# Design and analysis of DSP systems using Kahn process networks

**J. Čeponis, E. Kazanavičius, A. Mikuckas**

*Digital Signal Processing Laboratory*

*Kaunas University of Technology*

## Introduction

Currently signal processing systems are designed using various methods and tools [1]. In DSP systems design and implementation very important factor is project realization time. Various tools and methods are created to reduce this time. Kahn process network model, discussed in this paper, is one of such methods [4].

While analyzing current situation in system design using Kahn process networks one can notice that designers concentrate on one problem or problem class. *Ptolemy* [3], *Artemis* and *Compaan* projects were analyzed, which use Kahn process network model in system design. In these projects network model is tightly related with the system being modeled.

System's Kahn process network can be modeled using personal computer. This helps in finding critical network operation points. Performance can be improved dynamically changing network parameters.

Hardware systems (embedded systems, programmable logic) dedicated to a particular problem class are used to minimize processing time. Kahn process network implementation in the hardware system would help in verifying efficiency of the solution in the particular situation [6].

## Kahn process network

The Kahn process network is a computation model in which many concurrent processes can be executed simultaneously. This model is presented as an oriented graph, where every node represents a process and every arc represents an unbounded FIFO buffer used for data transmission [2].

Kahn process network can be described as a graph G=(N, D, F), where

- *N* is a set of vertexes, which correspond to network nodes;
- *D* is a set of arcs, which correspond to network channels;
- *F* is a function, which determines an ordered pair of vertexes (network nodes) $(n_i, n_j)$, $n_i, n_j \in N$ corresponding to every arc of the graph.

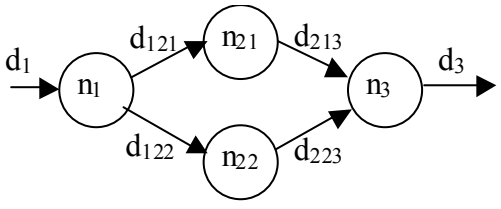


Fig. 1. **Kahn process network as a graph**

Network nodes are connected with unidirectional FIFO channels thus $F(d) = (n_i, n_j)$ is an ordered pair.

Producer node writes and inserts data into the queue, while consumer node reads the data and removes it from the queue. This is a natural model for describing signal processing systems where infinite streams of data samples are incrementally transformed by a collection of processes executing in sequence or in parallel.

Reading from an empty input channel is blocked. Writing into the channel is never blocked because of the unbounded channel length. In this model computation results do not depend on the execution order of the nodes. The only thing that depends on the execution order is the computation time.

The node trying to read from an empty channel is blocked. The node cannot check to see whether the channel is empty before trying to read from it. While being blocked the node does not do any other operation including reading from the other channels if they exist. Because of the determinate nature of the computation model, computation result is independent of the execution order: Kahn process networks can be executed sequentially or in parallel with the same outcome.

There are two possible methods for network scheduling while implementing Kahn process network in the system where concurrent node execution is not possible [5,7]:

1. Static: the order of nodes execution is specified before network execution.
2. Dynamic: nodes execution order is changed during network execution according to the situation.

While implementing the network scheduling one must carefully consider the operations of the network nodes and especially the path of data transmission. Static network scheduling method is not suitable for sequential computations. Static network scheduling has no sense when we are executing nodes concurrently. It is important only when two nodes share the same computation resources.

While implementing dynamic network scheduling one must consider current network situation [4]. One of the most important parameters is the amount of data in channels. Considering this, network node is chosen which performs data reading and computations. This ensures network operation efficiency and memory and computation resources optimization.

## Network modifications

**Observing network operation.** Network channels operation parameters are being observed and analyzed

during the initial stage of system modeling. In this stage it is important to simulate network operation without performing any calculations that would not let evaluate network efficiency.

Observer is assigned to each network channel. The observer observes, analyzes and saves these parameters:

- channel length and state;
- data stream intensity;
- network operation changes when channel length is reduced or increased.

Initially, data transmission efficiency is evaluated and used memory amount is optimized. Then network nodes are analyzed. Considering the node's operations the node observer observes, analyzes and saves these parameters:

- execution time and operations quantity;
- node's execution and idleness time;
- node's priority;
- the state of input and output channels.

Data packet path through the network is observed aiming to find optimal network operation parameters: channels lengths and nodes priorities.

During network modeling all channels and nodes must be observed. General observer is also required. Observation functions are implemented in channels and nodes themselves. Node can save the information about reading and writing operations performed. From this we can judge about channel occupation. Saving information about intensity of operations performed helps in calculating needed resources.

On the contrary to traditional Kahn process network model the presence of data in the channel is checked. Node checks to see whether there is a data packet prepared in the channel before reading. In final implementation (when system is fully tested) data observation and buffer checking function can be turned off.

**Memory allocation.** Traditional Kahn process network model has unbounded FIFO channels. In real systems there is bounded amount of memory. Thus we need to ensure proper memory allocation among all channels.

Lets say we have a network with $K$ channels and we can allocate $M$ amount of memory in this network. The possible strategies for memory allocation are:

- all channels get $M/K$ amount of memory;
- channel length is estimated according to expected data stream intensity (length of all channels does not exceed $M$);
- channel length is alternating according to network state.

First two memory allocation strategies are static. Memory is allocated to channels in the beginning and it's amount allocated to each channel does not change. The third memory allocation strategy is dynamic: memory is allocated during network operation. Initially channels are allocated $M$-$R$ memory amount ($M$ – all the memory we have; $R$ – memory dedicated for increasing channel length). Initial memory allocation can be twofold:

- all channels get the same memory amount;
- channels are allocated different memory amounts according to expected data stream intensity.

During network operation channels occupation is observed. If channel gets full, its length is increased. Amount of memory allocated to the channel also depends on the data type (integer, real, array, pointer etc.) transmitted through this channel. The amount of allocated memory must be proportional to the length of the data packet being transmitted. Channel is implemented as a FIFO buffer thus we need to save the information about the order of the records.

The amount of memory required in the system is:

$$X_{mem} = m * x * (n + y), \quad (1)$$

where $m$ – number of channels, $x$ – maximum channel length, $n$ – data packet length, $y$ – amount of memory needed for saving the pointer to the next element.

During system operation memory allocator must ensure that the amount of used memory would not exceed $X_{mem}$.

**Feedback possibility.** Often while implementing certain algorithm we need to use a feedback feature in the system. In traditional Kahn process network such configuration is not possible because the process would block waiting for data from the channel where data will be available only after the series of calculations. To solve this problem we introduce the default value. This conflicts Kahn process network determinism feature but helps preventing deadlocks and enables feedback possibility. For example, we have an implementation of IIR filter (see Fig.2).

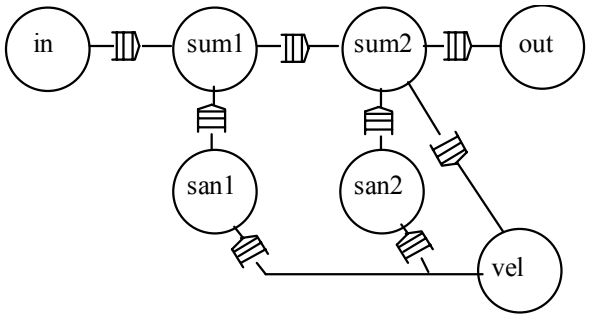


Fig. 2. **IIR filter Kahn process network model**

Initially node *in* writes the first packet into his output channel. Node *sum1* reads the packet and tries to read the data from the other input channel. If there is no data in this channel the whole network operation is blocked, because the rest of the nodes also do not get the data. At that time node *in* can overflow his output channel because reading from is channel has stopped. To ensure further network operation node's *sum1* execution must be continued even if there is no data in the input channel.

After a series of attempts to read the data node uses the default value. For example IIR filter can use 0.

Default value helps to prevent deadlocks but inappropriate use of this value can cause wrong calculation results. Avoiding this we need to introduce sufficient node execution blocking periods.

**Dynamic network reorganization.** The computation resources among network nodes must be properly allocated to ensure continuous real time data flow processing. The mechanism for changing nodes priorities is necessary. This

mechanism would dynamically change nodes priority according to the intensity of input data and resources needed for computations. This would enable optimal use of available resources.

During network operation one can change not only network parameters but also reorganize the whole network configuration. For example we have seven-node IIR filter Kahn process network model (see Fig. 2). Nodes *sum1* and *sum2*, nodes *san1* and *san2* can be grouped. This grouping is quite straightforward because their operations are the same. The new network model is demonstrated in Fig. 3.

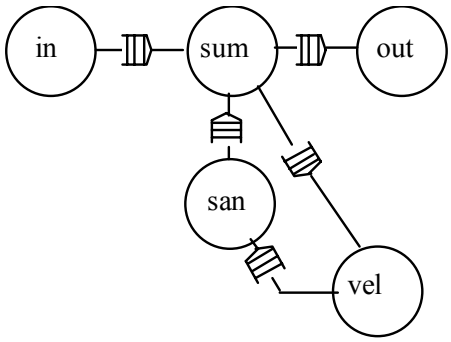


Fig. 3. **Modified IIR filter Kahn process network model**

Such modifications to network configuration may increase computation time but reduce amount of needed memory and free some computation resources.

## Implementing and analyzing modifications

During Kahn process network model investigation personal computer (AMD Duron 700MHz, 320MB RAM, 20GB HDD) with software (Ms Windows 2000 SP2, Ms Development Environment v.7.0.9254, Ms Visual C#.NET 7.0) was used.

**FIR filter.** To demonstrate Kahn process network operation 11 taps FIR filter was implemented. This filter performs filtering of two summed sinusoids (see Fig. 4).

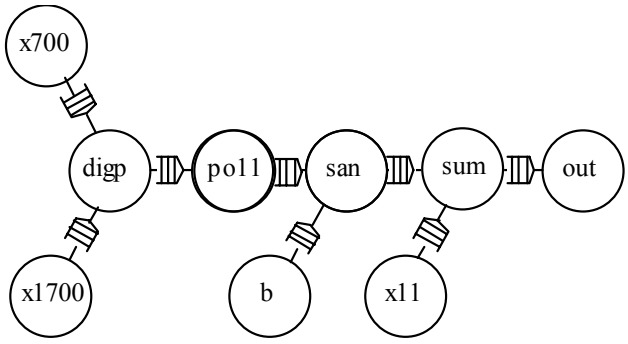


Fig. 4. **FIR1 filter Kahn process network model**

In this implementation FIR filter is expanded into small operations. Network nodes *x11*, *po11* and *b* do not perform any operations with data only its transmission. Reducing the number of network nodes FIR filter was implemented using the scheme presented in Fig. 5.

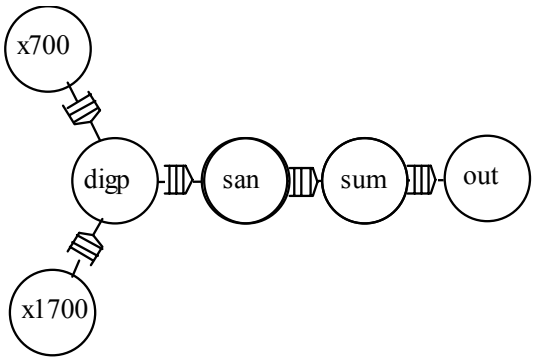


Fig. 5. **FIR2 filter Kahn process network model**

In this network nodes *x700*, *x1700*, *digp* and *out* remain the same. Node *san* performs grouping of packets and multiplication by coefficients and node *sum* sums and gives the results to the node *out*.

**IIR filter.** IIR filter was implemented using two different Kahn process network structures: with minimum number of network nodes and expanding filter taps. IIR filter was chosen for implementation to test feedback mechanism.

3 taps IIR filter Kahn process network model with minimum number of nodes is presented in Fig. 6.

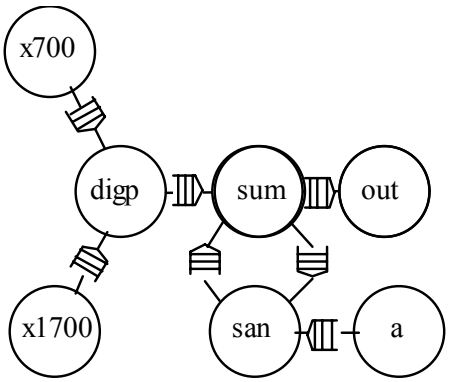


Fig. 6. **IIR1 filter Kahn process network model**

The expanded IIR filter Kahn process network model is presented in Fig. 7. Separate nodes *sum* and *san* are used for every tap. New node *mux* is introduced which spreads data among the nodes. Node *a* is liquidated and his functions are passed to nodes *san1*.. *san3*.

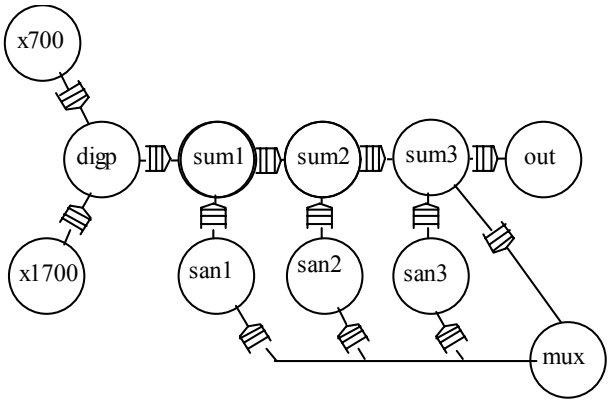


Fig. 7. **IIR2 filter Kahn process network model**

**Results of network observation.** During the experiment two FIR and two IIR filters Kahn process network modifications were implemented which perform filtering of two summed sinusoids. During modeling fixed amount of data was used to enable execution time measurement. Network nodes ended execution only when processing of all the data was finished.

Execution time variation according to channel length changes was observed during network operation. Channel lengths were changed from 50 to 600 packets. Change was performed manually. Results are presented in Fig. 8.

The quantity of packets transmitted trough channels was also observed during network operation.

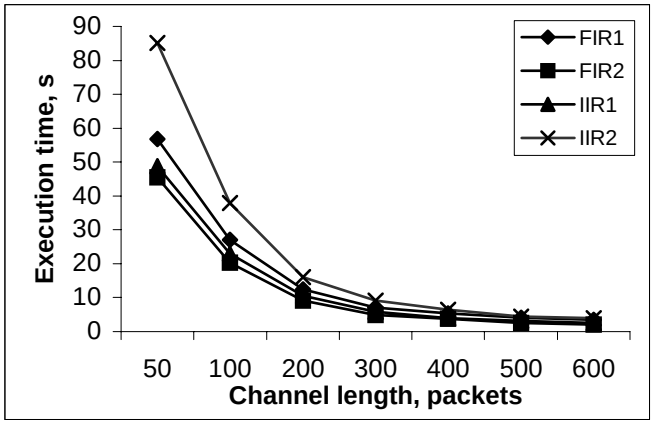


Fig. 8. **Execution time dependence on channel length**

Kahn process network models of the filters were executed and observed using personal computer, thus nodes were executed sequentially switching among them.

**Dynamic network parameters change.** Dynamic network parameters change was performed during network operation. The channel length was changed according to the number of packets transmitted. This number was estimated during network operation observation. The results are displayed in Fig. 9.

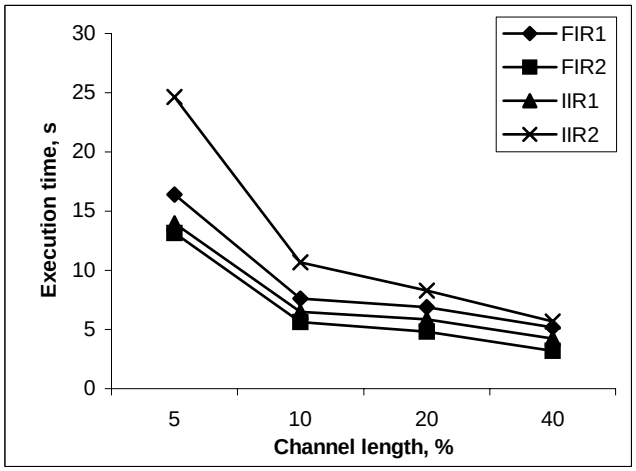


Fig 9. **Execution time dependence on channel length (when the particular percent of the whole length is allocated)**

While analyzing the results one can notice that network execution time reduces when channel length is increased. The best results outcome when channels get maximum amount of memory needed for computations.

If Kahn process network model would be implemented in parallel system, changing channel length would cause quite different results.

## Conclusion

In DSP systems design and implementation very important factor is project realization time. Various tools and methods are created to reduce this time. Kahn process network model, discussed in this paper, is one of such methods.

In analyzed digital signal processing systems modeling tools Kahn process network is widely used. But the implementation of the model usually uses classic model. Such use limits the possibilities of adapting Kahn process network in complicated real time systems.

In this paper Kahn process network operation observation parameters were determined and their changing strategies according to the systems needs were described.

System's Kahn process network can be modeled using personal computer. This helps in finding critical network operation points. Performance can be improved dynamically changing network parameters. Performing computations in sequential system the great number of network nodes negatively influences execution time. Channels lengths must be increased to reduce idle time of the network operation.

The results of our work can be used implementing real systems. Network operation observation results outline memory, computation and communication resources thus facilitating the hardware platform selection.

**References**

1. **Marven C., Ewers G.** A Simple Approach to Digital Signal Processing. – New York: Wiley-Interscience Publication. 1996.
2. **Kienhuis B., Rijpkema E., Deprettere E. F.** Campaan: Deriving Process Networks from Matlab for Embedded Signal Processing Architectures. 8th International Workshop on Hardware/Software Codesign. San Diego. 2000. 05. P.425-431.
3. **Davis J., Hylands C., Kienhuis B.** Ptolemy II: Heterogeneous Concurrent Modeling and Design in Java. Berkeley: The Regents of the University of California. 2001.
4. **Stevens R., Wan M., Laramie P.** Implementation of Process Networks in Java. 1997. www.ait.url.navy.mil/pgmt/Pnpaper.pdf
5. **Lavagno L.** Static Scheduling for Embedded Systems. University of Udine. 2001. http://tima.imag.fr/MPSOC/2001/lavagno.pdf
6. **Žvironas A., Kazanavičius E.** Partitioning of DSP tasks to Kahn network. KTU. Kaunas. Ultragarsas. 2002. Nr. 2(43).
7. **Lee E. A., Parks T. M.** Data flow process network. Proceeding of the IEEE. May 1995. Vol.83. No.5. P.773-799.

J. Čeponis, E. Kazanavičius, A. Mikuckas

**DSP sistemų analizė ir projektavimas naudojant Kahno procesų tinklus**

Reziumė

Darbe pateiktas Kahno tinklo procesų metodas ir jo taikymas DSP architektūrų analizei ir projektavimui, sprendžiant skaitmeninių signalų apdorojimo uždavinius. Pasiūlyto naujos Kahno tinklo modifikacijos ir patikrintas jų efektyvumas skaitmeninio filtravimo uždavinių klasei. Pasiūlytu metodu galima spręsti signalų apdorojimo uždavinius skirtingose DSP architektūrose.

Pateikta spaudai 2002 12 19

DOI: 10.5755/j01.u.45.4.8156