Proc.10th Int. Conf. “The Experience of Designing and Application of CAD Systems in Microelectronics”, CADSM' 2009, 24-28 Feb. 2009. –IEEE Library. –2009. –P. 146-149.

Tensor Approach to the Application Specific Processor Design

Oleg Maslennikow, Anatolij Sergiyenko, Yurij Vinogradow

Abstract - **A method for mapping an algorithm, which is represented by the loop nest into the application specific structure is proposed. The method consists in translating the loop nest into the tensor equation. The tensor equation represents a set of structural solutions. The optimized solution finding consists in solving this equation in integers. The proposed limitations to the parts of the tensors help to derive the pipelined structure and simplify the mapping process. The method is illustrated by the example of the IIR-filter structure synthesis. It is intended for mapping DSP algorithms into FPGA.**

Keywords – **Algorithm mapping, SDF, DSP, FPGA.**

I. Introduction

At present, the field programmable gate arrays (FPGAs) are widely used to implement the high-speed DSP algorithms. But its programming is usually consists in the functional network drawing, which contains the ready to use modules. The design of such modules, or the programming tools like module generators remains the very complex task. This task is formalized now for the acyclic algorithms like FIR filters, and is still under investigations for the cyclic algorithms like IIR filters [1,2]. Therefore, the development of the programming tools which help to map the DSP algorithms into the networks, which are adapted to the FPGA architecture are of demand.

Consider a DSP algorithm, which can be represented by a set of recurrent equations or a loop nest. In the kernel of the regular loop nest stay one or a set of assignments like:

$$\mathrm{Sti}: a[\boldsymbol{I}] = f(a[\boldsymbol{I}+\boldsymbol{D}_1], b[\boldsymbol{I}+\boldsymbol{D}_2],...),$$

where $\boldsymbol{I}$ is an index vector, which represents a point in the iteration space, $\boldsymbol{D}_j$ is an index increment vector of the j-th variable, which represents the data dependence between $\boldsymbol{I}$-th, and $(\boldsymbol{I}+\boldsymbol{D}_j)$-th iterations. The irregular loop nest can be remapped into the regular one by the data pipelining or global transfer removing techniques [3,4].

If the loop kernel has a set of independent operators Sti, then this set can be represented as a single vector I in the iteration space. The methods of the systolic array synthesis are well known which utilize the mapping of such algorithms [3-5]. These methods are based on the affined transformation with the matrix P of the iteration space Z^n, $\boldsymbol{I} \in Z^n$ into the subspace of structures Z^m and events Z^{n-m}, so as the operator St_i belonging to the iteration $\boldsymbol{I}$ is calculated in the processor unit (PU) with the coordinates $\boldsymbol{K}_s = P_s \boldsymbol{I}$ in the clock cycle marked by $\boldsymbol{K}_t = P_t \boldsymbol{I}$, $\boldsymbol{K}_s \in Z^n$, $\boldsymbol{K}_t \in Z^n$. If we consider the usual situation when $n - m = 1$, then the conditions of such a mapping are $\boldsymbol{D}_j P_t \geq 0$ (monotony condition) and $\det P \neq 0$ (injection condition).

The systolic array synthesis methods have a set of limitations, which do not provide their direct utilization in the DSP system design. They consider that operators St_i, from a single iteration must be implemented simultaneously and their duration must be equal no more than a single clock cycle. Therefore, the complex operators could not be implemented in the pipelined mode.

In the representation a new method of DSP application specific processor design is proposed on the base of mapping the algorithms, which are given by the regular loop nests. The resulting processors have the pipelined ALUs and fit effectively the modern FPGAs.

II. Initial data for the synthesis

Consider the a single loop algorithm:

```
for $i = 1, U_i$ do
    $(y_1(i), ..., y_p(i)) = f(x_1(i - d_{i1}), ..., y_q(i - d_{iq}))$        (1)
end.
```

Here the function f is calculated using U_j binary assignments St_j. Therefore, the algorithm (1) can be represented as the following:

```
for $i = 1, U_i$ do
    {statement $St_1$}
      . . .
      $St_j$: $y[i,j] = \varphi_{j,k}(y[i-d_{i1}, j], y[i-d_{i2}, j])$        (2)
    . . .
    {statement $Stu_j$}
end,
```

where $\varphi_{j,k}(x,y)$ is the operator of the k-th type, which is implemented at the operands x, y. This loop can be transformed into the next three level loop nest, such that in the (i, j, k)-th iteration only j-th operator of k-th type is implemented.

```
for $i = 1,U_i$ do
  for $j = 1,U_j$ do
      for $k = 1,U_k$ do                                                    (3)
    if $(j,k) \in \Phi$ then $y[i,j] = \varphi_{j,i}(y[i-d_{i1}, j], y[i-d_{i2}, j])$
     end
```

Oleg Maslennikow – Polytechnica Koszalinska, Poland,
E-mail: oleg@moskit.ie.tu.koszalin.pl
Anatolij Sergiyenko, Yurij Vinogradow – National Technical University of Ukraine E-mail: aser@comsys.ntu-kpi.kiev.ua

```
    end
  end,
```

where Φ is a set of allowed couples (j,k), which give type and implementation order of operators in the algorithm (3).

As a result, the loop (1), which contains a set of different operators, can be transformed into the loop nest of three cycles (3), and its calculations are mapped into the three dimensional iteration space $K^3 = \{1 \le i \le U_i, 1 \le j \le U_j, 1 \le k \le U_k\} \subset Z^3$. The loop nest of higher dimensions can be derived analogously. Each operator is represented in the space K^3 the vector $\boldsymbol{K}_i \in K^3$. The data dependence between operators, represented by $\boldsymbol{K}_i$, $\boldsymbol{K}_l$, is equivalent to the dependence vector $\boldsymbol{D}_j = \boldsymbol{K}_i - \boldsymbol{K}_l$.

In the resulting structure each processing unit (PU) is specialized to implement a single function φ_k. The base set of such PUs for DSP applications contains the simplest PSs like adder, multiplier, ROM. Their local memory is the FIFO buffer, or a single result register.

III. Mapping the regular loop nest into the processor structure

In the methods [3-5] the algorithm graph G_A is represented in the n-dimensional space Z^n. Hence G_A is the regular lattice graph. It is represented by its compact form of a set of different vectors-edges $\boldsymbol{D}_j$ of data dependences. If the loop nest contains a set of operators like (2), then the compact form is the synchronous dataflow graph (SDF) or the scalable SDF [6]. This oriented graph has N operators-nodes $\mathbf{K}_i$, which are connected by respective dependence vectors-edges $\mathbf{D}_j$. Consider the following algorithm:

```
for $i = 1, N$ do
    for $j = 1, M$ do
        St$_1$: $a[i,j] = b[i-1, j-1]$;
        St$_2$: $b[i,j] = a[i, j]$;
    end
end.
```

This algorithm is represented by the SDF graph shown in the Fig.1.

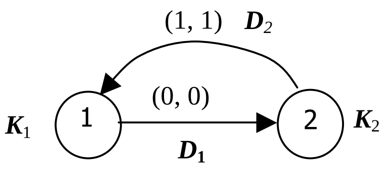

Figure1. Example of SDF graph

Vectors $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ represent the data a, b movings between operators St$_1$ and St$_2$. They labeled by the vectors of relative transfer delays (0, 0), and (1, 1). To represent the SDF graph G_{AR} in the n-dimensional space both the matrix D of the vectors $\boldsymbol{D}_j$ of data dependences and the matrix K of the vectors $\boldsymbol{K}_i$ are needed. Here the vector $\boldsymbol{K}_i$ is equal to the coordinates of the i-th operator node. An incident matrix A of the graph G_{AR} is needed to impress the linear dependence between both matrices K and D:

$$D = KA; \qquad (5)$$

A set of matrices K, D and A form, so called, algorithm configuration K_A. Due to its nature, the matrices A, K, D are tensors of both algorithm and resulting structure, and the equation (5) is the tensor equation. In [7] it is shown that the properties of many technical objects can be described by the tensor equation. Due to the tensor theory, the complex technical system can be described by its tensor. The tensor is the generalized matrix, which can be exchanged by the allowed transformations. Therefore, a set of different implementations of a system can be described by a tensor, and one implementation can be transferred to another one by some transformation of its tensor. The system synthesis consists in building of the tensor equation, and in directed search of such tensor transformation, which minimizes the effectiveness criteria. In this representation it is shown how to find the optimized structural solutions by the algorithm mapping using the principles of the tensor theory.

The next definitions and relations are true for the configuration K_A. Configuration K_A is correct, if $\boldsymbol{K}_i \ne \boldsymbol{K}_j$; $i,j = 1,...,N$, $i \ne j$, i.e. if all the vectors-nodes are placed separately in the space Z^n.

A back linear dependence between configuration matrices is present, i.e.

$$K = D_0 A_0^{-1}, \qquad (6)$$

where A_0 is the incidence matrix of the maximum spanning tree of the graph G_{AR}, D_0 is the matrix of the vectors-edges of this tree, including the base vector which connects the graph node with the coordinate system.

The sum of vectors-edges $\mathbf{D}_j$, belongig to a graph cycle, must be equal to a zero, i.e. for the i-th cycle

$$\sum_j b_{i,j} \boldsymbol{D}_j = 0, \qquad (7)$$

where b_{ij} is the element of the i-th row of the cyclomatic matrix of the graph G_{AR}.

Configurations $C_{A1} = (K_1, D_1, A_1)$ and $C_{A2} = (K_2, D_2, A_2)$ are equivalent if they are correct and represent an algorithm graph, i.e. $A_1 = A_2$. Correct configuration C_{A1} is equivalent to the configuration C_{A2} iff $A_1 = A_2$ and $K_2 = F(K_1)$, where F is the injection function. For example, the following transformations give the equivalent configurations: vector $\boldsymbol{K}_i$ transposition in the space Z^n, row or column transposition of the matrix K_1, multiplication of the matrix K_1 to the non-singular matrix P.

Due to the tensor theory, any tensor object description must have the invariant tensor, which is immune to any tensor transformations. Here the matrix A and its submatrix A_0 represent the invariant tensors. The matrix K codes some variant of the synthesized structure. The structure optimization consists in generating of equivalent configurations, which are different in their matrices K, and in selection of the best one due to the some criterion.

The processor structure graph G_s is represented by its structure configuration $C_{As} = (K_s, D_s, A)$, where K_S is the matrix of vectors-nodes $\boldsymbol{K}_{Si} \in Z^m$, which give the PU coordinates, and D_S is the matrix of vectors-edges $\boldsymbol{D}_{sj} \in Z^m$, which represent the connections between PUs, $m < n$.

The event configuration $C_T = (K_T,D_T,A)$ consists of the matrix K_T of the vectors $\boldsymbol{K}_{Ti} \in Z^{n-m}$, matrix D_T of vectors $\boldsymbol{D}_{Tj}$, and matrix A. Here vectors K_T represent the events of the operator implementation. In the correct configuration C_T vector $\boldsymbol{D}_{Tj} = \boldsymbol{K}_{Tl} - \boldsymbol{K}_{Ti}$ means that the operator, represented by $\boldsymbol{K}_{Ti}$, must precede the operator, represented by $\boldsymbol{K}_{Tl}$.

The timing function $R(\boldsymbol{K}_{Ti}) = t_i$ performs the mapping of the space of events Z^{n-m} to the time axis, and derives the time of the operator implementation.

The configuration C_T is correct, in other words, the precedence condition is true if for any couple of vectors $\boldsymbol{K}_{Ti}$ and $\boldsymbol{K}_{Tl}$ the inequality is true $R(\boldsymbol{K}_{Tl}) > R(\boldsymbol{K}_{Ti})$, where $\boldsymbol{K}_{Ti}$ precedes $\boldsymbol{K}_{Tl}$.

If the function R is linear and monotonous one then the configuration C_T is correct iff $\boldsymbol{D}_{Tj} \succcurlyeq 0$, $j = 1,...,M$, where $\boldsymbol{D}_{Tj}$ the vectors-nodes of the SDF, which are not marked by the relative transfer delays (or zeroed ones).

The function $R(\boldsymbol{D}_{Tj})$ gives the delay between the variable computing in one PU and entering the another PU, i.e. the higher limit of the FIFO buffer length.

Consider the mapping of the algorithm (2) into the structure, which calculates the loop kernel in the pipelined mode with the period of L clock cycles. When this algorithm is represented in the three dimensional index space, the vectors $\boldsymbol{K} = (j, k, i)^T$, where j,k,i means operator number, operator type, and cycle number respectively. Similarly the additional dimension q of the clock cycle is added to the algorithm configuration, then $\boldsymbol{K} = (j, k, i, q)^T$. The vector-edge, which represents the interiteration dependence, is equal to $\boldsymbol{D}_b=(0,0,-p,0)$, where p is the distance between iterations.

Algorithm configuration C_A is equal to the composition of structure configuration C_S and event configuration C_T, and if $\boldsymbol{K}_l = (j, k, i, q)^T$, then $\boldsymbol{K}_{Sl} = (j, k)^T$ and $\boldsymbol{K}_{Tl} = (i, q)^T$. In the vector $\boldsymbol{K}_{Sl} = (j,k)^T$, the coordinates j,k are equal to the PU number, where the l-th operator of the k-th type is implemented.

Firstly the space component of the mapping is searched. The matrix K_S forming is the combinatorial task. By this process M_K operators of k-th type are distributed among more than $]M_K/L[$ PUs of the k-th type. In the matrix K_S M_S groups of equal columns are formed, each of them contains up to L columns, where M_S is the PU number in the resulting structure. The j-th PU has the maximum loading if the number of columns with the j-th coordinate is equal to L. Then the matrix D_S is derived from the equation $D_S = K_S A$.

The time component of the mapping represented by the matrices K_T and D_T is searched with respect to the conditions of the correctness of the algorithm configuration and event configuration, and equation (7). Besides, the algorithm is implemented correctly with the iteration period L iff

$$\forall \boldsymbol{K}_{Ti} \in K_T (\boldsymbol{K}_{Ti} = (i, q)^T, i \geq 0, q \in (0, 1,...,L\text{-}1)).$$

The strategies of searching of the space and timing components can be investigated in the following example of the structure synthesis.

IV. Example of the processor synthesis

Consider the synthesis of the second order IIR filter structure, which calculates the equation:

$$y[i] = x[i] + a{\cdot}y[i-2] + b{\cdot}y[i-1].$$

This equation is calculated by the following loop:

```
for i = 1, $N$ do
    St$_1$: $y_1$[i] = $a$*$y$[i–2];
    St$_2$: $y_2$[i] = $b$*$y$[i–1];
    St$_3$: $y_3$[i] = $x$[i] + $y_1$[i];
    St$_4$: $y$[i] = $y_2$[i] + $y_3$[i];
end.
```

The SDF graph of this algorithm is shorn in Fig.2.

Each operator is calculated no less then a single clock cycle. The loaded edges mean the delays of the variable $y[i]$ to one and two cycles, and could not express the delay of the operator St$_4$. Therefore, in these edges additional nodes are set. The modified SDF graph is shown in the Fig.3.

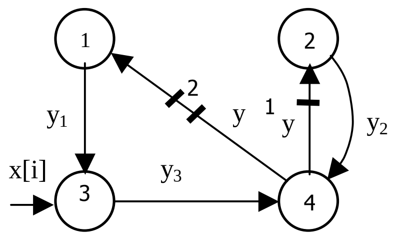

Figure 2. Initial SDF graph of the IIR filter

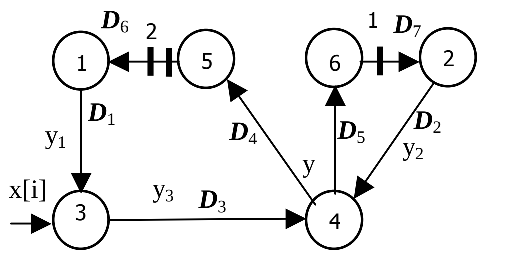

Figure 3. Extended SDF graph of the IIR filter

This graph represents the following algorithm

```
for $i$ = 1, $N$ do
    St$_1$: $y_1[i]$ = $a$*$y_5[i–2]$;
    St$_2$: $y_2[i]$ = $b$*$y_6[i–1]$;
    St$_3$: $y_3[i]$ = $x$[i] + $y_1[i]$;
        St$_4$: $y[i]$ = $y_2$[i] + $y_3$[1];
        St$_5$: $y_5[i]$ = $y[i–2]$;
        St$_6$: $y_6[i]$ = $y[i]$;
end.
```

The calculation period is $L = 2$, which means that a single couple of adder and multiplier can calculate it.

By the search of the space component the permissible coordinates $\boldsymbol{K}_{si}$ are set:

$$K_S = \begin{array}{c} j \\ k \end{array}\begin{pmatrix} 112233 \\ 112200 \end{pmatrix}.$$

Here coordinates k = 0, 1, 2 mean multiplication, addition, equality operators. The matrix $\boldsymbol{D}_S$ is derived from the equation

$$D_S = K_S A = \begin{array}{c} j \\ k \end{array}\begin{pmatrix} 110\ 1\ \ 1\ -2-21 \\ 110-2-2\ 1\ \ 1\ 1 \end{pmatrix}.$$

When the time component of the mapping is searched, the known coordinates are set in the weighted vectors-edges $\boldsymbol{D}_{T6}$=(–2 0)^T and $\boldsymbol{D}_{T7}$ = (–1 0)^T. The timing function is selected R=(L 1)=(2 1). To minimize the register number the vectors $\boldsymbol{D}_{Tj}$, which leave the nodes 1,…,4 must have the coordinates providing $R{\cdot}\boldsymbol{D}_{Tj}$ = 1 or 2, i.e. (0 1)^T, (1 –1)^T, or (1 0)^T, which provide the monotony condition.

To provide the injection condition, the vectors $\boldsymbol{K}_{Ti}$ with equal coordinate q must be different, for example, when $\boldsymbol{K}_{T1}$ = (X 0)^T, or (X 1)^T, then $\boldsymbol{K}_{T2}$ = (X 1)^T, or (X 0)^T where X is unknown value. The coordinates q of the vectors $\boldsymbol{D}_{Tj}$ are derived from the set of equations:

$$D_T = K_T A;$$
$$\boldsymbol{D}_{T1}+\boldsymbol{D}_{T3}+\boldsymbol{D}_{T4}+\boldsymbol{D}_{T6} = 0;$$
$$\boldsymbol{D}_{T2}+\boldsymbol{D}_{T5}+\boldsymbol{D}_{T7} = 0.$$

Due to these conditions the following solution is found:

$$K_T = \begin{array}{c} i \\ q \end{array}\begin{pmatrix} i\ i\ i\ i+1\ i+2\ i+1 \\ 0\ 1\ 1\ \ 0\ \ \ 0\ \ \ 1 \end{pmatrix}.$$

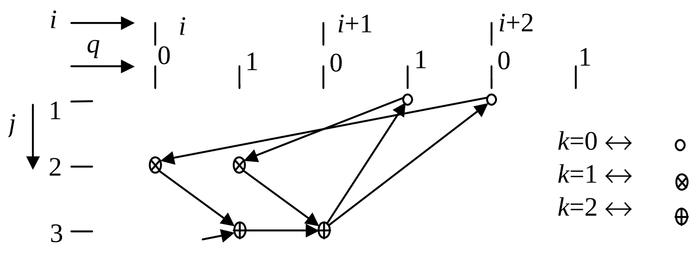

Figure 4. Algorithm configuration of the IIR filter

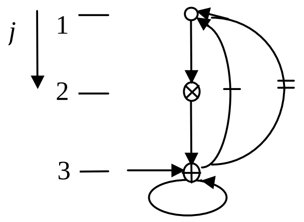

Figure 5. Structure configuration of the IIR filter

Fig.4 illustrates the derived algorithm configuration, and the Fig.5 does the respective structure configuration. This solution is distinguished by maximum hardware loading of the PUs and operation in the pipelined mode. It is the only structural solution of the second order IIR filter, in which the minimum clock cycle is equal to a single multiplier delay, and the input data run with the period of two cycles.

V. Conclusion

A method of application specific processor design is proposed which is based on the tensor theory of the system design. Its expansion was proven and widely used in the successive development of a set of DSP applications configured in the FPGAs, for example, published in [9, 10].

References

[1] J.Isoaho, J.Pasanen, O.Vainio “DSP Sytem Integration and Prototyping With FPGAs” *J. of VLSI Signal Processing.* V 6, 1993, pp. 155-172.

[2] “System Generator for DSP. Getting Started Guide” August, 2007, 85p. See http://www.xilinx.com

[3] S.V.Rajopadhye “Synthesizing systolic arrays with control signals from recurrence equations” *Distributed Computing.* V3, 1989, pp. 88-105.

[4] J.Fortes, D.Moldovan “Data broadcasting in linearly scheduled array processors” *Proc. 11 th Annual Symp. on Comp. Arch*., 1984, pp. 224-231.

[5] S.Y.Kung “VLSI Array Processors” Eigenwood Cliffs, N.J.: Prentice Hall, 1988.

[6] S.Ritz, M.Pankert, and H.Meyr “Optimum vectorization of scalable synchronous dataflow graphs” *Proc. Int. Conf. on Application Specific Array Processors*. October. 1993.

[7] G.Kron “Tensor analysis of networks” MacDonald, London, 1965. 635 p.

[8] A. Sergyienko, O. Maslennikov. “Implementation of Givens QR Decomposition in FPGA” *Lecture Notes in Computer Science*, Springer, 2002, Vol. 2328, pp. 453-459.

[9] A. Sergyienko, V.Simoneko “DSP algorithm mapping into FPGAs” *Proc. Int. Conf. Simulation*-2006. Kiev. Energy Problem Modeling Institute of NAS of Ukraine. 2006. pp. 189-193.

[10] O.Maslennikov, Ju. Shevtshenko, A. Sergiyenko “Configurable Microprocessor Array for DSP Applications” Lecture Notes in Computer Science. V. 3019. 2004. pp. 36-41.