition; c) Kogge-Stone Parallel Prefix Network for N=8 inputs; + represents logic OR; d) Brent-Kung Parallel Prefix Network for N=8 inputs; + represents logic OR.

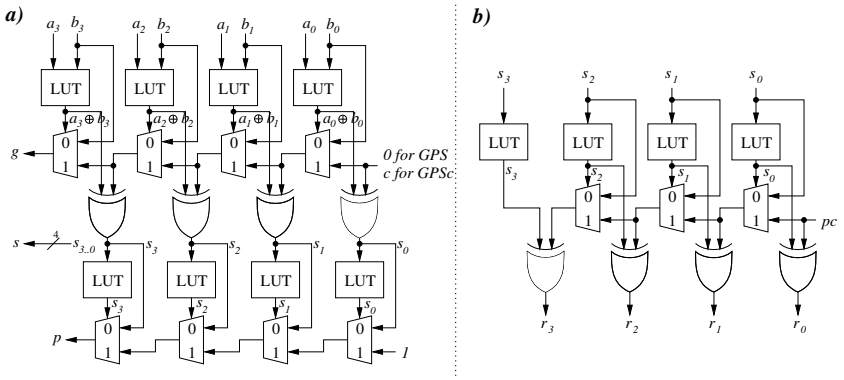

Fig. 3. Components of the proposed adder and modular adder, implemented using fast carry chains of Xilinx FPGAs: a) GPS (generate-propagate-sum) and GPSc (generate-propagate-sum with carry) unit, b) S (sum) unit.

w bits represents the situation when the carry in propagates through the given block. This situation happens only if all bits of the sum are equal to one. Checking this condition is equivalent to checking whether adding one to the sum produces carry out. This calculation can be very efficiently performed using fast carry chains of Xilinx FPGAs, as shown in Fig. 3a.

Similarly, in the third layer of the adder, in the sum units, S, the propagate signals of the traditional adder p_i are replaced by the w-bit sum signals S_i. The carry signals produced by these adders are discarded, as they have been already taken into account.

The Parallel Prefix Network changes its size from n inputs to $N = n/w$ inputs, which substantially reduces the area of the circuit. This PPN may be of the Kogge-Stone [20] type (Fig. 2c) or of the Brent-Kung type [21] (Fig. 2d), depending on our optimization target (latency vs. latency · area). In the following discussion, we will denote our high-radix PPN adder based on the Kogge-Stone PPN by HR-KS, and our adder based on the Brent-Kung PPN by HR-BK.

The critical path of our adder includes

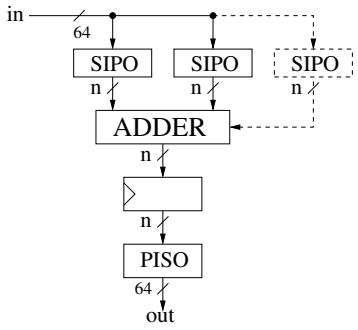

Fig. 5. Test circuit for benchmarking adders and modular adders. Notation: SIPO - Serial In Parallel Out unit, PISO - Parallel In Serial Out unit.

1) the delay of the GPS unit (from a_0, b_0 to p in Fig. 3a),
2) the delay of the Parallel Prefix Network (the longest delay from g_i, p_i to pc_i in Figs. 2c,d),
3) the delay of the Sum unit, S (from pc to r_3 in Fig. 3b)

In general, with an increase in word size w, the contributions of 1 and 3 increase, while the contribution of 2 decreases. The selection of the optimal word size - w, for a given size of operands, is FPGA device dependent for the following reasons:

- the optimal word size depends on the delay of a fast carry chain in a given FPGA device,
- the PPN delay is related to the internal structure of the basic FPGA logic cells (LUTs).

V. Architecture of a Novel Long-Operand Modular Adder

Any adder can be easily extended to a modular adder by using the following procedure [6]: To calculate the result of the n-bit modular addition ($R = A + B \pmod P$), two intermediate values must be computed: $A+B$ and $A+B-P$. The equation (1) demonstrates how to select the final result from the above intermediate values.

$$R = \begin{cases} A+B-P, & \text{if } A+B \geq 2^n \vee A+B-P \geq 0 \\ A+B, & \text{otherwise} \end{cases} \quad (1)$$

This concept is illustrated in Fig. 4. Both additions in Fig. 4 can be implemented using any type of a non-redundant adder: ripple carry, Kogge-Stone, Brent-Kung, and many others. The biggest advantage of the utilization of our novel adder is that the two pure additions required for the modular addition can be overlapped in time, i.e., the second addition can start a long time before the first addition is completed.

An optimized modular adder based on our basic adder described in Section IV is shown in Fig. 6. This adder consists of two layers of GPS units, two PPNs, a layer of MUXes, and a layer of sum units. The GPS units from the top layer operate exactly the same as those in the basic adder. The GPS units from the layer below are slightly different. They take as inputs: the sum signal from the GPS immediately above, the carry out (generate) signal from the less significant GPS above, and the two's complement of the modulus P, $IP = 2^n - P$. Taking into account that compared to the pure GPS, these units have one more input, carry in, we denote them by GPSc (generate-propagate-sum with carry). The internal structure of a GPSc implemented in a Xilinx FPGA is shown in Fig. 3a.

In our modular adder shown in Fig. 6, the critical path includes the delay of two GPS units, the delay of a single PPN, and the delays of an OR gate, a 2-to-1 MUX, and a w-bit adder. As a result, the delay of one of the two PPNs does not influence the critical path.

VI. Generalization for Addition/Subtraction and Modular Addition/Subtraction

Our basic adder can be easily extended to an adder/subtractor by conditionally one's complementing the operand B and adding one at the least significant position. The latter of these two operations can be accomplished by replacing GPS by GPSc at position 0 (in Fig. 2b), and setting the carry input of this GPSc conditionally to 1. The internal structure of GPSc, when implemented using fast carry chains of Xilinx FPGAs is shown in Fig. 3a.

Our generic modular adder (shown in Fig. 4) can be easily extended to an adder/subtractor by the approach shown in Fig. 7. The middle diagram in this figure represents modular subtraction, $R = A - B \pmod P$. This subtraction is described by equation (2).

$$R = \begin{cases} A-B+P, & \text{if } A-B<0 \\ A-B, & \text{otherwise} \end{cases} \quad (2)$$

The left diagram (for modular addition) and the middle diagram (for modular subtraction) can be combined together into the right diagram for modular addition/subtraction. The select signal SUB is used to choose between these two operations.

In the optimized block diagram of the high-radix PPN modular adder from Fig. 6, the following changes would need to be made:

1) $B = b_{N-1}..b_0$ should be multiplexed with the one's complement of B (using the select signal SUB)
2) $IP = 2^n - P = ip_{N-1}..ip_0$ should be multiplexed with P (using the select signal SUB)
3) the top GPS unit at position 0 should be replaced by a GPSc, and its c input connected to SUB
4) the select signal sel should be calculated using the modified logic shown in Fig. 7 with cout#1 replaced by fpc_N and cout#2 replaced by spc_N.

VII. Development and Testing Methodology

All adders described in this paper, including adders from [12], have been described in VHDL-93, using a consistent coding style in order to eliminate any differences that could originate from different coding styles and optimization techniques. In case of the AAM architecture from [12], the choice of the adder block sizes (i.e., the widths of consecutive adder stages) has been based on the detailed instructions provided in the paper.

Due to the limited number of pins in modern FPGAs, we have used a test circuit shown in Fig. 5 for our verification and

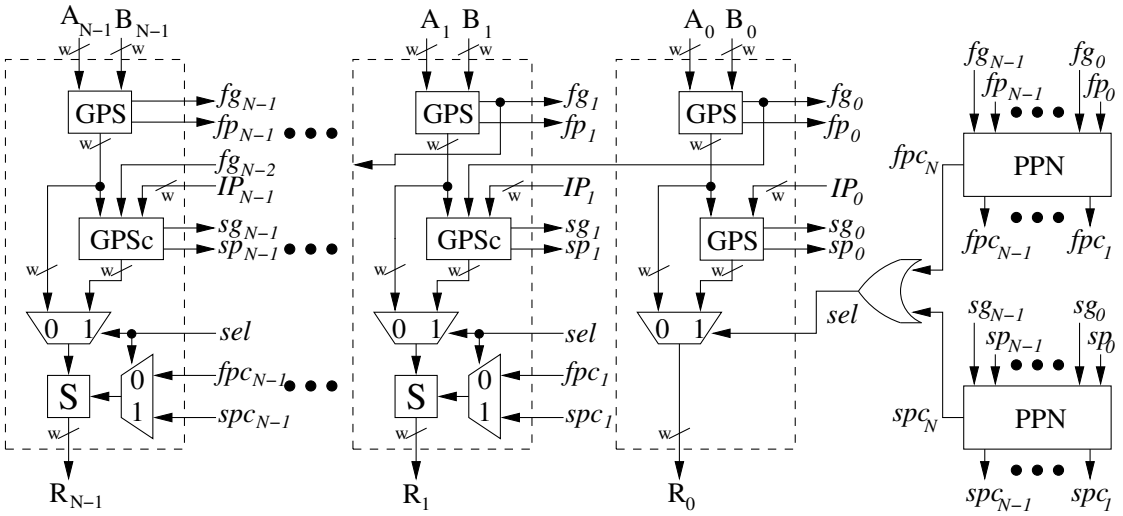

Fig. 6. Proposed novel high-radix Parallel Prefix Network modular adder, performing operation R=A+B mod P. Notation: n - numbers of bits of P, IP = $2^n - P$.

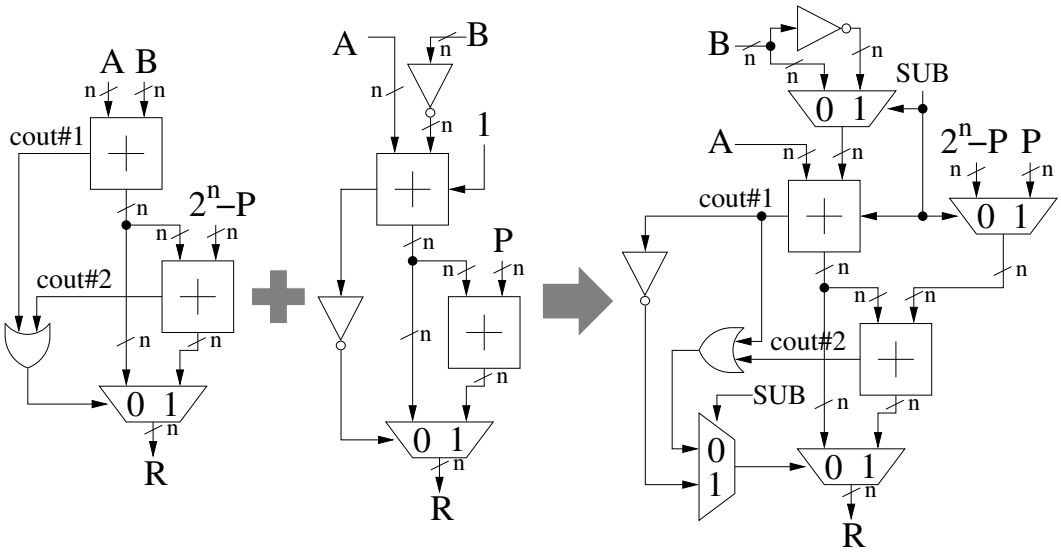

Fig. 7. Proposed novel high-radix Parallel Prefix Network modular adder/subtractor, performing operation R=A+B mod P or R=A-B mod P, depending on the value of the control input SUB. Notation: n - numbers of bits of P.

benchmarking runs. In this diagram, SIPO stands for the Serial In Parallel Out unit, and PISO for the Parallel In Serial Out unit. The combinational adder is surrounded by SIPOs at the inputs and a regular register at the output. The goal of our tests is to determine the latency characteristics of all investigated adders for different argument sizes. It should be stressed that in practical applications, such as public key cryptography, long-operand adders will be typically surrounded by other logic, and not connected directly to any FPGA pins. Our tests very accurately determine the maximum clock frequency, and thus the minimum latency of investigated adders as expected in real-life scenarios.

All adders were implemented using two high performance FPGA devices: a 65nm Altera Stratix III and a Xilinx Virtex 5, and two low-cost devices: a 40nm Altera Cyclone IV and a 45nm Xilinx Spartan 6. All architectures have been synthesized, placed and routed using Xilinx ISE 13.1 and Altera Quartus II 12.1, for Xilinx and Altera FPGAs, respectively. Maximum clock frequencies have been determined using the static timing analysis tools provided as part of the respective software packages ($quartus\_sta$ for Altera and $trace$ for Xilinx). The generation of a large number of results was facilitated by an open source benchmarking environment, called ATHENa (Automated Tool for Hardware EvaluatioN) [22].

VIII. Results and Their Analysis

The parameter exploration for the 1024-bit addition and the 1024-bit modular addition is shown in Tables I and II, respectively. The word size, w, is chosen to be a power of two, and is varied between 8 and 128. The number of inputs to the Parallel Prefix Network (PPN), N, is inversely proportional to the word size, and is equal to $N = 1024/w$. Two high-radix designs are considered. One based on the Kogge-Stone PPN

TABLE I. PARAMETERS EXPLORATION FOR THE 1024-BIT ADDITION

adder (w, N)	latency	area	latency · area
Altera Cyclone IV (ep4cgx110df31c7)			
	[ns]	**[LE]**	**[ns · LE $\cdot 10^3$]**
HR-KS (8,128)	11.81	7577	89.5
HR-KS (16,64)	**11.91**	**6931**	**82.6**
HR-KS (32,32)	13.44	6880	92.5
HR-KS (64,16)	16.50	6890	113.7
HR-KS (128,8)	23.58	6929	163.4
HR-BK (8,128)	17.84	6855	122.3
HR-BK (16,64)	16.37	6614	108.3
HR-BK (32,32)	15.36	6800	104.5
HR-BK (64,16)	17.54	6865	120.4
HR-BK (128,8)	24.20	6907	167.1
Altera Stratix III (ep3sl150f1152c2)			
	[ns]	**[ALUT]**	**[ns · ALUT $\cdot 10^3$]**
HR-KS (8,128)	5.91	3393	20.1
HR-KS (16,64)	6.23	2701	16.8
HR-KS (32,32)	6.92	2416	16.7
HR-KS (64,16)	7.81	2386	18.6
HR-KS (128,8)	11.04	2172	24.0
HR-BK (8,128)	5.69	2711	15.4
HR-BK (16,64)	**5.86**	**2535**	**14.8**
HR-BK (32,32)	6.58	2390	15.7
HR-BK (64,16)	8.05	2322	18.7
HR-BK (128,8)	11.18	2157	24.1
Xilinx Spartan 6 (xc6slx150fgg900-3)			
	[ns]	**[LUT]**	**[ns · LUT $\cdot 10^3$]**
HR-KS (8,128)	6.43	3763	24.2
HR-KS (16,64)	8.05	3454	27.8
HR-KS (32,32)	8.15	3123	25.5
HR-KS (64,16)	7.42	3052	22.6
HR-KS (128,8)	8.13	3099	25.2
HR-BK (8,128)	7.10	5155	36.6
HR-BK (16,64)	7.34	3361	24.7
HR-BK (32,32)	**7.19**	**2970**	**21.3**
HR-BK (64,16)	7.68	3027	23.2
HR-BK (128,8)	8.13	3099	25.2
Xilinx Virtex 5 (xc5vlx155tff1738-3)			
	[ns]	**[LUT]**	**[ns · LUT $\cdot 10^3$]**
HR-KS (8,128)	5.32	4019	21.4
HR-KS (16,64)	**4.99**	**3577**	**17.8**
HR-KS (32,32)	5.69	3327	18.9
HR-KS (64,16)	5.79	3248	18.8
HR-KS (128,8)	6.77	3206	21.7
HR-BK (8,128)	5.44	3663	19.9
HR-BK (16,64)	5.40	3501	18.9
HR-BK (32,32)	5.68	3336	19.0
HR-BK (64,16)	5.69	3255	18.5
HR-BK (128,8)	6.77	3206	21.7

TABLE II. PARAMETERS EXPLORATION FOR THE 1024-BIT MODULAR ADDITION

adder (w, N)	latency	area	latency · area
Altera Cyclone IV (ep4cgx110df31c7)			
	[ns]	**[LE]**	**[ns · LE $\cdot 10^3$]**
HR-KS (8,128)	23.28	12525	291.6
HR-KS (16,64)	**23.22**	**10741**	**249.4**
HR-KS (32,32)	25.50	10963	279.5
HR-KS (64,16)	29.35	11046	324.2
HR-KS (128,8)	39.81	11058	440.2
HR-BK (8,128)	31.57	10751	339.4
HR-BK (16,64)	29.15	10199	297.3
HR-BK (32,32)	28.10	10718	301.2
HR-BK (64,16)	29.90	10970	328.1
HR-BK (128,8)	39.68	11083	439.8
Altera Stratix III (ep3sl150f1152c2)			
	[ns]	**[ALUT]**	**[ns · ALUT $\cdot 10^3$]**
HR-KS (8,128)	11.72	6734	78.9
HR-KS (16,64)	12.41	5352	66.4
HR-KS (32,32)	13.57	4810	65.3
HR-KS (64,16)	14.89	4704	70.1
HR-KS (128,8)	19.02	4335	82.4
HR-BK (8,128)	11.54	5568	64.2
HR-BK (16,64)	**12.08**	**4838**	**58.5**
HR-BK (32,32)	13.84	4612	63.8
HR-BK (64,16)	15.29	4579	70.0
HR-BK (128,8)	19.04	4307	82.0
Xilinx Spartan 6 (xc6slx150fgg900-3)			
	[ns]	**[LUT]**	**[ns · LUT $\cdot 10^3$]**
HR-KS (8,128)	12.63	8685	109.7
HR-KS (16,64)	10.69	6560	70.1
HR-KS (32,32)	10.07	6696	67.5
HR-KS (64,16)	11.34	6031	68.4
HR-KS (128,8)	**10.61**	**5692**	**60.4**
HR-BK (8,128)	9.08	8160	74.1
HR-BK (16,64)	11.30	6707	75.8
HR-BK (32,32)	10.46	6397	66.9
HR-BK (64,16)	11.34	6031	68.4
HR-BK (128,8)	10.61	5692	60.4
Xilinx Virtex 5 (xc5vlx155tff1738-3)			
	[ns]	**[LUT]**	**[ns · LUT $\cdot 10^3$]**
HR-KS (8,128)	8.04	7076	56.9
HR-KS (16,64)	7.59	6713	50.9
HR-KS (32,32)	8.35	6076	50.8
HR-KS (64,16)	**6.82**	**5689**	**38.8**
HR-KS (128,8)	8.37	5514	46.2
HR-BK (8,128)	8.96	6312	56.6
HR-BK (16,64)	7.86	5919	46.5
HR-BK (32,32)	7.45	5750	42.8
HR-BK (64,16)	8.78	6218	54.6
HR-BK (128,8)	8.37	5514	46.2

and the other based on the Brent-Kung PPN. The best designs, for a particular FPGA family, in terms of the product of latency times area are highlighted in bold.

We verified that for traditional PPN adders, the Brent-Kung adder is always significantly better than the Kogge-Stone adder in terms of the product of latency and area, independent of the selected FPGA family. The same is not true for high-radix PPN adders. It is difficult to predict the better of the two PPN choices without a proper design space exploration. We summarized these choices in Table I. For 1024-bit adders, HR-KS is a better design for Cyclone IV and Virtex 5, while HR-BK performs better for Stratix III and Spartan 6. The optimum value of the word size w is 16 for three out of four FPGA families, with the exception of Spartan 6, where the optimum value is equal to 32.

The best choices of adder types and word sizes carry from the basic adders to modular adders in case of Cyclone IV and Stratix III (see Table II). For Spartan 6, the adder type changes from HR-BK for an optimal adder to HR-KS for an optimal modular adder. For Virtex 5, the adder type remains the same, but the optimum word size changes significantly from 16 to 64.

The obtained results indicate no clear pattern, and as a result, we recommend performing the similar space exploration for each new operand size, n, and FPGA family. Our VHDL code has been parameterized so changing an adder type or a word size is equivalent to simply changing values of two top-level generics or constants.

In Tables III and IV, we present the comparison between the implementation results of the best traditional PPN design (Brent-Kung), the best previous work design based on [12], and the best high-radix PPN design (selected through the PPN type and the word size space exploration). In these tables Δ BK represents a relative change of the specific performance measure compared to the traditional Brent-Kung PPN adder, and Δ AAM a relative change compared to the AAM architecture from [12]. A standard carry chain adder (CCA), implemented as a (+) in VHDL, is also included in Table III for comparison. Our experiments have shown that the standard carry chain adder consistently performed significantly worse, in terms of latency and the product latency·area, than any PPN adder for the operand size of 1024 bits.

A. Comparison of basic adders

With the exception of Cyclone IV, the best high-radix PPN adder has a similar latency as the traditional Brent-Kung adder.

TABLE III. BEST IMPLEMENTATION RESULTS FOR THE 1024-BIT ADDITION

adder (w, N)	latency	area	latency · area
	[ns]	[LE/ALUT/LUT]	[ns · LE/ALUT/LUT $\cdot 10^3$]
Altera Cyclone IV (ep4cgx110df31c7)			
Brent-Kung	15.11	7315	110.5
AAM	11.00	7034	77.4
CCA	68.78	6173	424.6
HR-KS (16,64)	11.91	6931	82.6
Δ BK	-21.2	-5.2	-25.3
Δ AAM	+8.3	-1.5	+6.7
Δ CCA	-82.7	+12.3	-80.6
Altera Stratix III (ep3sl150f1152c2)			
Brent-Kung	5.82	3063	17.8
AAM	6.12	2216	13.6
CCA	32.31	1026	33.2
HR-BK (16,64)	5.86	2535	14.8
Δ BK	+0.7	-17.2	-16.7
Δ AAM	-4.2	+14.4	+9.5
Δ CCA	-81.9	+147.1	-55.2
Xilinx Spartan 6 (xc6slx150fgg900-3)			
Brent-Kung	7.48	7757	58.0
AAM	7.80	3889	30.3
CCA	N/A	N/A	N/A
HR-BK (32,32)	7.19	2970	21.3
Δ BK	-3.9	-61.7	-63.2
Δ AAM	-7.8	-23.6	-29.6
Δ CCA	N/A	N/A	N/A
Xilinx Virtex 5 (xc5vlx155tff1738-3)			
Brent-Kung	4.97	6021	29.9
AAM	5.54	4109	22.8
CCA	25.23	1985	50.1
HR-KS (16,64)	4.99	3577	17.8
Δ BK	+0.4	-40.6	-40.4
Δ AAM	-9.9	-12.9	-21.6
Δ CCA	-80.2	+80.2	-64.4

TABLE IV. BEST IMPLEMENTATION RESULTS FOR THE 1024-BIT MODULAR ADDITION

adder (w, N)	latency	area	latency · area
	[ns]	[LE/ALUT/LUT]	[ns · LE/ALUT/LUT $\cdot 10^3$]
Altera Cyclone IV (ep4cgx110df31c7)			
Brent-Kung	25.41	12331	313.3
AAM	24.61	11569	284.7
HR-KS (16,64)	23.22	10741	249.4
Δ BK	-8.6	-12.9	-20.4
Δ AAM	-5.6	-7.2	-12.4
Altera Stratix III (ep3sl150f1152c2)			
Brent-Kung	12.21	7576	92.5
AAM	12.39	6205	76.9
HR-BK (16,64)	12.08	4838	58.5
Δ BK	-1.0	-36.1	-36.8
Δ AAM	-2.5	-22.0	-24.0
Xilinx Spartan 6 (xc6slx150fgg900-3)			
Brent-Kung	28.64	12086	346.2
AAM	19.21	7371	141.6
HR-KS (128,8)	10.61	5692	60.4
Δ BK	-63.0	-52.9	-82.6
Δ AAM	-44.8	-22.8	-57.4
Xilinx Virtex 5 (xc5vlx155tff1738-3)			
Brent-Kung	15.79	10431	164.7
AAM	12.87	7319	94.2
HR-KS (64,16)	6.82	5689	38.8
Δ BK	-56.8	-45.5	-76.4
Δ AAM	-47.0	-22.3	-58.8

For Cyclone IV, the advantage of the high-radix adder exceeds 20%. The improvement in terms of area is between 5% and 17% for Altera families, and between 40% and 62% for Xilinx FPGA families. For the latency times area product, once again the advantage is smaller for Altera families (25% and 17%, respectively), and larger for Xilinx families (62% and 40%, respectively). Compared to the AAM architecture from [12], the latency of our adder is comparable for all investigated families. The area improves for three out of four families. The product latency times area is significantly (20%-30%) better for Xilinx families, and only slightly (less than 10%) worse for Altera families.

B. Comparison of modular adders

For modular addition, our high-radix PPN adders substantially outperform the traditional Brent-Kung adder in terms of all three performance measures. In terms of latency, the gain varies from 1% for Stratix III to 63% for Spartan 6. In terms of area, the gain is between 13% for Cyclone IV and 53% for Spartan 6. In terms of the product latency times area, the gain varies from 20% for Cyclone IV to 83% for Spartan 6. In general, our high-radix adders offer very substantial performance gains, in terms of all performance measures, on Xilinx devices, and medium gains on Altera devices. Compared to the AAM architecture from [12], the latency of our adder is comparable for Altera FPGA families and 44%–47% better for Xilinx FPGA families. The area improves by at least 20% for Stratix III, Spartan 6, and Virtex 5, and stays almost the same for Cyclone IV. The product latency times area is about 60% better for Xilinx families, 12% better for Cyclone IV, and 24% better for Stratix III.

C. Comparison of modular adders, subtractors, and adders/subtractors

As summarized in Table V, modular high-radix PPN subtractors offer very similar performance to modular high-radix PPN adders. The performance penalty for adding subtraction on top of addition is also relatively small. In terms of the product of latency times area, this penalty does not exceed 30% in any investigated FPGA family. For Stratix III, this penalty is particularly small and remains below 10%.

IX. PROPOSED NEW DEDICATED RESOURCES OF MODERN FPGAS

Our basic high-radix Parallel Prefix Network adder, shown in Fig. 2b, can be implemented almost entirely using the hardwired fast carry chains of modern FPGAs, demonstrated schematically in Fig. 1. The only part of this adder which cannot be implemented using such resources is a Parallel Prefix Network. As a result, it seems natural to consider an extension of modern FPGAs with hardwired PPNs. The optimal sizes of PPNs required by our 1024-bit adders are between 8 and 64 bits (see optimum values of N in Tables I and II). These PPNs could be used for the implementation of traditional fast adders shown in Figs. 2a,c,d, as well as our new adders proposed in this paper shown in Figs. 2b, 6, and Fig. 7.

In order to allow further speed-up by parallel execution of multiple additions at the same time, these PPN should be equipped with pipeline registers, which can be either activated or bypassed (similar to the registers present in DSP units of modern FPGAs). By an appropriate choice of the word size w and the number of pipeline stages, very long-operand adders could benefit from fixed-size PPNs and require limited resources of modern FPGAs. These hardwired PPNs could be also used to substantially speed up arithmetic operations performed on standard-size operands (such as 32 and 64 bits) used in multiple scientific and engineering applications, such as digital signal processing, computer graphics, and communications.

TABLE V. SELECTED BEST IMPLEMENTATION RESULTS FOR THE 1024-BIT MODULAR ADDITION, SUBTRACTION AND ADDITION/SUBTRACTION UNIT

	adder (w, N)	latency	area	latency · area	Δ latency	Δ area	Δ latency · area
		[ns]	[LE/ALUT/LUT]	[ns · LE/ALUT/LUT $\cdot 10^3$]	[%]	[%]	[%]
Altera Cyclone IV (ep4cgx110df31c7)							
HR-KS (16, 64)	Adder	23.22	10741	249.4	-	-	-
	Subtractor	23.04	10750	247.7	-0.8	+0.1	-0.7
	Add/Sub	25.47	11881	302.6	+9.7	+10.6	+21.3
Altera Stratix III (ep3sl150f1152c2)							
HR-BK (16, 64)	Adder	12.08	4838	58.5	-	-	-
	Subtractor	11.65	5069	59.0	-3.6	+4.8	+1.0
	Add/Sub	13.02	4830	62.9	+7.7	-0.2	+7.6
Xilinx Spartan 6 (xc6slx150fgg900-3)							
HR-KS (32, 32)	Adder	13.12	6068	79.6	-	-	-
	Subtractor	14.12	5882	83.1	+7.6	-3.1	+4.3
	Add/Sub	16.57	6135	101.7	+26.3	+1.1	+27.7
Xilinx Virtex 5 (xc5vlx155tff1738-3)							
HR-KS (64, 16)	Adder	9.62	5657	54.4	-	-	-
	Subtractor	9.70	6335	61.4	+0.9	+12.0	+12.9
	Add/Sub	9.51	6509	61.9	-1.2	+15.1	+13.7

X. CONCLUSIONS

In this paper, we have presented a novel, low-latency family of high-radix Parallel Prefix Network adders and modular adders. These adders take advantage of the fast carry chains of modern FPGAs. Two different designs based on the Kogge-Stone and Brent-Kung PPNs were developed, implemented, investigated, and compared with previous work. An extensive paramater exploration has been performed, and the best designs and word sizes have been selected for four modern FPGA families from Xilinx and Altera.

Our basic adders have been shown to match the traditional PPN adders in terms of latency, and to outperform them in terms of area and the product latency times area for all four investigated FPGA families. Our adders also outperform the best FPGA-optimized adders presented at FPL 2011 [12] in terms of the product latency times area by more than 20% on Xilinx FPGAs.

For our modular adders, the gains are even more substantial and more consistent across the FPGA families. Our adders outperform the traditional PPN adders in terms of all performance measures. In particular, the gain in terms the product latency times area is between 20% and 83%, depending on the FPGA family. The corresponding gain compared to the modular adder based on the AAM adder architecture from [12] varies between 12% and 63%.

The presented adders and modular adders can also be very easily extended, with a very small performance penalty, to a combined adder/subtractor. Their performance could be also further enhanced by embedding medium-size hardwired PPN structures in the next generation of high-performance FPGAs.

REFERENCES

[1] S. A. Khan, *Digital Design of Signal Processing Systems: A Practical Approach*. Wiley, 2011.

[2] C. Paar and J. Pelzl, *Understanding Cryptography: A textbook for Students and Practitioners*. Springer, 2010.

[3] C. K. Koç, Ed., *Cryptographic Engineering*. Springer, 2009.

[4] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. Oxford University Press, 2009.

[5] M. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann, 2003.

[6] J.-L. Beuchat, "Some modular adders and multipliers for field programmable gate arrays," in *Parallel and Distributed Processing Symposium*, 2003.

[7] S. Knowles, "A family of adders," in *15th IEEE Symposium on Computer Arithmetic*, 2001, pp. 277–281.

[8] F. de Dinechin, H. D. Nguyen, and B. Pasca, "Pipelined FPGA adders," Ecole Normale Superieure de Lyon, Tech. Rep., 2010.

[9] ——, "Pipelined FPGA adders," in *2010 International Conference on Field Programmable Logic and Applications (FPL)*, 2010, pp. 422–427.

[10] T. Preusser and R. Spallek, "Mapping basic prefix computations to fast carry-chain structures," in *2009 International Conference on Field Programmable Logic and Applications (FPL)*, 2009, pp. 604–608.

[11] S. Ghosh, D. Mukhopadhyay, and D. R. Chowdhury, "High speed Fp multipliers and adders on FPGA platform," in *Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2010.

[12] H. Nguyen, B. Pasca, and T. Preusser, "FPGA-specific arithmetic optimizations of short-latency adders," in *2011 International Conference on Field Programmable Logic and Applications (FPL)*, 2011, pp. 232–237.

[13] D. F. Aranha, P. S. L. M. Barreto, P. Longa, and J. E. Ricardini, "The realm of pairings," in *20th International Workshop on Selected Areas in Cryptography (SAC 2013)*, Burnaby, 2013.

[14] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. dissertation, Swiss Federal Institute of Technology Zurich, 1997.

[15] K. Vitoroulis and A. J. Al-Khalili, "Performance of Parallel Prefix Adders implemented with FPGA technology," in *IEEE Northeast Workshop on Circuits and Systems NEWCAS*, 2007, pp. 498–501.

[16] H. Parandeh-Afshar, A. Neogy, P. Brisk, and P. Ienne, "Compressor tree synthesis on commercial high-performance FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, December 2011.

[17] A. K. Verma and P. Ienne, "Improved use of the carry-save representation for the synthesis of complex arithmetic circuits," in *IEEE/ACM International Conference on Computer Aided Design (ICCAD), 2004*, 2004.

[18] F. de Dinechin, H. D. Nguyen, and B. Pasca, "Generating high-performance custom floating-point pipelines," in *2009 International Conference on Field Programmable Logic and Applications (FPL)*, 2009, pp. 59–64.

[19] S. Hauck, M. M. Hosler, and T. W. Fry, "High-Performance Carry Chains for FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 138 – 147, Apr. 2000.

[20] P. Kogge and H. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, 1973.

[21] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, Mar 1982.

[22] K. Gaj, J.-P. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, and B. Y. Brewster, "ATHENa – Automated Tool for Hardware EvaluatioN: Toward fair and comprehensive benchmarking of cryptographic hardware using FPGAs," in *20th International Conference on Field Programmable Logic and Applications - FPL 2010*. IEEE, 2010, pp. 414–421.