Genetic Programming of Discrete Cosine Transform Processors

Anatoliy Sergiyenko
Dept. of Computer Engineering
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
aser@comsys.kpi.ua

Anastasia Serhienko
Dept. of System Programming and Specialized Computer Systems
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
an.ser.313kpi@gmail.com

Vitaliy Romankevich
Dept. of System Programming and Specialized Computer Systems
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
romankev@scs.ntu-kpi.kiev.ua

***Abstract*—A method of the pipelined datapath synthesis which is based on the genetic programming is used for the discrete cosine transform (DCT) processor synthesis. The method is based on the representation of the algorithm by the spaced synchronous data flow graph (SDF), transferring it to a chromosome, and using the genetic optimization process. The method is implemented in the SDFCAD program system which is able to perform the input of algorithms specified by the SDF and automatically perform the synthesis of pipelined processors. The DCT processors synthesized by this system have less hardware volume and higher throughput than the similar processors configured in FPGA.**

Keywords—SDF, FPGA, VHDL, DCT, datapath, genetic programming

I. Introduction

The system-level design is a new technology to speed-up the ASIC and FPGA design. But the system synthesis algorithms are much complex comparing to the logical or physical design [1]. The behavioral synthesis as a case of the system synthesis is based on mapping the behavioral algorithm to the device structure. This mapping is based in the scheduling an algorithm represented by a data flow graph (DFG) to the selected set of computational resources. During the behavioral synthesis, such greedy algorithms like force-directed scheduling [2] give the frequent results but which are far from the optimum ones. The use of the integer linear programming approach enables the optimum solutions. But this method is not applicable to large DFGs [3].

The use of the genetic programming (GP) algorithms in the behavioral synthesis makes it possible to synthesize the processors of any complexity [2,4]. This approach is often used for the program development but sometimes for the hardware synthesis[5].

A method spatial synchronous dataflow graph (SDF) mapping into the structure configured in FPGA is proposed in [6]. Here, its modernization to GP is considered and the results of its application for the discrete cosine transform (DCT) processor synthesis are shown.

II. Genetic Programming Methods

The genetic algorithm and its variation GP consider the search space $\mathbb{G}$ which is abstracted to a set of all possible chromosomes, that is, the strings $g \in \mathbb{G}$ that encode specific solutions. The chromosome g encodes a specific genotype. The individual $x \in \mathbb{X}$ is a candidate for the solution from the solution space $\mathbb{X}$. It is modeled as a phenotype $x = \text{Phenotype}(g)$. The level of the fitness to the optimum solution is determined by the fitness function $v(x)$, which moves the evolution process in the desired direction [4].

The optimization process consists in the evolution of several generations starting at initial random population Pop(0), and finishing with the population Pop(N) containing the optimum solution. The fitness value v is determined for the chromosomes g of each individual by calculating the efficiency criterion Fitness(g) and comparing it with other individuals of the set Pop(t), $t \le N$. Each new generation is the result of the function Select(*Pop*($t-1$)) which selects a set *Popm* of *ps* individuals that are suitable for the reproduction, and function Reproduce(*Popm*), which generates a new population using create, mutate, crossover operations.

GP most often deals with the optimization of DFG, which has a tree structure. This structure is coded as a bit vector which is used as the chromosome $g \in \mathbb{G}$ [7]. GP in various methods is performed only at the separate stages of the structure synthesis: resource allocation, scheduling, resource assignment. Thus, in the SPARCS method, the schedule is searched using linear programming, and the FPGA resource allocation is found by GP [8]. In [9], GP is used to improve the list scheduling, and in [10] does the cycle parallelization.

In several GP approaches the pair-based chromosome encoding is used, when an operator is related to some resource. The NSGA-II method uses the chromosome, in which the gene represents the operator-resource relationship. This method is based on the sequential deriving the structure, schedule, and operator assignment based on DFG [11].

In the PDGP method, the tree-like DFG is represented in the n-dimensional lattice. The evolution is subject to the operation nodes and the links between them taking into account the restrictions [12].

The ECGP method is used to find the parallel algorithms that are running on a system with limited hardware. This method is called as Cartesian GP method because the graph nodes are represented in a two-dimensional lattice with the Cartesian coordinates. The execution of an operator node that stands to the left precedes the execution of one that stands to the right. The chromosome is a series of integers that encode the nodes of the lattice. Numeric vertex tags, i.e., genes, encode links between nodes and their functions. This method has many different modifications [13, 14].

The GP methods are adapted only to perform individual steps in the structural synthesis of computing devices, which are performed sequentially, so the optimization is often imperfect. In most methods, the result of the synthesis is a processor that does not perform the algorithm in the

pipelined mode, or the resulting datapath operates only with a period of one cycle.

Among all methods, the ECGP method is promising, in which the chromosomes encode the spatial position of the DFG operator nodes. Such encoding enables the effective finding of the fitness function.

III. Genetic Programming of the Pipelined Datapath

The method of the pipelined datapath design is based on the representation of SDF in 3-dimensional space as a spatial SDF $K_G = (K, D, A)$. Here, K is the matrix of node-vectors $\boldsymbol{K}_i$ corresponding to the operators; D is a matrix of edge-vectors $\boldsymbol{D}_j$ that represent links between operators; A is the incidence matrix of SDF. In the vector $\boldsymbol{K}_i = (k_i, s_i, t_i)^T$, the coordinates k_i, s_i, t_i correspond to the operator type, the processing unit (PU) coordinate, and the time slot, in which the result of this operator is written to the register [6, 15].

K_G consists of the spatial configuration $K_{GS} = (K_S, D_S, A)$, which codes the device structure, and the event configuration $K_{GT} = (K_T, D_T, A)$ which does the schedule. Then, the mapping of SDF into the structure consists of splitting K_G into K_{GS} and K_{GT} by cutting the matrices K and D into submatrices K_S, K_T, and D_S, D_T.

The matrix K of a spatial SDF encodes a feasible solution. The pipelined datapath synthesis using the spatial SDF consists of constructing a number of optimized solutions K_G and selecting the optimized solution due to an effectiveness criterion $v(K)$. The optimized solutions are obtained by equivalent transformations of the spatial SDF, for example, by rearranging the vectors $\boldsymbol{K}_i$ in space providing the conditions of the spatial SDF K_G correctness.

The spatial SDF method is implemented in GP. Then, the genotype is coded by the matrix K, and the vectors $\boldsymbol{K}_i$ serve the genes. For the traditional chromosome representation, the columns of the matrix K should be flattened in a single string g.

There are four basic reproducing operations in GP: a creation that creates a new genotype, a duplication that gives multiple copies of one individual, a mutation that is a random change in the genotype, and recombination (crossover) that combines two related genotypes into a new one.

When creating a genotype g of a spatial SDF, its gene parameters s_i, t_i are set at random so that the individual belongs to the solution space $g \in \mathbb{G}$. In this case, an individual g is viable if the following conditions are satisfied.

$$\forall\, \boldsymbol{K}_i, \boldsymbol{K}_j\, (\boldsymbol{K}_i \neq \boldsymbol{K}_j, i \neq j\,), \tag{1}$$

i.e, in the correct spatial SDF all the nodes are placed in the space separately.

The following conditions are satisfied by the living individual g:

$$\forall\, \boldsymbol{K}_i, \boldsymbol{K}_j (k_i = k_j,\, s_i = s_j\,) \Rightarrow t_i \not\equiv t_j \bmod L, \tag{2}$$

$$\forall \boldsymbol{D}_j \neq \boldsymbol{D}_{Dj}\, (t_i \geq 0), \tag{3}$$

$$\sum_j b_{i,j}\, \boldsymbol{D}_j = (0,0,0)^T, \tag{4}$$

ie, they prove the schedule correctness. Here, L is the iteration duration in clock cycles, $\boldsymbol{D}_{Dj} = (k_j, s_j, -wL)^T$ is the edge, which means the delay to w iterations, $b_{i,j}$ is the element of the i-th row of the cyclomatic matrix of SDF, $\boldsymbol{D}_j$ is the edge that belongs to some closed cycle of SDF. The condition (2) assures that the resulting structure correctly implements the circular schedule with the period of L clock cycles providing that the operations from different iterations can be overlapped. The conditions (3) and (4) provide the correct transfer of variables from the previous iteration to the present iteration of the algorithm. The conditions (1) – (4) must be satisfied in other reproducing operations as well.

The duplication is used in the case of an effective solution which properties should be used in the following generations:

$$g_n = \text{duplicate}(g),\, g \in \mathbb{G}.$$

The mutation is performed to create a new viable genotype by modifying an existing one. It happens either randomly or determined:

$$g_n = \text{mutate}(g),\, g \in \mathbb{G},\, g_n \in \mathbb{G}.$$

In the case of a gene (k_i, s_i, t_i), the parameters s_i, t_i are randomly changed so that the genotype g remains viable. Similarly, a mutation of a group of genes (allele) is performed.

The permutation is such a mutation when two genes of the same type k_i (k_i, s_p, t_p) and (k_i, s_q, t_q) are selected and their parameters are swapped giving the genes (k_i, s_q, t_q) and (k_i, s_p, t_p).

The recombination performs mapping $\mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}$. It can happen either accidentally or deterministically:

$$g_n = \text{recombine}(g_a, g_b) : g_a, g_b \in \mathbb{G} \Rightarrow g_n \in \mathbb{G}.$$

Two i-th genes (k_i, s_{ai}, t_{ai}), and (k_i, s_{bi}, t_{bi}) are selected for the recombination from the two randomly selected genotypes g_a, g_b. Then, the parameters of the first gene are replaced by the parameters of the second one, giving the gene (k_i, s_{bi}, t_{bi}) of the new genotype g_n. Similarly, the recombination of alleles is performed.

The reproducing operations are illustrated by Fig. 1. The spatial SDF subgraphs are considered in it. The exchanging of the nodes in the space is shown by the red arrows. Therefore, reproduction is the forming of a new population by performing operations of creation, duplication, mutation, and recombination. All four operations are performed arbitrarily. But in reproduction, the individual genes, that form an intron, are not subject to change because they do not affect the characteristics of the phenotype. Among them the genes of input-output nodes are.

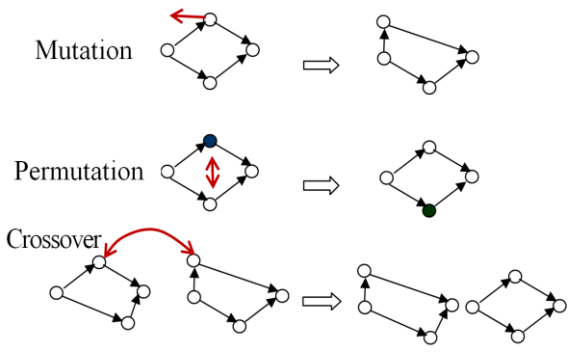

Fig. 1. Reproduction operations of the spatial SDF

The most complex task of the GP algorithm is the implementation of the function Fitness(g). When using spatial SDF, this function becomes much simple than in other GP methods. For example, the hardware complexity is estimated as the weighted number of PUs, which is calculated as the power of a set of nodes with equal coordinates k_i.

IV. Framework for Genetic Programming the Datapaths

A framework SDFCAD that implements the GP algorithm for the spatial SDF has been developed. The GP methods are typically differentiated due to different functions Reproduce(g), which selects the individuals for the new generation. Three GP methods are implemented in this framework. In the method QValue, the i-th individual is selected to a new generation with the probability

$$P_i = \frac{Q_{\max} - Q_i}{\sum_i (Q_{\max} - Q_i)},$$

where Q_i is the cost function of the i-th individual, $Q_{\max}$ is the maximum cost value, i.e., the worse fitness.

In the Roulette method, a ruler that simulates roulette is created. The line starts from zero and consists of numerical intervals, each of which corresponds to a specific individual. The width of the interval for a given individual g_i is equal to

$$\Delta_i = 1/\nu_i,$$

where ν_i is the criterion value for the individual g_i. That is, the larger the value of the criterion, the smaller the interval. The obtained intervals are plotted on a numerical line of a roulette wheel from zero (not inclusive). A random number is taken in the range from 0 to the end of the line. The random number falls in the interval which corresponds to the chosen individual. Then this interval is dropped from the line and the other parents are searched for in the same way.

The method of the Roulette with nishes, is distinguished in the following. A set K of niche coefficients is formed, in which each individual g_i is assigned a niche coefficient k_{Hi}. If the Roulette method without niches is used, then all niche coefficients are equal to $k_{Hi} = 1$.

The width of the interval for a particular individual in the simulates roulette is equal to

$$\Delta_i = k_{Hi} / \nu_i.$$

Thus, the niche ratio shows how much worse there are individuals in the niche compared to the best individual in the generation. Multiplying by k_{Hi} the width of the interval Δ_i increases the chances of survival of individuals in this niche in order to preserve the genetic diversity in the next generation. If for the particular SDF it turns out that the niches degrade quickly and their number decreases to zero, then it is necessary to increase the niche coefficients.

Fig.2 illustrates some generation that is divided into niches. It the generality function Sh(d, σ) depending on the vector criterion (f_1, f_2), where f_1 is the hardware cost, and f_2 is the critical path delay. A thick line shows the boundary to which the points of individuals are touched that are optimal for Pareto.

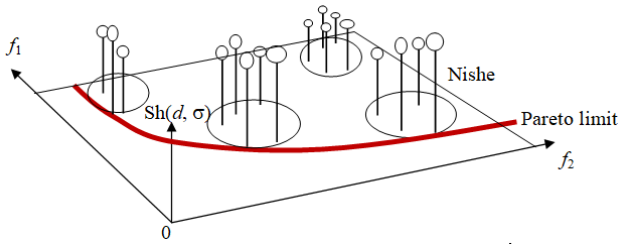

Fig. 2. A generation divided into niches

Ellipses denote niches, which are distinguished by the generality function Sh(d, σ), which shows the extent to which two individuals g_1, g_2 share the same place in $\mathbb{X}$, where σ is a constant that specifies the radius of a section that defines a cluster of related persons, the so-called niche.

The generality function Sh(d, σ) is used to distribute the individuals on a number of hills in the landscape of the Fitness(g) function, and the hill receives a share of the population that is proportional to its height. Therefore, the Fitness(g) function should select the best individuals, provided that the diversity of individuals is preserved so that the convergence of the algorithm does not lead to a local optimum. At the same time, the generality function makes it possible to identify niches and concentrate outstanding individuals from several niches in the archive that stores the elite individuals. In addition, if the niche gathers a large number of individuals with approximately the same efficiency, it indicates a convergence to the local optimum. To reduce it, it is necessary to throw out such individuals from the generation at the stage of selection.

The optimization process takes two stages, each of them implements the GP algorithm. At the first stage, the initial spatial SDF is optimized. At the second stage, both the register number and the multiplexer input number are minimized according to the approach described in [6].

V. DCT Processor Synthesis

The development of the application-specific DCT processors is a typical test for the high-level synthesis systems. The structures of the processors consist of an input buffer memory, two pipelines that perform DCT of size 8, and a buffer memory between them to transpose the matrix of the intermediate results. The common control unit is designed to generate the necessary address sequences and control signals.

Fast algorithms for calculating one-dimensional DCT in most cases are variants of Chen's algorithm [16]. These algorithms are characterized by a minimum number of operations (the record is 11 multiplication operations). But due to the inhomogeneity of the algorithm graph, it makes it difficult to build a pipelined DCT datapath.

As a test algorithm for DCT, a 8-point DCT is taken in many works [17]. The spatial SDF of this algorithm is put in the SDFCAD system as is shown in Fig. 3. The horizontal axis in it represents the clock time, and the number of the processing unit where the operator is done is represented by the vertical axis. The algorithm contains 13 multiplication operations and 29 additions. During the period of L = 8 cycles, it consumes 8 data and outputs 8 results through one input port and the output port in the natural order.

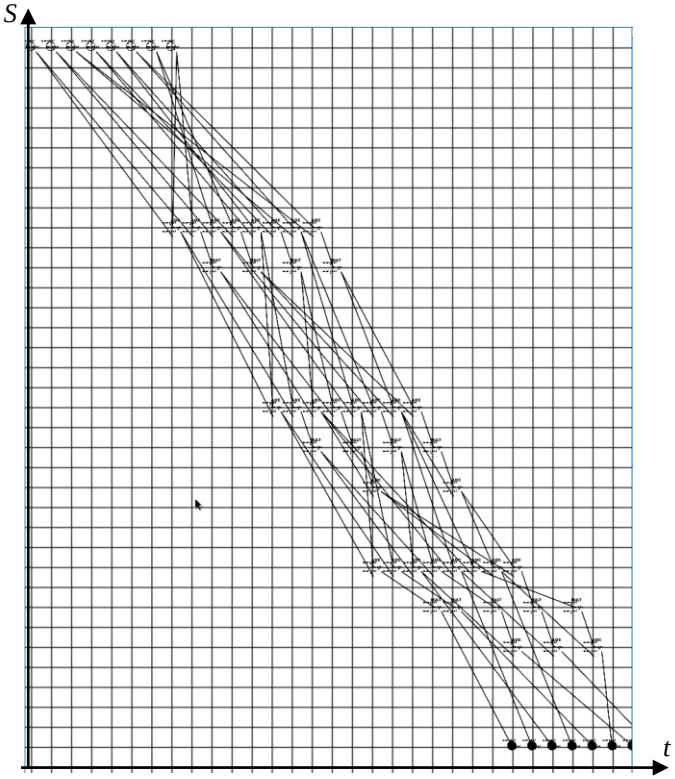

Fig. 3. Spatial SDF of the 8-point DCT

The synthesis of the DCT processor was performed according to the methods described above using the means of the automatic synthesis program of SDFCAD. Initially, the first generation of individuals was generated due to a random gene assignment. The number of generations is limited by 300. Although the chromosomes have random coordinates of the node vectors, the resulting structures are correct according to the relations (1) – (4), that is, each individual is mapped to a correct but suboptimal structure and schedule.

Then the GP algorithm executed. To analyze the work of the evolutionary approach, the algorithm was performed with three types of selection functions: Qvalue, Roulette, and Roulette with niches (Roulette+N).

The GP evaluation is usually illustrated by a chart of the fitness function *v* on the generation number. Fig. 4 shows this function at the first stage of synthesis for three selection functions.

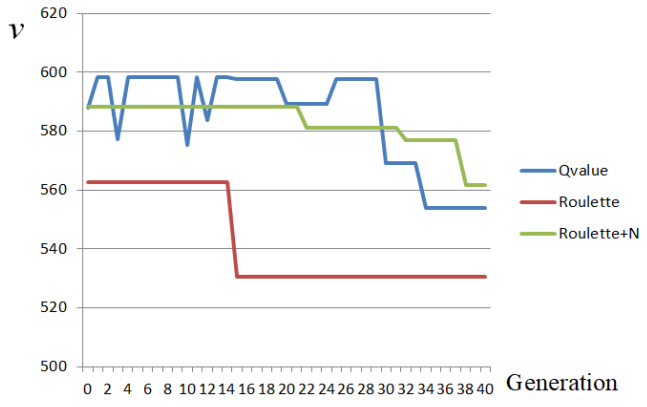

Fig. 4. Fitness function at the first synthesis stage

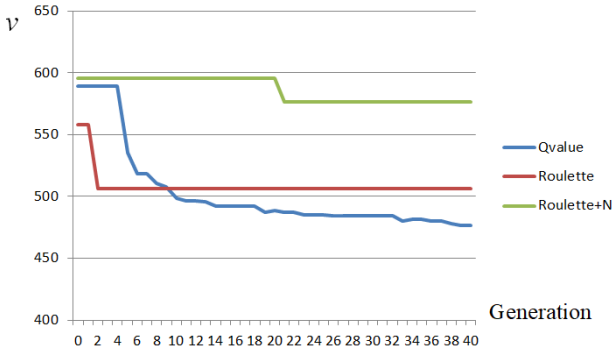

Fig. 5. Fitness function at the second synthesis stage

Fig. 5 shows the same function at the second stage.

The analysis of the charts in Fig.4, 5 shows that the algorithms at these two stages converge to a stable local minimum after 40 generations. Table 1 shows the hardware costs before (B) (the best individual in the first generation) and after (A) the synthesis of the DCT processor by these three methods, as well as the processor, whose SDF is optimized manually.

TABLE I. HARDWARE COSTS BEFORE AND AFTER SDF OPTIMIZATION BY DIFFERENT METHODS

Method	**Multi-pliers**		**Adders**		**Registers**		***v***	
	B	**A**	**B**	**A**	**B**	**A**	**B**	**A**
Qvalue	7	3	18	9	96	28	584	480
Roulette	7	3	19	6	107	63	608	546
Roulette+N	6	3	19	11	101	38	562	506
By hand	—	3	—	6	—	58	—	275

Fig. 4, 5 as well as Table 1 analysis shows that the GP method is able to automatically optimize the pipelined datapath with an improvement of the criterion by 11% – 22%. The quality level of the synthesis results significantly depends on the random definition of the initial generation (deviation range is 10%). The solution quality found automatically can be 80% worse than the solution found manually by the method of spatial SDF. All proved methods reduce the number of multiplication units, registers, adders to the minimum values at the cost of increasing the number of the multiplexor inputs. For example, the number of multiplexer inputs is in twofold higher than in the processor which is optimized by hand. The best results were achieved by the Roulette method.

The hardware parameters of the synthesized DCT processors are shown in Table 2 where they are compared to the characteristics of 1D DCT processors, which are obtained by other methods of synthesis. Therefore, the DCT processor, which is synthesized automatically with the method of GP in its main parameters is not worse than the known examples of such processors designed by other methods.

TABLE II. HARDWARE COSTS OF SYNTHESIZED DCT PROCESSORS

Method	Adders	Multipliers	Registers
SDF folding [16]	6	3	29
Retiming and scheduling [17]	7	3	34
Proposed, Roulette	6	3	63
Proposed, QValue	9	3	28

By itself, a 1D DCT processor has no practical significance. But it is part of the 2D DCT processor, which is the basis of the image compression processors. There are many samples of such processors in the literature and Table 3 compares many of them. For comparison, a 16-bit 2D DCT processor is shown which is developed on the base of the synthesized 1D DCT processor, which is discussed above.

TABLE III. HARDWARE COSTS OF DEVELOPED 2D DCT PROCESSORS

Processor	FPGA serie	Bitwidth	LUTs	Triggers	DSP48 units	RAM blocks	Maximum clock frequency, MHz
Synthesized	Virtex-2	16	1575	1747	6	0	155
Hannig [18]	Virtex-2	16	1754	1152	8	1	130
Xilinx [19]	Virtex-2	16	6832	7964	0	2	96
ALDCT [20]	Virtex-2	8	599	—	6	0	230
Synthesized	Spartan-3e	16	1531	1779	6	0	112
Kusuma [21]	Spartan-3e	8	1750	1696	11	1	85
Pradeepthi [22]	Spartan-3e	8	1239	1551	8	1	101
Synthesized	Virtex-5	16	1041	1778	6	0	173
DCTAAN [23]	Virtex-5	8	830	602	4	0	325
Pradnya [24]	Virtex-5	8	1152	578	4	1	—

It is worth comparing the resulting processor with the Hannig processor, which was synthesized using the PARO framework [18]. This framework involves parallelizing the algorithm represented by the loop nest using the loop unfolding and loop folding techniques and solving the scheduling problem by the classical method. Therefore, with the same bit rate, the new processor has three fewer multiplication units, 11% less hardware in look-up tables (LUTs), and 19% more clock speed by increasing the number of triggers by 1.5 times. This indicates the advantages of the GP method of spatial SDF over other methods of synthesis. Note that usually, FPGAs have the triggers with the excess providing the high pipelining level.

Also, a comparison of the synthesized processor with samples of manually optimized processors shows that this processor is not worse than them, but in some respects, such as maximum clock speed it is even better, providing the increased bit rate. This is due to the fact that the synthesized processor has a high level of pipelining, which is confirmed by the excessive number of triggers used.

In addition, the experiments with the settings of the compiler-synthesizer proved that the maximum clock period is not improved when setting the mode of synthesis with the retiming.

Thus, the DCT processor design has shown high efficiency of the GP method for spatial SDF. These processors can be used in all new photo and video image processing devices, including those that require high-bandwidth processing.

VI. CONCLUSIONS

A new method of GP for programming of spatial SDFs has been developed, which differs in that SDF is optimized by the method of genetic programming. In this method, the representations of the chromosome, functions of initialization of individuals, their suitability, selection, and reproduction, as well as a two-stage optimization algorithm are proposed.

The framework SDFCAD is designed to enter the spatial SDF and its optimization by the proposed method of genetic programming. The reproduction functions QValue, Roulette, and Roulette with nishes are built in this framework. The application allows the user to set such parameters as the suitability function, the reproduction function with its probability of mutation, the share of the elite set, the size of the population.

A set of DCT processors has developed automatically using the method of the genetic programming of spatial SDF built in the framework which has utilized three reproduction functions. The developed processors have a high ratio of performance – hardware costs and the natural order of input data and results. It found out that the function Roulette gives the best results for the DCT processor synthesis. The comparison of the automatically developed DCT processors with samples of manually optimized processors and processors designed by the known methods shows that this processor is not worse than them, and have the maximum clock speed.

The developed GP method needs improvement and implementation in the automatic high-level synthesis tools.

REFERENCES

[1] S. Bhattacharyya, M. Wolf, “Tools and Methodologies for System-Level Design”, in *Electronic Design Automation for IC System Design, Veriication, and Testing*, CRC Press. 2016, pp. 39–57.

[2] G. Wang, W. Gong, and R. Kastner, “Operation Scheduling: Algorithms and Applications”, in *High-level synthesis: from algorithm to digital circuit*, Philippe Coussy, Adam Morawiec Ed-s. Springer, 2008, pp. 231–255.

[3] C.T. Hwang, T.H. Lee, and Y.C. Hsu, “A formal approach to the scheduling problem in high level synthesis”, *IEEE Trans. Comput. Aided Des.*, no. 10, 1991, pp. 464–475.

[4] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, M. Steinbrecher, “Computational Intelligence. A Methodological Introduction”, *Springer*, 2-nd Ed. 2016.

[5] J. F. Miller, “Cartesian Genetic Programming”, *Springer*, Berlin. 2011.

[6] A. M. Sergiyenko, V. P. Simonenko, “Method of synchronous data-flow scheduling”, *System Research and Information Technologies*, no. 1, pp. 51–62, 2016. DOI: 10.20535/SRIT.2308-8893.2016.1.06.

[7] M. Affenzeller, S. Winkler, S. Wagner, A. Beham, “Genetic Algorithms and Genetic Programming. Modern Concepts and Practical Applications”, *Chapman & Hall* / CRC, 358 p., 2009.

[8] I. Ouaiss, S. Govindarajan, V. Srinivasan, M. Kaul, R. Vemuri, “An Integrated Partitioning and Synthesis System for Dynamically Reconfigurable Multi-FPGA Architectures”, in: *Proc. of the Reconfigurable Architectures Workshop (RAW’98)*, Springer, Berlin/Heidelberg, vol. 1388, pp. 31–36, 1998.

[9] M.Grajcar, “Conditional Scheduling for Embedded Systems using Genetic List Scheduling”, in *Proc. 13th International Symposium on System Synthesis (ISSS)*, Madrid, Spain, pp. 123–128, 2000.

[10] T. Weise, “Global Optimization Algorithms. Theory and Application”, *Version:* 2009-06-26. 820 p. Available at http://www.it-weise.de/

[11] R. Poli, “Evolution of Graph-like Programs with Parallel Distributed Genetic Programming”, in *Genetic Algorithms: Proceedings of the Seventh International Conference*, 1997, pp. 346–353.

[12] J. F. Miller, J. A. Walker, “Embedded cartesian genetic programming and the lawnmower and hierarchical-if-and-only-if problems”, in *Genetic and Evolutionary Computation Conference, GECCO06*, Seattle, Washington, USA July, 2006, pp. 911–918.

[13] J. Husa, R. Kalkreuth, “A Comparative Study on Crossover in Cartesian Genetic Programming”, in *Proceedings 21-st European Conference “Genetic Programming”, EuroGP*, Parma, Italy, April 4–6, 2018. LNCS, vol. 10781, 2018, pp. 203–219.

[14] O. Koncal, L. Sekanina, “Cartesian Genetic Programming as an Optimizer of Programs Evolved with Geometric Semantic Genetic Programming”, in *Proceedings 22-nd European Conference, EuroGP Genetic Programming*, Leipzig, Germany, April 24–26, 2019, pp.98–113.

[15] A. Sergiyenko, A. Serhienko, A. Simonenko, "A method for synchronous dataflow retiming," *IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, Kiev, 2017, pp. 1015-1018.

[16] J. Nikara, J. Takala, D. Akopian, J. Saarinen, “Pipeline Architecture for DCT/IDCT”. *IEEE Int. Symp. on Circuits and Systems*, (ISCAS 2001), May 6-9, Sydney, Australia, 2001. P. 902–905.

[17] S.-F. Hsiao, W.-R. Shiue, J.-M.Tseng, “A cost efficient fully-pipelinable architecture for DCT/IDCT”. *IEEE Trans. On Communications*, 1991. V. 39. No. 5. P. 640–643.

[18] F.Hannig, “Scheduling Techniques for High-Throughput Loop Accelerators”. *Erlangen*, 2009. 295 P.

[19] 2-D Discrete Cosine Transform (DCT). V2.0. Xilinx Inc Product Specification. 2002. March 14. 11 p. URL: http://www.xilinx.com.

[20] A. M. Sergiyenko, V. L. Lepekha, T. M. Lesyk, “Specprocessory dla dvovymirnogo dyskretnogo kosynusnogo peretworennya”, *Vistnyk NTUU “KPI”, Ser.: Informatyka i obchysluvalna technika.* V. 47. 2007. P. 230–233. (In Ukrainian). Сергієнко А. М., Лепеха В. Л., Лесик Т. М. Спецпроцесори для двовимірного дискретного косинусного перетворення. *Вісник НТУУ «КПІ», Сер.: Інформатика і обчислювальна техніка*: зб. наук. праць. Т. 47. 2007. С. 230–233.

[21] Kusuma E. D., Widodo T. S. FPGA implementation of pipelined 2D-DCT and quantization architecture for JPEG image compression. *2010 International Symposium on Information Technology*, Kuala Lumpur, 2010, pp. 1-6.

[22] Pradeepthi T., Ramesh A. P. Pipelined architecture of 2D-DCT, quantization and zigzag process for JPEG image compression using VHDL. *International Journal of VLSI Design & Communication Systems*, V. 2, 2011. No. 3. P. 99-110.

[23] Uzenkov O., Sergiyenko A. Pipelined DCT/IDCT. Created 01.02.2010. Available at: https://opencores.org/projects/dct_idct

[24] Pradnya P. Parate1, Nilesh A. Mohota, “FPGA Implementation of 2D-DCT for Image Compression”. *International Journal of Science and Research (IJSR)*. 2013. P.142–145.