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

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

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

***Index Terms*—high-radix carry-save adder, FPGA, parallel prefix network, Kogge-Stone, Brent-Kung, ripple carry adder**

I. Introduction

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

Numerous hardware architectures of adders have been proposed, investigated and optimized for various applications and implementation platforms [6], [7], [8], [9], [10], [11], [12] and [13]. 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 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 [14], Diffie-Hellman [15], DSA, Elliptic Curve Cryptosystems [16], [17], Pairing Based Cryptosystems [18], etc. 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 in the range between 160 and 15,424 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 perform these operations efficiently, we need to employ all relevant embedded (hardwired) resources of modern FPGAs, as well as to adapt the best hardware architectures of adders proposed earlier for ASICs.

In this paper, we describe and analyze several novel hardware architectures for addition and modular addition, taking advantage of fast carry chains of modern FPGAs, and highly applicable for operations on very long operands in the range of one thousand bits and beyond.

II. Background and Previous Work

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 have already long operands, and therefore are not intended to be connected in series.

A. Fast Carry Chains of Modern FPGAs

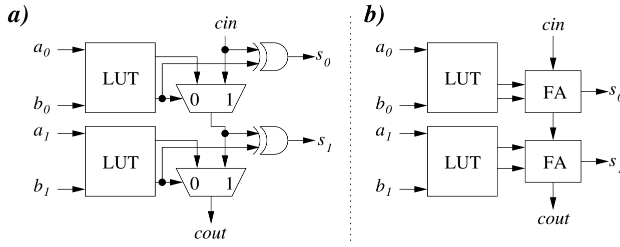

Fig. 1. Fast Carry Chains in a) Xilinx FPGAs, b) Altera FPGAs

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 function of 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

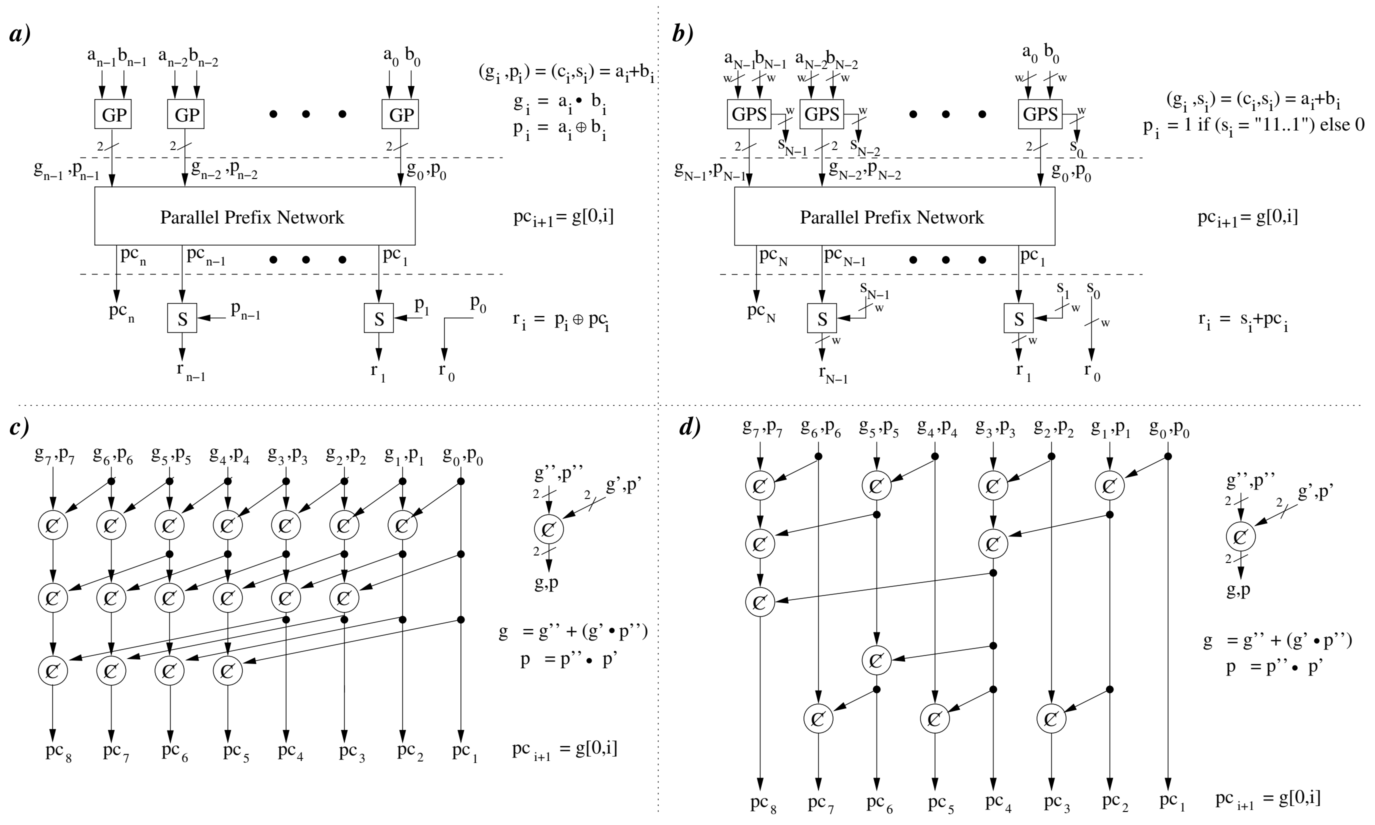

Fig. 2. a) Traditional Parallel Prefix Network adder; + represents an arithmetic addition; b) Proposed novel high-radix Parallel Prefix Network adder; + represents an arithmetic addition; c) Kogge-Stone Parallel Prefix Network for N=8 inputs; + represents logic OR; d) Brent-Kung Parallel Prefix Network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

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 $i+1$. As a result, the generate outputs from PPN correspond to projected carries at positions from 1 to N. 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. 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 [19], and the Brent-Kung PPN (Fig. 2d) for the minimum product of latency times area [20].

Parellel 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.

C. High-Radix Carry-Save Representation

Carry-save adders have been introduced originally as a means of performing fast multi-operand addition [21]. They can be also used for an efficient implementation of a sequence of consecutive additions of long numbers. In such implementation, the conversion to the non-redundant representation needs to happen only once, at the very end of the entire sequence of carry-save additions. This technique has been used to develop one of the fastest reported Montgomery multipliers for long operands in excess of 1024 bits [22]. However, this method is not applicable for the cases where additions are interleaved with other operations. On top of that, this kind of addition does not take advantage of fast carry chains of modern FPGAs.

In the high-radix carry-save representation, each word of an output of a long-operand addition is represented using a w-bit sum word and a single-bit carry. The high-radix carry-save addition can be implemented using a top layer of the circuit shown in Fig. 2b. Although by itself not sufficient to implement a generic fast adder, this idea can be combined with the concept of Parallel Prefix Networks, as described in Section III.

A high-radix carry-save representation has been investigated in [23] from the point of view of its application to modular multiplication of large operands using FPGAs, but it has not been earlier explored in the context of long operand addition and modular addition using fast carry chains of modern FPGAs.

III. Architecture of a Novel Long-Operand Adder

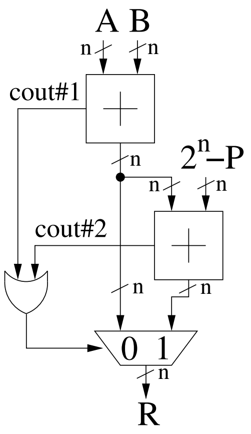

Fig. 4. Modular adder performing operation R=A+B mod P. Notation: n - number of bits of P.

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, propagate signal for the same block of w bits - p_i, and the sum - 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 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 the sum minus a sequence of w ones is greater or equal to zero (i.e., the result of this subtraction stays within a range for unsigned numbers). Equivalently, we may also check whether adding one to the sum produces carry out. Both calculations 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 [19] type

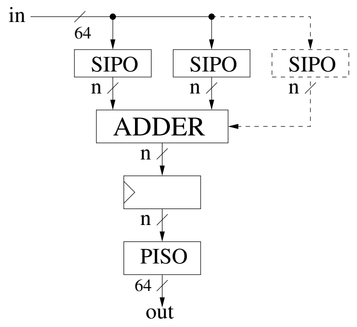

Fig. 5. Test circuit for benchmarking adders and modular adders. Notation: SIPO - Serial In Parallel Out unit, PISO - Parallel In Serial Out unit.

TABLE I
Parameters exploration for the 1024-bit addition

adder (w, N)	**latency**	**area**	**latency · area**
Altera Cyclone IV (ep4cgx110df31c7)			
	[ns]	**[LE]**	**[ns · LE ·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 ·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 ·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 ·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

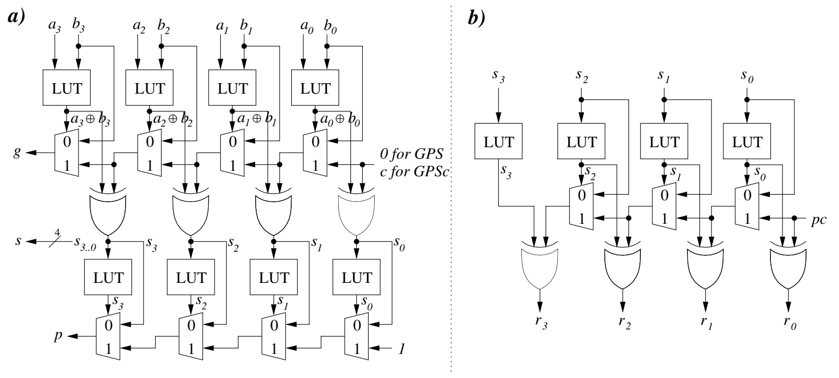

Fig. 3. Components of the proposed adder and modular adder, implemented using fast carry chains of Xilinx FPGAs: a) GPS (generate-propagate-sum) and GPSc (generate-propagate-sum with carry) unit, b) S (sum) unit.

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 ·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 ·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 ·10^3]**
HR-KS (8,128)	13.42	7758	104.1
HR-KS (16,64)	12.88	6659	85.8
HR-KS (32,32)	**13.12**	**6068**	**79.6**
HR-KS (64,16)	17.85	6027	107.6
HR-KS (128,8)	15.34	5912	90.7
HR-BK (8,128)	13.00	7012	91.2
HR-BK (16,64)	14.18	7039	99.8
HR-BK (32,32)	**13.22**	**6118**	**80.9**
HR-BK (64,16)	18.75	6013	112.7
HR-BK (128,8)	15.34	5912	90.7
Xilinx Virtex 5 (xc5vlx155tff1738-3)			
	[ns]	**[LUT]**	**[ns · LUT ·10^3]**
HR-KS (8,128)	11.75	7349	86.3
HR-KS (16,64)	9.33	6667	62.2
HR-KS (32,32)	9.59	5991	57.4
HR-KS (64,16)	**9.62**	**5657**	**54.4**
HR-KS (128,8)	10.59	5501	58.2
HR-BK (8,128)	9.88	6502	64.2
HR-BK (16,64)	9.22	6322	58.3
HR-BK (32,32)	9.65	6070	58.6
HR-BK (64,16)	**9.78**	**5771**	**56.4**
HR-BK (128,8)	10.59	5501	58.2

(Fig. 2c) or of the Brent-Kung type [20] (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

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 (from g_i, p_i to pc_i in Figs. 2c,d), and
3) the delay of the Sum unit, S (from pc to r_3 in Fig. 3a)

In general, with the increase in the 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, seems to be a very FPGA device dependent factor 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.

IV. ARCHITECTURE OF A NOVEL LONG-OPERAND MODULAR ADDER

Any adder can be quite easily extended to a modular adder by using the following concept demonstrated for instance in [24]. In order to calculate a final result of the n-bit modular addition ($R = A+B \pmod{P}$), two intermediate values have to 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

TABLE III
IMPLEMENTATION RESULTS FOR THE 1024-BIT ADDITION

adder (w, N)	**latency**	**area**	**latency** $\cdot$ **area**	Δ **latency**	Δ **area**	Δ **latency** $\cdot$ **area**
	[ns]	**[LE/ALUT/LUT]**	**[ns** $\cdot$ **LE/ALUT/LUT** $\cdot 10^3$**]**	**[%]**	**[%]**	**[%]**
Altera Cyclone IV (ep4cgx110df31c7)						
Kogge-Stone	10.79	21799	235.2	-	-	-
Brent-Kung	15.11	7315	110.5	-	-	-
HR-KS (16,64)	11.91	6931	82.6	+10.4	-68.2	-64.9
HR-BK (32,32)	15.36	6800	104.5	+42.4	-68.8	-55.6
Altera Stratix III (ep3sl150f1152c2)						
Kogge-Stone	5.70	14418	82.1	-	-	-
Brent-Kung	5.82	3063	17.8	-	-	-
HR-KS (32,32)	6.92	2416	16.7	+21.4	-83.2	-79.7
HR-BK (16,64)	5.86	2535	14.8	+2.8	-82.4	-81.9
Xilinx Spartan 6 (xc6slx150fgg900-3)						
Kogge-Stone	17.84	12405	221.3	-	-	-
Brent-Kung	7.48	7757	58.0	-	-	-
HR-KS (64,16)	7.42	3052	22.6	-0.7	-60.7	-60.9
HR-BK (32,32)	7.19	2970	21.3	-3.9	-61.7	-63.2
Xilinx Virtex 5 (xc5vlx155tff1738-3)						
Kogge-Stone	5.58	12832	71.6	-	-	-
Brent-Kung	4.97	6021	29.9	-	-	-
HR-KS (16,64)	4.99	3577	17.8	+0.4	-40.6	-40.4
HR-BK (64,16)	5.69	3255	18.5	+14.5	-45.9	-38.1

TABLE IV
IMPLEMENTATION RESULTS FOR THE 1024-BIT MODULAR ADDITION

adder (w, N)	**latency**	**area**	**latency** $\cdot$ **area**	Δ **latency**	Δ **area**	Δ **latency** $\cdot$ **area**
	[ns]	**[LE/ALUT/LUT]**	**[ns** $\cdot$ **LE/ALUT/LUT** $\cdot 10^3$**]**	**[%]**	**[%]**	**[%]**
Altera Cyclone IV (ep4cgx110df31c7)						
Kogge-Stone	25.77	36974	952.9	-	-	-
Brent-Kung	25.41	12331	313.3	-	-	-
HR-KS (16,64)	23.22	10741	249.4	-8.6	-12.9	-20.4
HR-BK (16,64)	29.15	10199	297.3	+14.8	-17.3	-5.1
Altera Stratix III (ep3sl150f1152c2)						
Kogge-Stone	12.63	28540	360.4	-	-	-
Brent-Kung	12.21	7576	92.5	-	-	-
HR-KS (32,32)	13.57	4810	65.3	+11.2	-36.5	-29.4
HR-BK (16,64)	12.08	4838	58.5	-1.0	-36.1	-36.8
Xilinx Spartan 6 (xc6slx150fgg900-3)						
Kogge-Stone	35.68	24802	884.9	-	-	-
Brent-Kung	28.64	12086	346.2	-	-	-
HR-KS (32,32)	13.12	6068	79.6	-54.2	-49.8	-77.0
HR-BK (32,32)	13.22	6118	80.9	-53.8	-49.4	-76.6
Xilinx Virtex 5 (xc5vlx155tff1738-3)						
Kogge-Stone	17.68	31042	548.8	-	-	-
Brent-Kung	15.79	10431	164.7	-	-	-
HR-KS (64,16)	9.62	5657	54.4	-39.1	-45.8	-67.0
HR-BK (64,16)	9.78	5771	56.4	-38.1	-44.7	-65.7

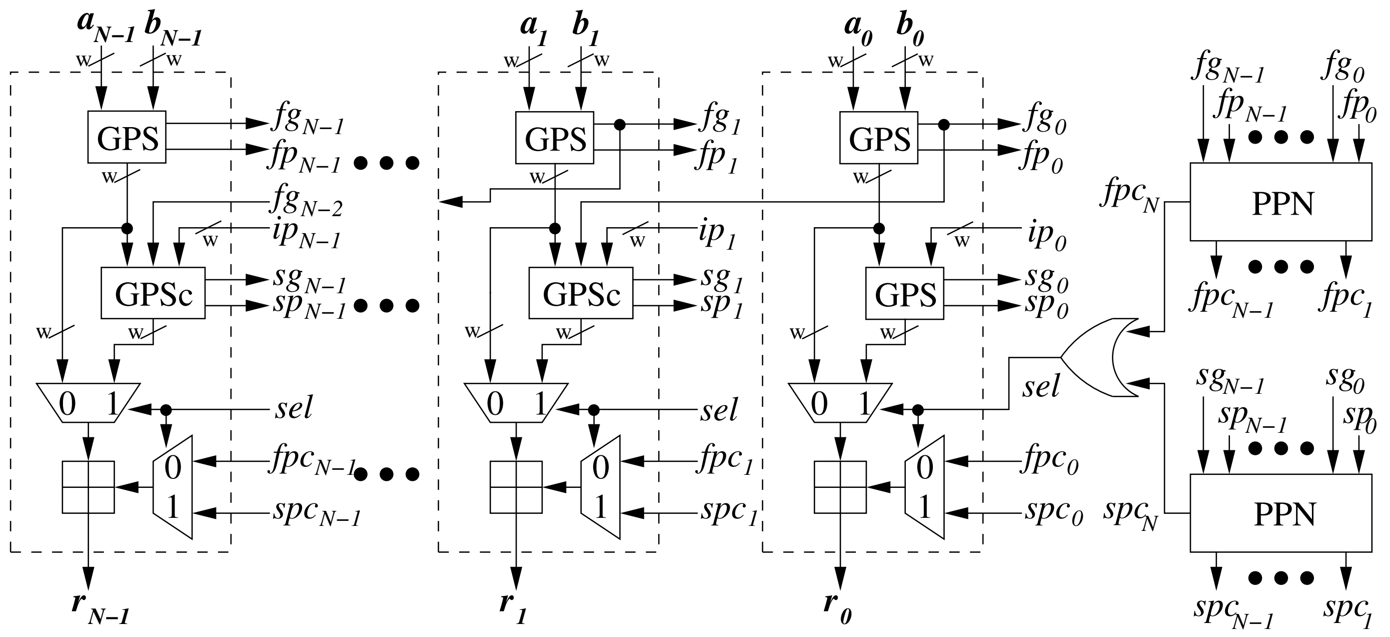

Fig. 6. Proposed novel high-radix Parallel Prefix Network modular adder, performing operation R=A+B mod P. Notation: n - numbers of bits of P, IP = $2^n - P$.

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 III is shown in Fig. 6. This adder consists of two layers of the 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 GPSc implemented using internal resources of Xilinx FPGAs 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, 2-to-1 MUX, and an w-bit adder. As a results, the delay of one of the two PPNs does not influence the critical path.

V. Testing Methodology

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

All adders were implemented on two high performance FPGA devices: 65nm Altera Stratix III and Xilinx Virtex 5, and two low-cost devices: 65nm Altera Cyclone III and 45nm Xilinx Spartan 6. All architectures have been modeled in VHDL-93, synthesized, placed and routed using Xilinx ISE 13.1 and Altera Quartus II 11.1, for Xilinx and Altera FPGAs, respectively.

Maximum clock frequencies have been determined using static timing analysis tools provided as a 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), developed at George Mason University [25].

VI. Results and Their Analysis

The parameter exploration for the 1024-bit addition is shown in Table I. From the point of view of the latency, the best word size, w, is equal to 16 for both investigated Altera families (Cyclone III and Stratix III), 16 for Xilinx Spartan 6, and 32 for Xilinx Virtex 5.

The parameter exploration for the 1024-bit modular addition is shown in Table II. From the point of view of the latency, the best word size, w, is equal to 16 for the Altera families. For Xilinx FPGAs, the optimum value of w is either 16 or 32, depending on the design of PPN (Kogge-Stone or Brent-Kung), and the optimization criteria, such as optimization for latency or the product latency times area.

In the Tables III and IV, we present a comparison among the implementation results of two proposed designs, HR-KS and HR-BK, and two fast traditional adders: Kogge-Stone and

Brent-Kung. Δ represents a change compared to the faster of the two traditional adders. These tables summarize the results collected after the *Place-and-Route* and *Fitter*, for addition and modular addition, respectively. All the aforementioned designs are compared in terms of latency, the area utilization, and the latency times area product, using four cutting edge FPGA devices.

Comparison of basic adders

Both classical designs are very difficult to beat in terms of latency across all the selected FPGA devices. At the same time, our designs substantially outperform the traditional adders in terms of the area and the product of latency times area. For the area, the gain is between 2% for Stratix III and 45% for Xilinx Virtex 5. For the product latency times area, the gain is between -13.5% and 52%. Based on Table III, we can also observe that in terms of latency, there is no clear winner.

Comparison of modular adders

For modular addition, our designs substantially outperform all traditional designs in terms of all three performance measures. For the latency, the gain (measured against the best of the two traditional designs) is between 15% for Stratix III and 60% for Spartan 6. For the area, the gain is between 25% for Cyclone III and 60% for Xilinx Virtex 5. For the product latency times area, the gain is between 45% for Stratix III and 81% for Spartan 6.

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

Our basic adder can be easily extended to 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 to 1.

Our generic modular adder (shown in Fig. 4) can be easily extended to adder/subtractor by using 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).

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

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 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.

VIII. 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 32 and 64 bits (see optimum values of N in Tables III and IV).

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 (similarly to registers present in DSP units of modern FPGAs).

By an appropriate choice of the word size w and the number of pipeline stages, even very long-operand adders could benefit from the fixed-size PPNs, taking 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, communications, etc.

IX. Conclusions

In this paper, we have presented a novel, low latency family of high-radix Parallel Prefix Network adder, taking advantage of fast carry chains of modern FPGAs. Two different designs based on the Kogge-Stone and Brent-Kung PPNs were developed, implemented, investigated and eventually compared with their well known predecessors.

For the operand size of n=1024 bits, the optimal choice of the word size w appeared to be either 16 or 32, depending on the FPGA family and the underlying PPN network.

Our basic adders have been shown to match the traditional adders in terms of latency, and to outperform them in terms of area by a factor between 2% and 45%, and in terms of the product of latency times area by a factor between 13% and 52%.

For our modular adders, the gains are much more substantial. These adders outperform the traditional PPN adders in terms of all performance measures. In particular, the gain in terms of latency is between 15% and 60%, in terms of area

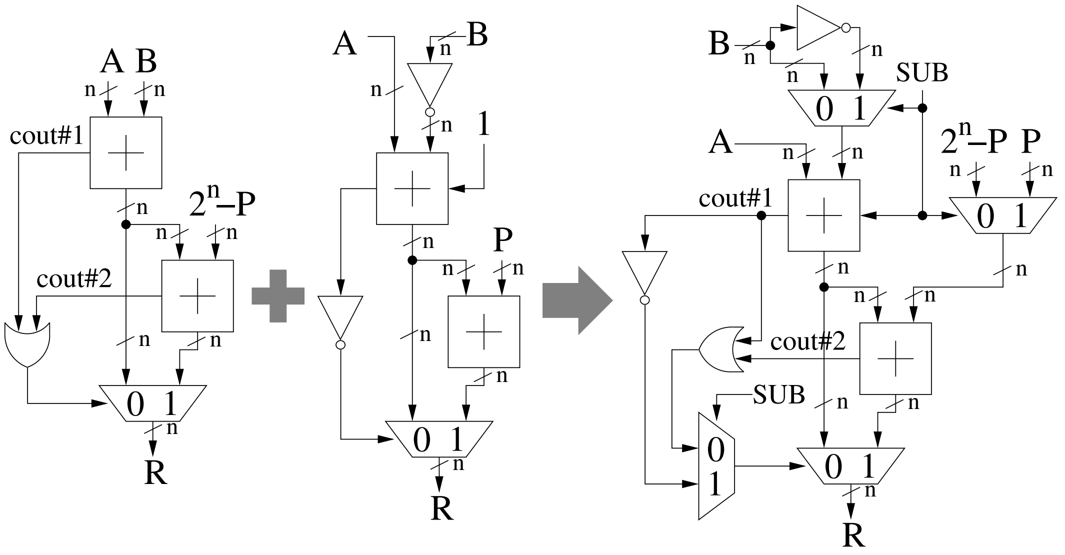

Fig. 7. Proposed novel high-radix Parallel Prefix Network modular adder/subtractor, performing operation R=A+B mod P or R=A-B mod P, depending on the value of the control input SUB. Notation: n - numbers of bits of P.

between 25% and 60%, and in terms of the product of latency time area between 45% and 81%.

The presented adders and modular adders can be also very easily extended to the combined adders/subtractors.

For the future work we are going to investigate the application of the aforementioned adders to efficient implementation of the high level applications, such as coprocessors supporting computations of cryptographic schemes based on the elliptic curve and pairing based cryptography over primes fields.

References

[1] K. Parhi, *VLSI digital signal processing systems: design and implementation.* John Wiley & Sons, 1999.

[2] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, Mar. 1965. [Online]. Available: http://dx.doi.org/10.1147/sj.41.0025

[3] X. Wu, "An efficient antialiasing technique," *SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 143–152, Jul. 1991. [Online]. Available: http://doi.acm.org/10.1145/127719.122734

[4] *Secure Hash Standard (SHS)*, National Institute of Standards and Technology (NIST), Oct. 2008, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

[5] X. Lai and J. L. Massey, "A proposal for a new block encryption standard," in *Advances in Cryptology - EuroCrypt '90*, ser. Lecture Notes in Computer Science (LNCS), I. B. Damgård, Ed., vol. 473. Berlin: Springer-Verlag, 1990, pp. 389–404.

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

[7] 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.

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

[9] T. O. Bachir and J.-P. David, "Performing floating-point accumulation on a modern fpga in single and double precision," in *The 18th Annual International IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM-2010)*, 2010.

[10] H. D. Nguyen, B. Pasca, and T. B. Preusser, "Fpga-specific arithmetic optimizations of short-latency adders," in *2011 International Conference on Field Programmable Logic and Applications (FPL)*, 2011.

[11] 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.

[12] N. Zamhari, P. Voon, K. Kipli, K. L. Chin, and M. H. Husin, "Comparison of parallel prefix adder (ppa)," in *World Congress on Engineering WCE 2012*, vol. II, July 2012.

[13] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, 2000.

[14] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb 1978.

[15] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, Nov 1976.

[16] V. S. Miller, "Uses of elliptic curves in cryptography," in *Advances in Cryptology — CRYPTO '85*, ser. Lecture Notes in Computer Science (LNCS), H. C. Williams, Ed., vol. 218. Berlin: Springer-Verlag, 1986, pp. 417–426.

[17] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, Jan 1987.

[18] A. Menezes, "An introduction to pairing-based cryptography," *Recent Trends in Cryptography*, vol. 477, pp. 47–65, 2009.

[19] P. Kogge and H. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Transactions on Computers*, 1973.

[20] 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.

[21] J. G. Earle, "Latched Carry Save Adder Circuit for Multipliers," U.S. Patent 3,340,388, Jul. 1965.

[22] C. McIvor, M. McLoone, and J. McCanny, "Modified Montgomery modular multiplication and RSA exponentiation techniques," in *Computers & Digital Techniques*, ser. IEEE Proceedings, vol. 151, Jul 2004, pp. 402–408.

[23] J.-L. Beuchat and J.-M. Muller, "Automatic Generation of Modular Multipliers for FPGA Applications," *IEEE Transactions on Computers*, vol. 57, no. 12, pp. 1600–1613, 2008.

[24] J.-L. Beuchat, "Some Modular Adders and Multipliers for Field Programmable Gate Arrays," in *Parallel and Distributed Processing Symposium*, 2003.

[25] 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, winner of the FPL Community Award.

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 ·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
HR-BK (16, 64)	Adder	29.15	10199	297.3	-	-	-
	Subtractor	31.55	10118	319.2	+8.2	-0.8	+7.3
	Add/Sub	30.53	11205	342.1	+4.7	+9.9	+15.1
Altera Stratix III (ep3sl150f1152c2)							
HR-KS (32, 32)	Adder	13.57	4810	65.3	-	-	-
	Subtractor	14.10	4814	67.9	+3.9	+0.1	+4.0
	Add/Sub	14.20	4810	68.3	+4.6	+0.0	+4.6
HR-KS (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
HR-BK (32, 32)	Adder	13.22	6118	80.9	-	-	-
	Subtractor	14.79	5807	85.9	+11.9	-5.1	+6.2
	Add/Sub	18.22	6988	127.3	+37.8	+14.2	+57.4
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
HR-BK (64, 16)	Adder	9.78	5771	56.4	-	-	-
	Subtractor	9.60	5829	55.9	-1.9	+1.0	-0.9
	Add/Sub	9.66	5757	55.6	-1.2	-0.2	-1.5

APPENDIX