Partitioning of DSP tasks to Kahn network

A. Žvironas, E. Kazanavičius

Kaunas University of Technology,Digital Signal Processing Laboratory, Department of Computer Engineering

Studentų 50-214[c], LT-3031, Kaunas, Lithuania, E-mail: {zviron,ekaza}@dsplab.ktu.lt

Introduction

Recently the number of digital signal processing (DSP) applications is increasingly growing. Modern extensive application domains are audio processing, digital communications, speech recognition, spatial positioning, etc. One essential feature of the mentioned applications is hard real time processing of real world signals. Occasionally chosen processing architecture may not satisfy processing time and hardware resource restrictions set by the application. Therefore, selection of application-dedicated architecture is needed. Architecture selection presents design-space-exploration task. This task in general is hardly solvable as there exist a lot of design alternatives to explore: number of different DSP algorithms, processing elements, design techniques and tools. Usual way to make architecture selection more effective is to build design methodology. The methodology should constrain in some way design space and use systematic approaches to solve problematic design aspects.

A lot of different methodologies for the DSP architecture design have been proposed. Recent methodologies mainly are dedicated for the evaluation of DSP architectures from the application system level description. Most typical ones were proposed in the projects Ptolemy [11], Match [10], also the SPADE methodology [7], etc. These methodologies deals mainly with application and architecture models, however, they provide less attention to mapping these application model into real-time hardware prototype.

This paper proposes a methodology for mapping the DSP task into different processing architectures. This methodology includes task specification, algorithm analysis, functional partitioning and mapping the task presented by Kahn network into the hardware-prototyped DSP architecture. The mapping process is illustrated with the multi-channel correlation processing task that is common in such applications as various measurements, sonar and radar positioning [1,4].

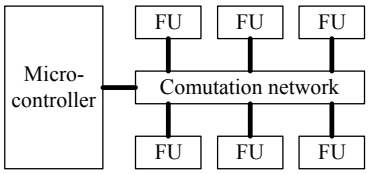
Methodology of mapping a DSP task into processing architecture

Overview of DSP architectures. DSP processing is possible in several different types of devices. Most applicable for this purpose are DSP processors with different cores, versatile programmable logic devices (FPGA, CPLD) [3,13,14], also heterogeneous architectures that contain several different type devices (Fig.1).

For multi-channel application most suitable are multiprocessor systems consisting of several processors or processors with several cores e.g. TMS320C80 and TMS320C6000. These processors can execute parallel algorithms and usually execute arithmetic operations (accumulation and multiplication) in a single machine cycle. Therefore, DSP processors can execute basic DSP functions such as filtering, correlation, FFT [4] in real time. A lot of ready-to-use routines and architectures of basic DSP operations exist with the known execution time and hardware resources [8].

Lately, the various modifications of programmable logic devices (PLD) became applicable for DSP applications. For example, FPGA Virtex-II Pro by Xilinx [13] provides chip hardware resources for implementation of 556 18x18 bit multiplication blocks.

Heterogeneous architectures (Figure 1) generally contain the microcontroller, commutation network and several functional units (FU), such as various DSP processors and FPGA. Such architectures mostly give use for the applications featuring simultaneously several different information processing modes e.g. signal processing, control, data transmission, etc.



FU – functional unit

Fig. 1. **Heterogeneous architecture**

Methodology. We propose the methodology for DSP task mapping into processing architecture (Fig.2). This methodology is considered to assist DSP system designers to choose the architecture rational for their application. In the first step designer must describe the task in imperative or Nested Loop Program (NLP) computation model, such as C or Matlab semantics. The manual derivation of parallel description from sequential program is difficult and tedious task. Some special compilers are used to form instruction level parallelism from the original specification [5].

For parallel processing tasks more suitable specifications are computation models of *process network* type [2,6]. Process network describes the application as a network of firing concurrent processes. In order to find concurrent processes we suggest decompose the task into the simplified graph of dependencies (Fig.3).

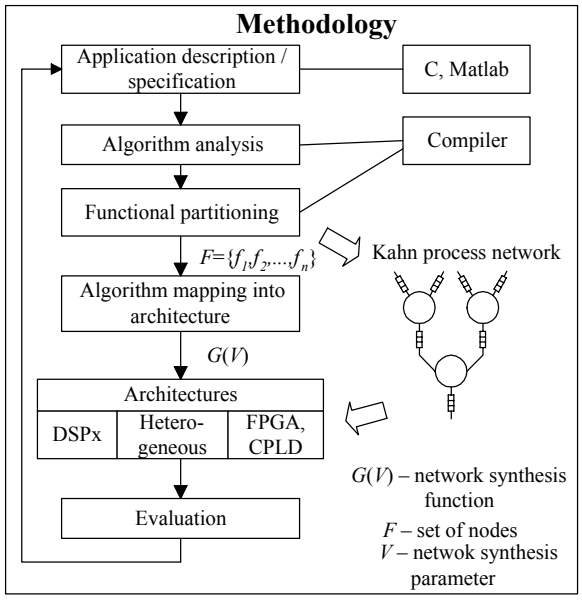


Fig. 2. **Methodology of mapping the DSP task into the architecture**

Consequently, this graph has to be overlaid by the Kahn process network model.

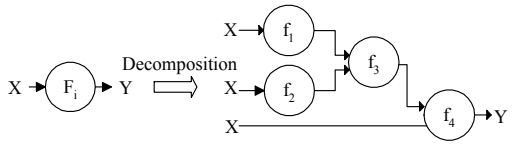


Fig. 3**. Decomposition example**

Kahn network [7] describes computation model in which computation processes combined with unbounded FIFO channels form a network. The action of any process node is driven by input data and stops if at least one of input channels becomes empty. Writing of the process data into the output channel is not blocked.

Since the DSP task is assumed to be multi-channel, so the number of channels is defined already in the task specification step. Compiler makes functional decomposition. Next, the decomposition result is used by Kahn network synthesis function $G(V)$, where V presents the network synthesis parameter vector. It contains the parameters $V =< M_i, F_i >, \quad i = \overline{1,N}$, where M_i is the number of *i*-th of architecture channel, F_i is the set of Kahn network nodes $\{f_1, f_2, \ldots, f_l\}$. Each node is written as the parameter vector $f_l =< s_l^{ki}, p_l^{ko}, o_l, ki_l, ko_l >$, where o_i - the operation type, ki_l – number of inputs, ko_l – number of outputs, s_l^{ki} - depth of the input FIFO, p_l^{ki} - depth of the output FIFO (Fig.4).

In the next design step the network synthesis function $G(V)$ has to be mapped to the architecture, which is implemented as set of Kahn processors. The Kahn processors architecture is described later.

The last step is evaluation of the task in the architecture chosen by the designer. This step is supported by the database that contains the information about estimations of arithmetic and basic DSP operations

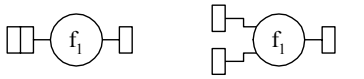


Fig. 4**. Examples of Kahn network node**

in different architectures. The evaluation step also should include the estimator having the responsibility to choose the appropriate architecture for the task. The estimator checks fulfillment of the following conditions: $(\varepsilon \le \varepsilon_0) \,\&\, (T \le T_0) \,\&\, (C \le C_0)$, where T is the execution time in the architecture, C is the architecture resources, ε - processing error, ε_0, T_0, C_0 are the corresponding constraints of the application [15]. The estimator provides the inference whether the chosen architecture is suitable for the task. In the case of the unfavorable inference the designer has to check other design space alternatives.

Description of Kahn processor. For the mapping of the Kahn network into the architecture the designer must describe the architecture like shown in Fig.5.

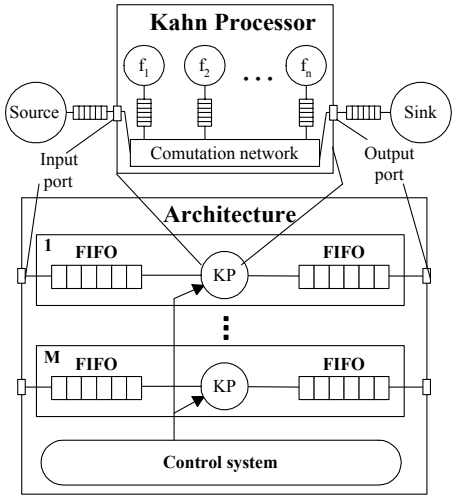


Fig. 5. **Kahn processor architecture**

The architecture may contain one or multiple Kahn processors (KP) - that depends on the available hardware resources in the architecture. Multiple KP are effective for implementation of multi-channel DSP tasks or single-channel tasks that are computation-intensive and need to be spread over several processors. The Kahn processor consists of set of Kahn network nodes $\{f_1, f_2, \ldots, f_l\}$ that designate arithmetic operations <+,–,*> and basic DSP operations. As mentioned earlier, the network synthesis function $G(V)$ composes Kahn process network. The network nodes f are interconnected by bounded FIFO channels with different depths. Kahn processors are managed by the control system that operates according to the parameter vector V. The parameter M_i indicates to the control system the KP that should be fired.

This approach allows effective evaluation of the architecture suitability to the application. Since the descriptions of KP in the architecture are identical, the

estimated processing time in a single KP lets to predict the processing time of the task assigned in multiple KP.

Case of multi-channel correlation processing task

Generally, correlation processing task fits into the functional structure shown in Fig.6. It contains M signal processing channels each of them having preprocessing function F_{prep}, DSP function F_{cf} for computing cross-correlation function (CCF) and post-processing function F_{post}. Usually F_{prep} designates analog filtering and digitization of the signal and F_{post} may cover a wide spectrum of functions such as peak detection, envelope extraction, averaging or sub-sample interpolation. Variables $y(t)^{1..M}$ designate continuous real time input signals, $y[n]^{1..M}$ - preprocessed digital samples, $r_{xy}[n]^{1..M}$ - CCFs between $y[n]^{1..M}$ and the reference signal $x[n]^{1..M}$, and $z^{1..M}$ - result obtained after the post-processing of the CCF.

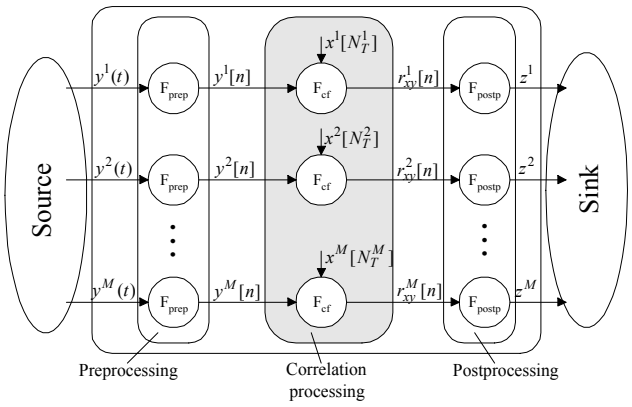


Fig. 6. **Structure of the multi-channel correlation task**

Computation of CCF in many applications requires a significant, sometimes a major part of overall DSP time and hardware resources. A reasonable example of such an application is multi-channel correlation processing in the ultrasonic vision system for a mobile robot [12]. Therefore, the further demonstration of the methodology shifts to mapping of the multiple F_{cf} into DSP processor architecture and the performance evaluation of the architecture.

Analysis of CCF computation algorithms. There exist several different algorithms to compute CCF [1,4,9]. Most common are the straightforward algorithm (A1) and the fast algorithm (A2) based on the fast Fourier transform [1,4,9]. The first algorithm computes the CCF of two sampled sequences x_i and y_i, each of length N, according to the following formula [1,4]:

$$r_{xy}[k] = \frac{1}{N}\sum_{i=1}^{N} y[i]\cdot x[i+k], \quad k \in \overline{0..N-1}. \qquad (1)$$

If N is sufficiently large (N>128 [4]), the second method A2 includes less arithmetic operations and is faster than the direct method A1. The A2 method is described by the expression [1,4]:

$$r_{xy} = \frac{1}{N} FFT^{-1}[FFT^{*}(x)\cdot FFT(y)], \qquad (2)$$

where FFT denotes the fast Fourier transform (*FFT*)and FFT^{-1} - the inverse *FFT*. In fact, this method requires computation of three FFTs and complex multiplication of signal spectrums. These DSP operations can be scheduled in the architecture differently and give different values of CCF computation time and hardware resources.

Example of CCF algorithm mapping. For this example the method A1 was taken because it can be transformed into series of parallel forms. Its algorithm is similar to the FIR filter's algorithm and described in Matlab by the expression:

```
for i=1:n;
 r(i)=0;
 for j=i:m;
  r(i)=r(i)+y(j)*x(j-i+1);
 end;
end;
```

This script corresponds to the CCF computation with the shift to the right. The CCF part with the shift to left can computed in much similar way.

After the decomposition we get the network synthesis function $G(V)$. It forms the Kahn process network that is shown in Fig.7. It is seen from the Kahn network that several values of the CCF can be computed in different channels.

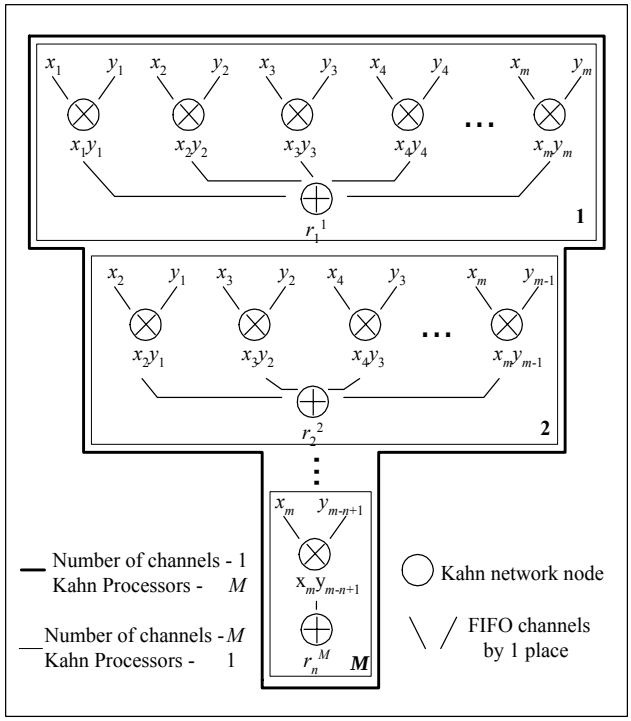


Fig.7. **CCF computation fragment in Kahn network**

After the algorithm was decomposed down to arithmetic operations we evaluate an execution time. The

experiment was based on the Motorola DSP5630x [8] processor that executes multiply-accumulate operation within 83 nanoseconds at 60 MHz clock cycle. Initialization cycles were not taken into account. If the CCF is computed in a single channel with the M Kahn processors the processing time is equal:

$$T = 83 \cdot \sum_{i=1}^{M} (m - i + 1) \quad [\text{ns}], \; m \in Z, \qquad (3)$$

where Z is the set of positive values.

If x and y arrays contain $m = 1024$ samples each (at this case $m = M$), then $T = 0.044$ s.

In other case, if there are assigned single KPs in each of several channels the CCF several values can be simultaneously processed in different channels. Then the execution time equals to:

$$T = 83 \cdot m \quad [\text{ns}], \; m \in Z. \qquad (4)$$

Overall CCF processing time is equal to the time of the CCF first value computation. For example, if $m = 1024$ samples, then $T = 0.085$ ms.

Conclusions and future challenges

This methodology was suggested with the aim to ease DSP task mapping into the processing architecture and evaluate the application-to-architecture match. Following the methodology, the DSP task is described by C or Matlab script and then compiled into the Kahn network synthesis function, which later used to compose the task-equivalent Kahn network. Each node in the network designates arithmetic or basic DSP operations – the granularity depends on the particular DSP algorithm used. The architecture evaluation step is needed to make the inference whether the chosen architecture is suitable for the task. In the case of the unfavorable inference the designer has to check other design space alternatives.

Future works should address the following challenges:

- Implementation of the tool for the automatic decomposition and Kahn network composition.
- Building the database for the architecture estimator.

References

1. **Bendat J. S., Piersol A. G.** Engineering Applications of Correlation and Spectral Analysis. John Wiley & Sons. 1980.
2. **Edwards S., Lavagno L., Lee E. A. and Sangiovanni-Vincentelli A.** Design of Embedded Systems: Formal Models, Validation and Synthesis. Proc. of the IEEE. 1997. Vol. 85. No.3. P.366-390.
3. **Ercegovac M. D., Lang T., Moreno J. H.** Introduction to Digital Systems. John Wiley & Sons. New York. 1999.
4. **Ifeachor E. C., Jervis B. W.** Digital Signal Processing: A Practical Approach. Addison-Wesley. 1993.
5. **Kienhuis B.** Matparser: An Array Dataflow Analysis Compiler. Technical report. Department EECS, University of California at Berkeley, Cory Hall 524, Berkeley, California, 94720, USA, Feb. 2000.
6. **Lee E. A., Parks T. M.** Data flow process network. Proceeding of the IEEE. May 1995. Vol. 83. No.5. P.773-799.
7. **Lieverse P., Van der Wolf P., Deprettere Ed., Vissers K.** A Methodology for architecture exploration of heterogeneous signal processing systems. Proc.1999 IEEE Workshop on Signal Processing Systems (SiPS'99). 1999. P.181-190.
8. **Motorola.** DSP56300 Family Manual. 24–bit Digital Signal Processor. Revision 2.0. August 1999
9. **Oppenheim A. L., Shafer R. W**. Discrete-time signal processing. Prentice-Hall, 1999.
10. **Periyayacheri S., Nayak A., Jones A., Shenoy N., Choudhary A. and Banerjee P.** Library Functions in Reconfigurable Hardware for Matrix and Signal Processing Operations in Matlab. Proc. 11th IASTED Parallel and Distributed Computing and Systems Conference (PDCS'99). Cambridge, MA. November 1999. (http://www.ece.nwu.edu/cpdc/Match/papers.html).
11. **Pino J. L., Ha S., Lee E. A. and Buck J. T.** Software Synthesis for DSP Using Ptolemy. Journal on VLSI Signal Processing. January 1995. Vol.9. No.1. P.7-21.
12. **Venteris R.** Exploration of DSP architectures in ultrasonic measurement applications. Ultragarsas, Nr.1(42). KTU. Kaunas. 2002.
13. **Xilinx**, **Inc.** Virtex-II Pro Platform FPGA Handbook. (http://www.xilinx.com/publications/products/v2pro/handbook/)
14. **Xilinx, Inc**. The Programmable Logic. Data Book. 1996.
15. **Ясиявичюс Р. Ю.** Параллельные пространственно-временные вычислительные структуры. Вильнюс: Мокслас. 1988.

A. Žvironas, E. Kazanavičius

SSA uždavinio padalijimas į Kahn'o tinklą

Reziumė

Aprašoma daugiakanalio uždavinio paskirstymo į skirtingas skaitmeninių signalų apdorojimo (SSA) architektūras metodika. Ji apima uždavinio specifikavimą, algoritmo analizę, funkcinį padalijimą ir uždavinio paskirstymą į architektūroje realizuotą Kahn'o tinklo prototipą. Paskirstymo procesas pateikiamas daugiakanaliu koreliacijos apdorojimo uždaviniu, kuris randamas tokiuose taikomuosiuose uždaviniuose kaip įvairūs matavimai, radiolokacinis ir sonarinis padėties nustatymas.

Pateikta spaudai 2002 07 8

DOI: 10.5755/j01.u.43.2.8120