UDC 004.315

PRIHOZHY A. A.

SYNTHESIS OF PARALLEL ADDERS FROM IF-DECISION DIAGRAMS

Belarusian National Technical University

Addition is one of the timing critical operations in most of modern processing units. For decades, extensive research has been done devoted to designing higher speed and less complex adder architectures, and to developing advanced adder implementation technologies. Decision diagrams are a promising approach to the efficient many-bit adder design. Since traditional binary decision diagrams does not match perfectly with the task of modelling adder architectures, other types of diagram were proposed. If-decision diagrams provide a parallel many-bit adder model with the time complexity of $O(\log_2 n)$ and area complexity of $O(n\times\log_2 n)$. The paper propose a technique, which produces adder diagrams with such properties by systematically cutting the diagram's longest paths. The if-diagram based adders are competitive to the known efficient Brent-Kung adder and its numerous modifications. We propose a blocked structure of the parallel if-diagram-based adders, and introduce an adder table representation, which is capable of systematic producing if-diagram of any bit-width. The representation supports an efficient mapping of the adder diagrams to VHDL-modules at structural and dataflow levels. The paper also shows how to perform the adder space exploration depending on the circuit fan-out. FPGA-based synthesis results and case-study comparisons of the if-diagram-based adders to the Brent-Kung and majority-invertor gate adders show that the new adder architecture leads to faster and smaller digital circuits.

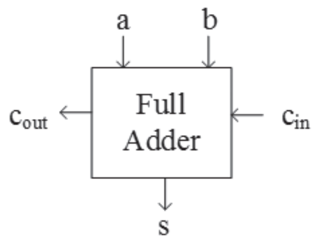
Keywords: *many-bit adders, decision diagrams, time delay, area, VHDL, FPGA, synthesis, adder space exploration.*

Introduction

Arithmetic operations (addition, multiplication and others) are timing critical operations in almost all modern processing units [1–8]. The parameters such as implementation area, adder latency and power dissipation decide the choice of adders for different applications. There is an extensive research attention towards designing higher speed and less complex adder architectures with lower power dissipation. Many works have been devoted to adder architectures and implementation styles. Decision diagram based approaches [9–19] are a promising direction in the efficient adder design. The traditional binary decision diagrams have been extended to functional, biconditional, if-decision and other diagrams, which are more suitable for the adder design and optimization. This work develops an if-diagram based blocked architecture of parallel adders, estimates their time and area complexity, introduces a table method of constructing and VHDL-modelling, and provides a comparison of adders through results of FPGA-synthesis.

Adder design overview

Full adder. A one-bit full-adder adds three 1-bit numbers a, b and c_{in}, and produces two 1-bit numbers s and c_{out} (Figure 1). A full adder can be implemented by $s = a \oplus b \oplus c_{in}$ and $c_{out} = (a \wedge b) \vee (c_{in} \wedge (a \oplus b))$ where $\wedge$, $\vee$ and $\oplus$ are Boolean conjunction, disjunction and exclusive or respectively [1].



a	b	c0	s	c1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig. 1. One-bit full adder

Ripple-carry adder (RCA) adds two n-bit numbers $a = a_{n-1},\dots, a_0$ and $b = b_{n-1},\dots, b_0$, produces a n+1-bit sum $s = s_n,\dots, s_0$, and consists of n full adders (one adder for one bit) (Figure 2). Each full adder inputs c_{in}, which is c_{out} of the previous full adder. RCA is relatively slow and is low-cost. RCA takes O(n) of time for carry to reach the most significant bit, and requires O(n) of cost for implementation.

Carry-lookahead adder (CLA) reduces the computation time against RCA [2]. For each bit

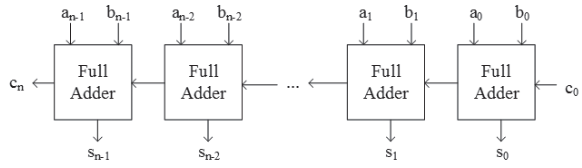


Fig. 2. N-bit ripple carry adder

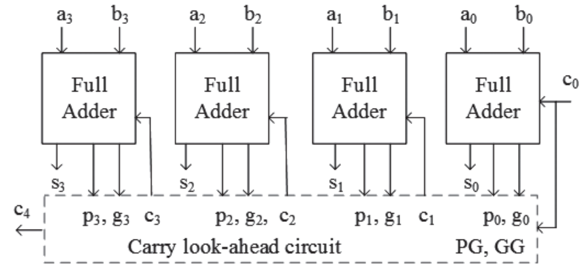


Fig. 3. Four-bit carry-lookahead adder

position, i, it creates two signals: a generation signal $g_i = a_i \wedge b_i$ and a propagation signal $p_i = a_i \oplus b_i$. Signal g_i sets output carry c_{i+1} to 1 regardless of input carry c_i. Signal p_i propagates carry from a less significant bit position.

Value 0 of both inputs kills carry in bit position i. The next-stage carry of CLA is $c_{i+1} = g_i \vee (p_i \wedge c_i)$. Figure 3 depicts a 4-bit CLA. The depth of CLA carry look-ahead circuit is significantly smaller against RCA. Several units of CLA provide the construction of a higher-level and wider-bit circuit.

Kogge–Stone adder (KSA) is a parallel prefix carry look-ahead adder [3]. KSA calculates the generalized propagation signal, P and generation signal, G recurrently in two cases: 1) given g_i and p_i, then $G = g_i$ and $P = p_i$; 2) given two pairs (G_i, P_i) and (G_j, P_j), then pair $(G, P) = (G_i, P_i) \lozenge (G_k, P_k)$ is a composition $\lozenge$ of two pair inputs: $G = (P_i \wedge G_k) \vee G_i$ and $P = P_i \wedge P_k$. The carry and sum signals are $C_i = G_i$ and $S_i = P_i \oplus C_{i-1}$ respectively. Figure 4 depicts the carry part of 8-bit KSA (rectangle represents case 1, and circle represents case 2.

Brent–Kung adder (BKA) reduces against KSA the power consumption and chip area as well as increases the speed [4, 5]. BKA is much quicker than RCA. BKA defines a recurrent computation of (G_i, P_i) in two cases: 1) $(G_i, P_i) = (g_i, p_i)$ for $i = 0$; 2) $(G_i, P_i) = (g_i, p_i) \lozenge (G_{i-1}, P_{i-1})$ for $i = 1, 2, \dots n-1$. It can be seen that

$$(G_{n-1}, P_{n-1}) = (g_{n-1}, p_{n-1}) \lozenge (g_{n-2}, p_{n-2}) \lozenge \dots \lozenge (g_0, p_0).$$

Brent and Kung proved that operator $\lozenge$ is associative, therefore, (G_{n-1}, P_{n-1}) can be computed

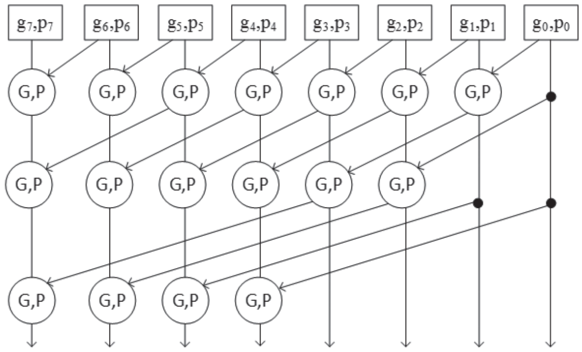


Fig. 4. Carry part of Kogge–Stone 8-bit adder

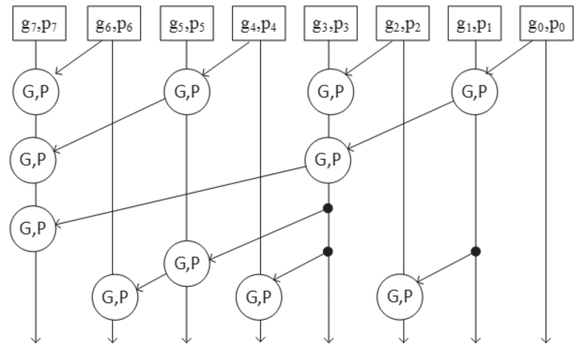


Fig. 5. Carry part of Brent-Kung 8-bit adder

in a tree-like manner. As a result, the addition of n-bit numbers consumes $O(log_2 n)$ of time and consumes $O(n \times log_2 n)$ of area. Figure 5 depicts the carry part of 8-bit BKA.

Majority-invertor gate adder (MIGA) exploits a representation of logic functions by a *majority-inverter* graph [6, 7]. MIGA consists of majority nodes and regular/complemented edges. A function M3(x, y, z) expressed by $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ defines the node semantics. Figure 6 depicts a 1-bit full-adder consisting of three M3 nodes. The authors of [7] optimize MIGAs via a new Boolean algebra. The optimized MIGAs have smaller depth than the original and-invertor adders.

Pass Exclusive-Not-OR gate circuits. Work [8] introduces a new logic style for p-n junction based digital graphene circuits: a pass-XNOR compact energy efficient logic style. The pass-XNOR gate shows a higher expressive power compared to the CMOS counterparts as it requires a smaller number of devices to implement XNOR / XOR-dominated logic functions, in particular adders.

Decomposition of Boolean incompletely specified functions

Works [9, 10] originally propose the concept of if-decision diagram that is a result of the

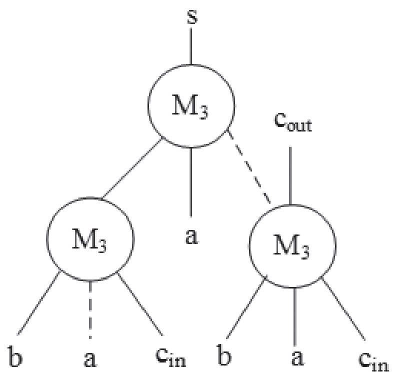


Fig. 6. *Majority-invertor* full-adder (dash line represents complementation

theory ofincompletely specified Boolean functions [11, 12]. Let $B = \{0, 1\}$ and $M = \{0, 1, dc\}$ where 0 and 1 are Boolean values and dc is a don't care value. An incompletely specified Boolean function $\varphi(x)$ of vector Boolean variable $x = (x_1, \ldots, x_n)$ is a mapping φ: $B^n \rightarrow M$. In φ, value $dc \in M$ can be arbitrarily replaced with 0 or 1. Function $\varphi(x)$ can be represented by three sets: on-set ON(φ) where $\varphi(x) = 1$, off-set OFF(φ) where $\varphi(x) = 0$, and don't care set DC(φ) where $\varphi(x) = dc$. Three Boolean characteristic functions describe the sets: $\varphi^{on}(x)$ takes value 1 if $x \in$ ON(φ) and value 0 otherwise; $\varphi^{off}(x)$ takes value 1 if $x \in$ OFF(φ) and value 0 otherwise; $\varphi^{dc}(x)$ takes value 1 if $x \in$ DC(φ) and value 0 otherwise. We call function $f(x) = \varphi^{on}(x)$ a value function, and call function $d(x) = \neg\varphi^{dc}(x)$ a domain function where $\neg$ is Boolean inversion. Pair $(f(x)\,|\,d(x))$ describes the incompletely specified function $\varphi(x)$.

In pair $(f(x)\,|\,d(x))$, we may replace $f(x)$ without changing $\varphi(x)$, by other function $v(x)$ of the slice

$$(f \wedge d)^{on} \subseteq v^{on} \subseteq (f \vee d)^{on} . \qquad (1)$$

Since the functions of slice (1) can produce digital circuits of various time and area, we introduce an operation $v(x) = min\ (f(x)\,|\,d(x))$ to select a best function of the slice [9 - 12]. The following theorem generalizes the widely known Shannon expansion. Let $min(f\,|\,d)$ and $min(f\,|\,\neg d)$ be residual functions (cofactors) of function f on function d.

Theorem. Expansion (2) holds for arbitrarily Boolean functions $f(x)$ and $d(x)$.

$$f = d \wedge \min(f \mid d) \vee \neg d \wedge \min(f \mid \neg d) \qquad (2)$$

Expansion (2) is capable of efficiently solving many optimization problems of digital system design.

If-decision diagrams

The if-decision diagram (IFD) [9] represents expansion (2) by nodes of a directed acyclic-labeled graph. We use it for the modelling, synthesis and optimization of adders. Figure 7 depicts a nonterminal node. Its three child nodes are the if-node d, high-node $g = min(f\,|\,d)$ and low-node $h = min(f\,|\,d)$, which form a node notation $ifd\ (d, g, h)$. A terminal node is either a constant 0, constant 1, variable x_i or its negation $\neg x_i$. IFD is a promising generalization of BDD [13].

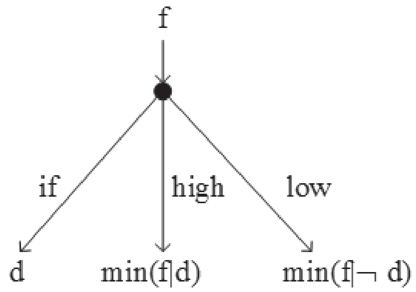


Fig. 7. Nonterminal node of if-decision diagram

Biconditional Binary Decision Diagrams (BBDD) are a special case of IFDs. Two ways are known to infer BBDD. According to work [11] (pages 197–201), if $d = x_i \oplus x_j$ then (2) looks like

$$f = (x_i \oplus x_j) \wedge f_{x_i=\neg x_j} \vee \neg(x_i \oplus x_j) \wedge f_{x_i=x_j} , \quad (3)$$

where $f_{xi=xj}$ and $f_{xi=xj}$ are cofactors (or residual functions) produced by operations $min(f\,|\,x_i \oplus x_j)$ and $min(f\,|\,(x_i \oplus x_j))$ respectively (pages 75–82).

Work [14] introduces BBDD through a biconditional expansion that is similar to expansion (3) and is a special case of the (x_i, p)-decomposition proposed in [15]. The authors provide a one-pass logic synthesis methodology and tool, which combines the logic optimization and technology mapping phases in a single step carried out through a common data structure. The Gemini tool [16] exploits the methodology to synthesize efficient pass-XNOR-gate-based circuits.

Modelling adders by if-decision diagrams

Works [17, 18] propose a method of modelling adders by IFDs. Figure 8 depicts a two-root IFD of 1-bit full adder. The diagram consists of three nonterminal *ifd*-nodes and seven terminal nodes.

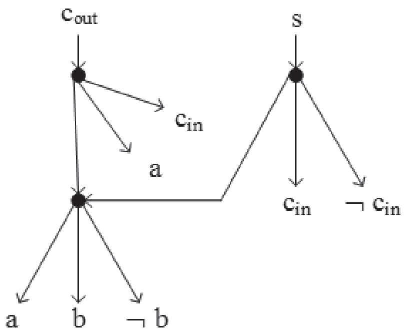


Fig. 8. Two-root IFD of 1-bit full adder

Many-root-IFD is a model for the construction of n-bit ripple carry adders (IFDRCA). Figure 9 depicts a 7-bit IFDRCA. While the advantage of the adder is the low cost (21 nonterminal ifd-nodes), its drawback is a big depth (8 nonterminal nodes).

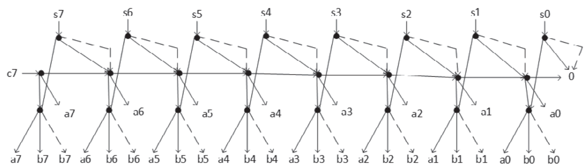


Fig. 9. IFD of 7-bit RCA (dash line is complementation)

From ripple carry to parallel IFD-based adders: transformation technique

In this section, we propose a technique of transforming an IFDRCA to an if-decision diagram based parallel adder (IFDPA) with reduced critical paths. The technique systematically cuts the critical paths of the diagram. We demonstrate the technique on the 7-bit IFDRCA, and in particular on the diagram of carry signal c_2 depicted in Figure 10 (the diagram size is 6 and the depth is 4 ifd-nodes).

Figure 11 (a) depicts a diagram of domain function d_2 that is capable of cutting the critical path in diagram c_2. Figure 11 (b) depicts a diagram that represents an inversion of d_2. It is obtained by applying the inversion operation ¬ to ifd-nodes: ¬ *ifd*(d, g, h) is functionally equivalent to *ifd*(d, ¬ g, ¬ h).

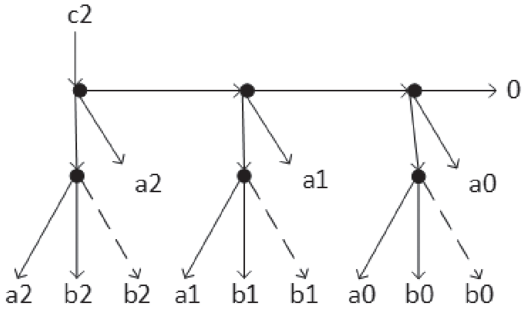


Fig. 10. IFD of carry signal c_2

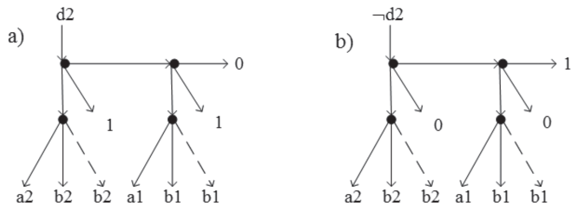


Fig. 11. IFDs of domain functions d_2 and $\neg d_2$

To perform the minimization operation on the IFDs, we provide four reduction rules. Given diagram $f = ifd(e, g, h)$ representing the value function, the result of the minimization operation $min(f|d)$ depends on the diagram that represents the domain function d:

1. If $d = ifd(e, u, 0)$ then $min(f|d) = min(g|u)$
2. If $d = ifd(e, 0, u)$ then $min(f|d) = min(h|u)$
3. If $d = ifd(e, 1, u)$ then
   $min(f|d) = ifd(e, g, min(h|u))$
4. If $d = ifd(e, u, 1)$ then
   $min(f|d) = ifd(e, min(g|u), h)$

Rules 1 and 3 reduce the result (Figure 12, (a)) of operation $min(c_2|d_2)$. Rules 2 and 4 simplify the result (Figure 12, (b)) of operation $min(c_2|\neg d_2)$. Assembling these two diagrams together with diagram d_2 and sharing *ifd*-nodes yield the integrated diagram depicted in Figure 13.

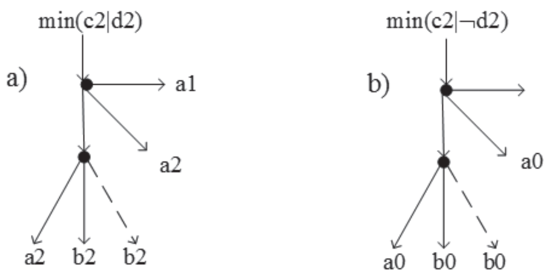


Fig. 12. Products of $min(c_2$ | d2) and $min(c_2 | \neg d_2)$

Observing two diagrams in Figures 10 and 13, we conclude that the second IFD consists of 7 nodes, one more than the first one, but the critical path of the second IFD is one node shorter (3 against 4).

Applying the transformation technique to each long path of the IFD yields a many-root parallel if-decision diagram of the whole adder. Figure 14 depicts an if-diagram of the carry

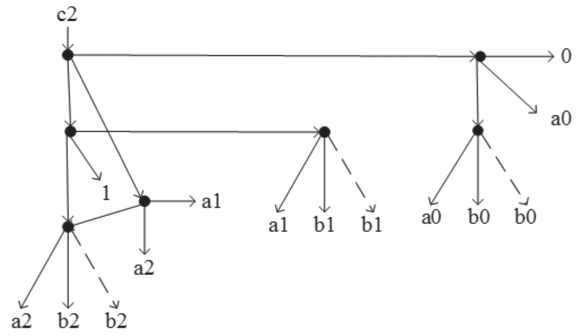


Fig. 13. IFD c_2 after assembling products and sharing nodes

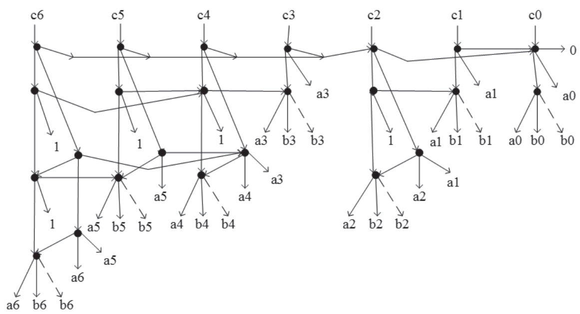


Fig. 14. Many-root IFD of carry part of 7-bit IFDPA

part of 7-bit IFDPA. Its overall size is 24 nodes (for comparison IFDRCA has 14 nodes), but its overall critical path is 4 nodes (IFDRCA has 8 nodes). Note, that the size and depth of the BDD-based adder [13] is much larger than those of IFDPA.

Blocked structure of IFDPA

The transformation technique is capable of producing an IFDPA for any n-bit width. All IFDPAs have the same structure. It consists of a chain of blocks (Figure 15); the bit-width of a block is twice larger than the width of the right-hand neighbor block. A scalar carry signal connects the neighbor blocks. The block widths from the right to the left are 1, 2, 4, 8, 16, 32 … bit. The critical path (depth) of the blocks is 2, 3, 4, 5, 6, 7 … node respectively. The largest number of nodes per bit in the blocks is 3, 5, 7, 9, 11, 13 … respectively.

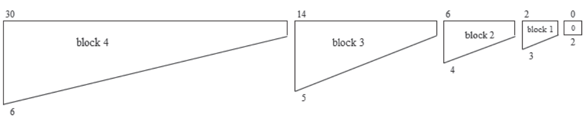


Fig. 15. Blocks of 31-bit IFDPA

Table representation of IFDPA

To represent the many-root IFDPA consisting of full blocks, we introduce a matrix $M[R×C]$ of *ifd*-nodes, where $C = width + 1$ is the number of columns, and $R = 3 + 2 × (depth − 2)$ is the number of rows. Figure 16 depicts matrix M of the 15-bit IFDPA constructed of four blocks k_0, k_1, k_2 and k_3. It consists of cells describing *ifd*-nodes. Each cell includes three elements, which are arguments of function *ifd*. The argument can be a reference (edge) to other cell (the edge indicates other nonterminal node), constants 0 and 1, and variables a_i and b_i (these elements are associated with terminal nodes of IFD). Cell (0, 0) has a reference to cell (1, 1). Other cells are empty.

The matrix enumerates the columns from the left to the right by 0, … , 15. Contrary, it enumerates adder bits (and associated variables) from the right to the left. The first row cells correspond to sum signals s_0, … , s_{14}, and the second row cells match carry signals c_0, … , c_{14}. All nonempty cells of a column describe nodes performing calculations on the corresponding bit. Specifier '*not*' indicates the complement edge.

Estimation of IFDPA parameters

Work [19] provides efficient methods of computer (digital) system analysis. The size $S^{rc}(n)$ and depth $D^{rc}(n)$ of the n-bit IFDRCA (at $n = 1, 3, 7, 15, 31…$) counted in the number of if-diagram nonterminal nodes can be estimated as

$$S^{rc}(n) = 3 \times n \qquad (4)$$

$$D^{rc}(n) = n + 1 \qquad (5)$$

The size $S^{pr}(n)$ and depth $D^{pr}(n)$ of the n-bit parallel adder is

$$S^{pr}(n) = n + (n+1) \times \log_2(n+1) \qquad (6)$$

$$D^{pr}(n) = 1 + \log_2(n+1) \qquad (7)$$

Table 1 provides a comparison of IFDRCAs against IFDPAs produced by means of transforming the if-diagrams. The diagram depth and size varies depending on the adder bit-width. The gain in the depth (measured in number of nodes) of IFDPAs over the depth of IFDRCAs increases up to 93.1 times for 1023 bit-width. At the same time, IFDPAs' size is larger against IFDRCAs up to 5.0x.

		k3								k2				k1		k0
	c14 0	s14 1	s13 2	s12 3	s11 4	s10 5	s9 6	s8 7	s7 8	s6 9	s5 10	s4 11	s3 12	s2 13	s1 14	s0 15
0	reference (1,1)	(8,1) (1,2) not(1,2)	(6,2) (1,3) not(1,3)	(6,3) (1,4) not(1,4)	(4,4) (1,5) not(1,5)	(6,5) (1,6) not(1,6)	(4,6) (1,7) not(1,7)	(4,7) (1,8) not(1,8)	(2,8) (1,9) not(1,9)	(6,9) (1,10) not(1,10)	(4,10) (1,11) not(1,11)	(4,11) (1,12) not(1,12)	(2,12) (1,13) not(1,13)	(4,13) (1,14) not(1,14)	(2,14) (1,15) not(1,15)	(2,15) 0 1
1		(2,1) (3,1) (1,9)	(2,2) (3,2) (1,9)	(2,3) (3,3) (1,9)	(2,4) (3,4) (1,9)	(2,5) (3,5) (1,9)	(2,6) (3,6) (1,9)	(2,7) (3,7) (1,9)	(2,8) b7 (1,9)	(2,9) (3,9) (1,13)	(2,10) (3,10) (1,13)	(2,11) (3,11) (1,13)	(2,12) b3 (1,13)	(2,13) (3,13) (1,15)	(2,14) b1 (1,15)	(2,15) b0 0
2		(4,1) 1 (2,5)	(4,2) 1 (2,5)	(4,3) 1 (2,5)	(4,4) 1 (2,5)	(4,5) 1 (2,7)	(4,6) 1 (2,7)	(4,7) 1 (2,8)	b7 a7 not a7	(4,9) 1 (2,11)	(4,10) 1 (2,11)	(4,11) 1 (2,12)	b3 a3 not a3	(4,13) 1 (2,14)	b1 a1 not a1	b0 a0 not a0
3		(4,1) (5,1) (3,5)	(4,2) (5,2) (3,5)	(4,3) (5,3) (3,5)	(4,4) b11 (3,5)	(4,5) (5,5) (3,7)	(4,6) b9 (3,7)	(4,7) b8 b7		(4,9) (5,9) (3,11)	(4,10) b5 (3,11)	(4,11) b4 b3		(4,13) b2 b1		
4		(6,1) 1 (4,3)	(6,2) 1 (4,3)	(6,3) 1 (4,4)	b11 a11 not a11	(6,5) 1 (4,6)	b9 a9 not a9	b8 a8 not a8		(6,9) 1 (4,10)	b5 a5 not a5	b4 a4 not a4		b2 a2 not a2		
5		(6,1) (7,1) (5,3)	(6,2) b13 (5,3)	(6,3) b12 b11		(6,5) b10 b9				(6,9) b6 b5						
6		(8,1) 1 (6,2)	b13 a13 not a13	b12 a12 not a12		b10 a10 not a10				b6 a6 not a6						
7		(8,1) b14 b13														
8		b14 a14 not a14														

Fig. 16. Matrix M representing 15-bit parallel adder (filled columns depict 4 blocks)

Table 1. **Depth and size of IFDRCA and IFDPA carry part vs. adder bit-width**

Blocks	Width, bit	IFDRCA		IFDPA		
		Depth	Size	Depth	Size	Fan-out
1	1	2	2	2	2	2
2	3	4	6	3	8	3
3	7	8	14	4	24	5
4	15	16	30	5	64	9
5	31	32	62	6	160	17
6	63	64	126	7	384	33
7	127	128	254	8	896	65
8	255	256	510	9	2048	129
9	511	512	1022	10	4608	257
10	1023	1024	2046	11	10240	513

Table 2. **Depth and size of IFDPFA carry part vs. adder fan-out**

fan-out	Bit-width							
	31		63		127		255	
	depth	size	depth	size	depth	size	depth	size
3	17	92	33	188	65	380	129	764
4	13	100	23	208	45	420	87	848
5	10	120	18	248	34	504	66	1016
6	9	120	16	246	28	504	54	1014
7	8	128	14	264	24	544	46	1096
8	8	130	12	280	22	570	40	1154
9	7	144	11	304	19	624	35	1264
10	7	140	11	294	18	608	32	1234
11	7	140	10	304	17	620	29	1264

Adder space exploration by means of fan-out

Observing the IFDPA shows that the block output carry signal has largest fan-out among other signals, which grows exponentially depending on the block index. Given a constraint, F on the fan-out of n-bit IFDPA, the subject is to reduce the adder depth and / or size, obtaining new fan-out constrained adder denoted IFDPFA. We build IFDPFA consisting of three parts: right, central and left. The central part includes several fan-out constrained blocks of width, F-1. The right and left parts includes blocks of the bit-width smaller than F-1. For instance, Figure 17 depicts a 7-bit IFDPFA at $F = 3$, consisting of 3 central blocks of width 2 and of a right block of width 1. Its depth is 5, one node larger against the 7-bit IFDPA (Figure 14), but its size is 20, four nodes smaller. The IFDPA has the fan-out of 5.

By the assignment of various value to F, we can perform the exploration of adder space, thus obtaining appropriate values of the adder depth and size. Table 2 reports the depth and size of fan-out constrained adders of four bit-widths: 31, 63, 127 and 255. The weakening of fan-out constraint from 3 to 11 leads to the reduce of adder depth and to the growth of adder size.

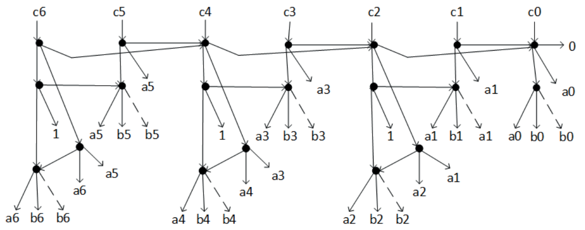


Fig. 17. Seven-bit IFDPFA; fan-is 3, depth is 5, and size is 20

VHDL modelling of if-diagram based adders

VHDL provides facilities for modeling adders at behavioral, dataflow and structural levels [20, 21]. Figure 18 depicts six types of cell concerning VHDL in the 15-bit IFDPA matrix *M*. As many as 29 cells correspond to XNOR-gate, 31 cells correspond to two-input multiplexer *MUX*, 17 cells correspond to OR-gate. We put into accordance a scalar signal $S_{c,r}$ to each non-empty cell, except the cells of row 0, which represent signals of sum *S*.

Figure 19 shows an example VHDL-model of a 3-bit IFDPA, which consists of two modules: entity *ADDER*_3 and architecture *STRUCTURE*_3. The first module describes the adder input and output ports, while the second one describes the adder at the structure-dataflow level. All the signal types are from the package *std_logic_1164* of IEEE library. The architecture models the adder structure by component instantiation and signal assignment statements, and by logic operators of VHDL. The adder architecture uses a component of two-input multiplexer *MUX*.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ADDER_3 is port(
    A: in std_logic_vector(2 downto 0);
    B: in std_logic_vector(2 downto 0);
    S: out std_logic_vector(3 downto 0));
end entity ADDER_3;
architecture STRUCTURE_3 of ADDER_3 is
    component MUX is port(
        SEL: in std_logic;
        D0: in std_logic;
        D1: in std_logic;
        RES: out std_logic);
    end component MUX;
    signal S0_1, S0_2: std_logic;
    signal S1_1, S1_2: std_logic;
    signal S2_1, S2_2, S2_3, S2_4: std_logic;
begin
    S(0) <= not S0_2;
    S0_1 <= S0_2 and B(0);
    S0_2 <= B(0) xnor A(0);
    S(1) <= S0_1 xnor S1_2;
    C10: MUX port map (S1_2, B(1), S0_1, S1_1);
    S1_2 <= B(1) xnor A(1);
    S(2) <= S1_1 xnor S2_4;
    C20: MUX port map (S2_2, S2_3, S0_1, S2_1);
    S2_2 <= S2_4 or S1_2;
    C21: MUX port map (S2_4, B(2), B(1), S2_3);
    S2_4 <= B(2) xnor A(2);
    S(3) <= S2_1;
end architecture STRUCTURE_3;
```

Fig. 19. VHDL model of 3-bit parallel adder

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	(1,1)	(8,1) (1,2) not(1,2)	(6,2) (1,3) not(1,3)	(6,3) (1,4) not(1,4)	(4,4) (1,5) not(1,5)	(6,5) (1,6) not(1,6)	(4,6) (1,7) not(1,7)	(4,7) (1,8) not(1,8)	(2,8) (1,9) not(1,9)	(6,9) (1,10) not(1,10)	(4,10) (1,11) not(1,11)	(4,11) (1,12) not(1,12)	(2,12) (1,13) not(1,13)	(4,13) (1,14) not(1,14)	(2,14) (1,15) not(1,15)	(2,15) 0 1
1		(2,1) (3,1) (1,9)	(2,2) (3,2) (1,9)	(2,3) (3,3) (1,9)	(2,4) (3,4) (1,9)	(2,5) (3,5) (1,9)	(2,6) (3,6) (1,9)	(2,7) (3,7) (1,9)	(2,8) b7 (1,9)	(2,9) (3,9) (1,13)	(2,10) (3,10) (1,13)	(2,11) (3,11) (1,13)	(2,12) b3 (1,13)	(2,13) (3,13) (1,15)	(2,14) b1 (1,15)	
2		(4,1) 1 (2,5)	(4,2) 1 (2,5)	(4,3) 1 (2,5)	(4,4) 1 (2,5)	(4,5) 1 (2,7)	(4,6) 1 (2,7)	(4,7) 1 (2,8)	b7 a7 not a7	(4,9) 1 (2,11)	(4,10) 1 (2,11)	(4,11) 1 (2,12)	b3 a3 not a3	(4,13) 1 (2,14)	b1 a1 not a1	b0 a0 not a0
3		(4,1) (5,1) (3,5)	(4,2) (5,2) (3,5)	(4,3) (5,3) (3,5)	(4,4) b11 (3,5)	(4,5) (5,5) (3,7)	(4,6) b9 (3,7)	(4,7) b8 b7		(4,9) (5,9) (3,11)	(4,10) b5 (3,11)	(4,11) b4 b3		(4,13) b2 b1		
4		(6,1) 1 (4,3)	(6,2) 1 (4,3)	(6,3) 1 (4,4)	b11 a11 not a11	(6,5) 1 (4,6)	b9 a9 not a9	b8 a8 not a8		(6,9) 1 (4,10)	b5 a5 not a5	b4 a4 not a4		b2 a2 not a2		
5		(6,1) (7,1) (5,3)	(6,2) b13 (5,3)	(6,3) b12 b11		(6,5) b10 b9				(6,9) b6 b5						
6		(8,1) 1 (6,2)	b13 a13 not a13	b12 a12 not a12		b10 a10 not a10				b6 a6 not a6						
7		(8,1) b14 b13														
8		b14 a14 not a14														

is XNOR   is OR   is AND   is NOT   is MUX   is reference

Fig. 18. VHDL model primitives of 15-bit IFDPA

Results

Comparison of IFDPA against Brent-Kung and majority-invertor-gate adders. The 8-bit IFDPA (it is based on 7-bit adder depicted in Figure 14) has the depth of 5, fan-out of 4 and size of 32 MUXs that represent 16 XNOR, 4 OR and 12 true MUX gates. The adder consists of 4 blocks which width is 1, 2, 2 and 3 bit. The 3-bit block is a right-hand slice of a 4-bit block. The 8-bit BKA (Figure 5) has the depth of 6, fan-out of 4 and size of 16 XOR, 11 OR and 30 AND gates. The number of OR / AND gates in BKA is 41, while the number of OR / AND gates in IFDPA is $4 + 12 \times 3 = 40$. Therefore, IFDPA is preferable to BKA with respect to the depth and size. Moreover, IFDPA is a multi-root decision diagram, and many advanced diagram-based design algorithms are applicable to it.

Table 3 provides a comparison of IFDPAs to MIGAs. Everyone can see that the gain of IFDPAs is significant against MIGAs [6] regarding both the size and depth.

FPGA-based synthesis of adders. We have synthesized the IFDRCA and IFDPA VHDL models of bit-width 15, 31, 63 and 127 for FPGA (device EP4CE115F29I8L Cyclone IV E) using the software Quartus Prime Version 18.0.0 Build 614 04/24/2018 SJ Lite Edition, copyright Intel Corporation [22]. Two optimization modes (aggressive performance-speed and aggressive area) have produced networks with distinct parameters: time delay and number of logic elements. Tables 4 and 5 report obtained results.

The gain of IFDPA over IFDRCA with respect to the time delay increases rapidly with the growth of bit-width from 15 to 127. At the same time, IFDRCA consumes slightly less area against IFDPA.

Table 3. Comparison of IFDPA against MIG adders

Width, bit	IFDPA		Width, bit	Optimized MIG	
	Size	Depth		Size	Depth
31	191	6	32	610	12
63	447	7	64	1159	11
127	1023	8	128	14672	19
255	2303	9	256	7650	16

Table 4. **Time delay (ns) of FPGA implementation of IFDRCA and IFDPA vs. adder bit-width**

Adder	Adder width, bit			
	15	31	63	127
IFDRCA, speed	21.3	40.6	73.9	137.3
IFDRCA, area	23.7	41.6	77.4	172.8
IFDPA, speed	17.5	16.7	26.7	37.3
IFDPA, area	19.0	22.6	27.5	40.5

Table 5. **Logic elements (area) in FPGA implementation of IFDRCA and IFDPA vs. adder bit-width**

Adder	Adder width, bit			
	15	31	63	127
IFDRCA, speed	31	74	172	327
IFDRCA, area	29	61	125	253
IFDPA, speed	37	81	172	359
IFDPA, area	36	77	157	321

Conclusion

We have analyzed known implementation models of many-bit adders and have developed a technique of transforming (by cutting the longest paths) a ripple-carry adder to a parallel adder represented by if-decision diagrams. The new parallel adder has a blocked structure, the performance of $O(\log_2 n)$ and the area size of $O(n \times \log_2 n)$. We have proposed the table representation of decision diagram-based adder and a method of mapping the diagram to a VHDL-model. The FPGA-based synthesis results and the comparisons of the if-diagram-based adders to the Brent-Kung and majority-invertor gate adders show that the new adders lead to faster and smaller circuits.

REFERENCES

1. **T.-K. Liu, K.R. Hohulin, L.-E. Shiau, S. Muroga.** «Optimal One-Bit Full-Adders with Different Types of Gates». IEEE Transactions on Computers. Bell Laboratories: IEEE, 1974, C-23 (1): 63–70.
2. **Rosenberger, G.B.** «Simultaneous Carry Adder». U.S. Patent 2,966,305. (1960–12–27).
3. **P.M. Kogge, H.S. Stone.** «A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations». IEEE Transactions on Computers. 1973, C-22 (8): 786–793.
4. **R.P. Brent, H. Te Kung,** «A Regular Layout for Parallel Adders». IEEE Transactions on Computers. 1982, C-31, (3): 260–264.
5. **N. Poornima, V.S. Kanchana Bhaaskaran.** «Area Efficient Hybrid Parallel Prefix Adders». Procedia Materials Science 10 (2015), pp. 371–380.

6. **L. Amarú, P.-E. Gaillardon, G. De Micheli,** «Majority-Inverter Graph: A New Paradigm for Logic Optimization» IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 5, pp. 806–819, May 2016.
7. **L. Amarú, P.-E. Gaillardon, A. Chattopadhyay, G. De Micheli,** «A Sound and Complete Axiomatization of Majority-n Logic» IEEE Transactions on Computers, vol. 65, no. 9, pp. 2889–2895, September 2016.
8. **V. Tenace, A. Calimera, E. Macii, M. Poncino.** «Pass-XNOR logic: A new logic style for P-N junction based graphene circuits». DATE, 2014, pp.1–4.
9. **Prihozhy, A.A.** If-Diagrams: Theory and Application / A.A. Prihozhy // Proc. 7th Int. Workshop PATMOS'97.– UCL, Belgium, 1997.– P. 369–378.
10. **Prihozhy, A.A.** Parallel Computing with If-Decision-Diagrams / A.A. Prihozhy, P.U. Brancevich // Proc. Int. Conference PARELEC'98.– Poland, Technical University of Bialystok.– 1998.– P. 179–184.
11. **Prihozhy A.A.** Incompletely Specified Logical Systems and Algorithms / A.A. Prihozhy / Minsk, Technical Literature.– 2013.– 343 c.
12. **Prihozhy A.A.** «Generalization of the Shannon Expansion for Incompletely Specified Functions: Theory and Application». System analysis and applied information science». 2013; (1–2): 6–11.
13. **C.Y. Lee,** Representation of Switching Circuits by Binary-Decision Programs, Bell Systems Technical Journal, 1959, Vol. 38, No 4, pp. 985–999.
14. **L. Amarú, P.-E. Gaillardon, G. De Micheli.** «Biconditional BDD: a novel canonical BDD for logic synthesis targeting XOR-rich circuits» in DATE'13, 2013, pp. 1014–1017.
15. **A. Bernasconi et al.,** «On decomposing Boolean functions via extended cofactoring» in DATE, 2009, pp. 1464–1469.
16. **V. Tenace, A. Calimera, E. Macii, M. Poncino.** One-pass logic synthesis for graphene-based Pass-XNOR logic circuits. DAC, 2015: 128:1–128:6.
17. **Prihozhy, A.A.** If-Decision Diagram Based Synthesis of Digital Circuits / A.A. P rihozhy // Proc. Int. Conf. «Information Technologies for Education, Science and Business».– Minsk, Belarus.– 1999.– P. 65–69.
18. **Prihozhy, A.A.** If-Decision Diagram Based Modeling and Synthesis of Incompletely Specified Digital Systems / A.A. Prihozhy, B. Becker // Electronics and communications, Electronics Design.– Kyiv.– 2005, pp. 103–108.
19. **Prihozhy, A.A.** Analysis, transformation and optimization for high performance parallel computing / A.A. Prihozhy // Minsk, BNTU.– 2019.– 229 p.
20. IEEE Standard VHDL Language Reference Manual. The Institute of Electrical and Electronics Engineers, Inc.– 2000.– 299 p.
21. **Prihozhy, A.A.** High-Level Synthesis through Transforming VHDL Models / A.A. Prihozhy // Chapter in Book «System-on-Chip Methodologies and Design Languages».– Kluwer Academic Publishers.– 2001.– P. 135–146.
22. Quartus Prime Lite Edition [Electronic resource].– Access mode: https://fpgasoftware.intel.com/?edition=lite.– Date of access: 24.04.2020.

ЛИТЕРАТУРА

1. **T.-K. Liu, K.R. Hohulin, L.-E. Shiau, S. Muroga.** «Optimal One-Bit Full-Adders with Different Types of Gates». IEEE Transactions on Computers. Bell Laboratories: IEEE, 1974, C-23 (1): 63–70.
2. **Rosenberger, G.B.** «Simultaneous Carry Adder». U.S. Patent 2,966,305. (1960–12–27).
3. **P.M. Kogge, H.S. Stone.** «A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations». IEEE Transactions on Computers. 1973, C-22 (8): 786–793.
4. **R.P. Brent, H. Te Kung,** «A Regular Layout for Parallel Adders». IEEE Transactions on Computers. 1982, C-31, (3): 260–264.
5. **N. Poornima, V.S. Kanchana Bhaaskaran.** «Area Efficient Hybrid Parallel Prefix Adders». Procedia Materials Science 10 (2015), pp. 371–380.
6. **L. Amarú, P.-E. Gaillardon, G. De Micheli,** «Majority-Inverter Graph: A New Paradigm for Logic Optimization,» IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 5, pp. 806–819, May 2016.
7. **L. Amarú, P.-E. Gaillardon, A. Chattopadhyay, G. De Micheli,** «A Sound and Complete Axiomatization of Majority-n Logic,» IEEE Transactions on Computers, vol. 65, no. 9, pp. 2889–2895, September 2016.
8. **V. Tenace, A. Calimera, E. Macii, M. Poncino.** «Pass-XNOR logic: A new logic style for P-N junction based graphene circuits». DATE, 2014, pp.1–4.
9. **Prihozhy, A.A.** If-Diagrams: Theory and Application / A.A. Prihozhy // Proc. 7th Int. Workshop PATMOS'97.– UCL, Belgium, 1997.– P. 369–378.
10. **Prihozhy, A.A.** Parallel Computing with If-Decision-Diagrams / A.A. Prihozhy, P.U. Brancevich // Proc. Int. Conference PARELEC'98.– Poland, Technical University of Bialystok.– 1998.– P. 179–184.
11. **Прихожий А.А.** Частично определенные логические системы и алгоритмы / А.А. Прихожий / Минск, БНТУ.– 2013.– 343 с.
12. **Прихожий, А.А.** Обобщение разложения Шеннона для частично определенных функций: теория и применение / А.А. Прихожий / Системный анализ и прикладная информатика.– 2013, № 1–2.– С. 6–11.
13. **C.Y. Lee,** Representation of Switching Circuits by Binary-Decision Programs, Bell Systems Technical Journal, 1959, Vol. 38, No 4, pp. 985–999.
14. **L. Amarú, P.-E. Gaillardon, G. De Micheli.** «Biconditional BDD: a novel canonical BDD for logic synthesis targeting XOR-rich circuits,» in DATE'13, 2013, pp. 1014–1017.
15. **A. Bernasconi et al.,** «On decomposing Boolean functions via extended cofactoring,» in DATE, 2009, pp. 1464–1469.

16. **V. Tenace, A. Calimera, E. Macii, M. Poncino.** One-pass logic synthesis for graphene-based Pass-XNOR logic circuits. DAC, 2015: 128:1–128:6.
17. **Prihozhy, A.A.** If-Decision Diagram Based Synthesis of Digital Circuits / A.A. Prihozhy // Proc. Int. Conf. «Information Technologies for Education, Science and Business».– Minsk, Belarus.– 1999.– P. 65–69.
18. **Prihozhy, A.A.** If-Decision Diagram Based Modeling and Synthesis of Incompletely Specified Digital Systems / A.A. Prihozhy, B. Becker // Electronics and communications, Electronics Design.– Kyiv.– 2005, pp. 103–108.
19. **Prihozhy, A.A.** Analysis, transformation and optimization for high performance parallel computing / A.A. Prihozhy // Minsk, BNTU.– 2019.– 229 p.
20. IEEE Standard VHDL Language Reference Manual. The Institute of Electrical and Electronics Engineers, Inc.– 2000.– 299 p.
21. **Prihozhy, A.A.** High-Level Synthesis through Transforming VHDL Models / A.A. Prihozhy // Chapter in Book «System-on-Chip Methodologies and Design Languages».– Kluwer Academic Publishers.– 2001.– P. 135–146.
22. Quartus Prime Lite Edition [Electronic resource].– Access mode: https://fpgasoftware.intel.com/?edition=lite.– Date of access: 24.04.2020.

Поступила	После доработки	Принята к печати
11.04.2020	*01.05.2020*	*01.06.2020*

ПРИХОЖИЙ А.А.

СИНТЕЗ ПАРАЛЛЕЛЬНЫХ СУММАТОРОВ ПО IF-ДИАГРАММАМ РЕШЕНИЙ

Сложение является одной из критичных ко времени операций в большинстве современных процессоров. В течение десятилетий проводились обширные исследования, посвященные проектированию высокоскоростных и менее сложных архитектур сумматоров, а также разработке передовых технологий реализации сумматоров. Диаграммы решений являются перспективным подходом к эффективному проектированию многоразрядных сумматоров. Поскольку традиционные двоичные диаграммы решений не полностью соответствуют задаче моделирования архитектур сумматоров, были предложены другие типы диаграмм. If-диаграммы решений являются параллельной моделью многоразрядного сумматора с временной сложностью $O(log_2 n)$ и технической сложностью $O(n \times log_2 n)$. Настоящая статья предлагает метод систематического разрезания длинных путей в графе диаграммы, который порождает модели сумматоров с такими характер истиками, Сумматоры на базе if-диаграмм конкурентоспособны по сравнению с сумматором Брент-Кунга и его многочисленными модификациями. Мы предлагаем блочную структуру параллельных сумматоров, построенных на if-диаграммах, и вводим их табличное представление, которое способно систематически создавать модели на основе диаграмм любой битовой ширины. Табличное представление сумматоров поддерживает эффективное отображение диаграмм в VHDL-модули на структурном и потоковом уровнях. В статье также исследовано пространство сумматоров посредством изменения коэффициента разветвления выходов. Результаты синтеза на основе ПЛИС и сравнения конкретных сумматоров, построенных на if-диаграммах, с сумматорами Брента-Кунга и мажоритарно-инверторными сумматорами показывают, что новые сумматоры дают более быстрые цифровые схемы меньшего размера.

Ключевые слова: *много-битовые сумматоры, диаграммы решений, временная задержка, площадь, VHDL, FPGA, пространство распараллеливания*

Anatoly Prihozhy is a full professor at the Computer and system software department of Belarusian national technical university, doctor of science (1999) and full professor (2001). His research interests include programming and hardware description languages, parallelizing compilers, and computer aided design techniques and tools for software and hardware at logic, high and system levels, and for incompletely specified logical systems. He has over 300 publications in Eastern and Western Europe, USA and Canada. Such worldwide publishers as IEEE, Springer, Kluwer Academic Publishers, World Scientific and others have published his works.