**Slide 1**

Dear colleagues, my name is Khilko Dmitry. I’m Senior Researcher at Federal Research Center “Informatics and Control”. Today I’m going to present our current results and problems in the area of Recurrent-Dynamic Approach research.

**Slide 2**

This approach combines several principles that together form new recurrent data-flow computational paradigm.

First principle is based on self-assembly phenomenon. This phenomenon is well-known in nanomaterial construction and biology.

The implementation of this phenomenon within approach is achieved by utilizing data-flow computational model that provides functionality to associate different computational branches in a native way (by comparison of tagged data). And a recurrency principle, that will be described further.

Second principle is based on self-sufficient data elements. Such elements combine data itself and the control information how to process it. Thereby a single self-sufficient (or self-sustained) data-flow is formed.

Third principle is recurrency. Basically, it’s mechanism that consists of two methods. The recurrent convolution is a method to compress and encode relatively large data-flow graph fragments into single ESD. And as a counterpart, recurrent involution is a method to decode and decompress ESD during computational process.

Combining self-assembly with recurrency provides ESD with functionality to define computational path by itself.

**Slide 3**

Based on this approach we have developed an architecture that is designed to be used in DSP domain.

In comparison with other architectures self-sufficient data requires less pipeline stages to be processed, because there is no need to address and read instruction and data separately.

And recurrency mechanism theoretically reduces data redundancy that is a well-known problem of data-flow architectures.

**Slide 4**

The self-assembly is a phenomenon that is being studied at the intersection of chemistry, physics and biology. Its essence lies in the spontaneous organization of randomly arranged molecules into ordered structures. Such organization is a result of specific local interactions between these molecules. As a result of such ordering, various nanostructures are formed, which are used to create nanomaterials with a controlled structure.

As slide shows there are two basic self-assembly strategies. Based on the research results of the scientist Makhiboroda A.V. we have chosen Bottom-up strategy.

**Slide 5**

Makhiboroda A. V. has shown that biology process of virus construction is a common example of self-assembly phenomenon. He has provided a bacteriophage T4 construction scenario that is shown on slide.

This process can be described as an algorithm that merges two separate chains of computation into one. As you can see each step of computation depends on previous in a recurrent way.

Also Makhiboroda has developed main concepts of self-assembly computational paradigm that we researched and adopted within recurrent-dynamic approach.

**Slide 6**

This slide explains main features of self-sufficient data in comparison with common data-flow model.

Within classic data-flow model computational process is organized as an unordered collection of packages with special format, demonstrated at top part of the slide. Each field in package (except flags) is a memory link. Package “firing” appears when selection device forms a pair of corresponding Tagged Operands by tags matching process. Therefore to compute encoded function read and deciphering stages are mandatory.

ESD also named Operand stores tagged data and all necessary control information (we call it functional fields). A package is formed by selection device by tags matching process. Therefore data-flow package consists only of two Operands. In this scheme read and deciphering stages become unnecessary. And main procedure stage extends by Functional transform operation that is performed simultaneously.

Function, Settings and Flags fields of Operand store control information that is encoded in a recurrent way and used during recurrent involution process in order to compute what operation should be performed on the next computational step. Within package both Operands contain that control information. This can be used to merge two independent chains of computation by special Discipline.

Initial sequence of Operands forms a “capsule” – main abstraction for executed program.

**Slide 7**

As it was said before recurrency mechanism consists of two methods: recurrent convolution and recurrent involution.

Both methods are based on a theory researched by mathematician Arnold V. I. He has shown that consequent application of the same functional operator to a finite sequence of zeroes and ones forms a predictable topology.

This slide demonstrates an example of Some functional operator, that when being configured by constant C, can form “rings” or tree-like topologies.

We call these topologies as Recurrent Chains.

Therefore recurrent convolution method includes searching for corresponding functional operator and computing proper initial value for Operand Functional Fields.

And recurrent involution method is a consequent configuration and application of functional operator.

**Slide 8**

We have synthesized an FPGA prototype of recurrent architecture and successfully tested it on DSP domain.

Within prototype recurrency mechanism was implemented with special block called Tag Transformer (TT). As shown on slide the TT block is located inside computing blocks at last pipeline stage.

However we couldn’t find general theoretical solution for setup and configuration of functional operator that could provide viable performance level.

Therefore we have developed Universal Tag Transformer device with code redundancy. This code redundancy allowed us to simplify configuration process and move it from run-time to compile-time in a cost of operand-word size increase.

**Slide 9**

This slide demonstrates the transformation graph of universal TT. It has tree-like structure and supports up to 3 steps of transformation of functional fields. Then all functional fields receive zero values and this zero-pole recurrent chain has to be merged with new recurrent chain in order to continue computations.

**Slide 10**

Despite that FPGA prototype has shown its workability and potential efficiency several problems within recurrent architecture were discovered.

The analysis of these problems allowed us to propose possible solutions that are listed on slide.

Short length of recurrent chains formed by universal TT greatly reduces efficiency of data compression and makes programming more complex.

The TT being located at last pipeline stage does not allow to use recurrency at full power. Each recurrent chain has to pass all pipeline stages. This can greatly reduce performance when merging recurrent chains that were created during computational process especially for algorithms with low degree of parallelism.

Unable to merge capsules is a major problem. At the current development stage capsules were designed such to communicate through special Control Processor. This creates delays between computation of sequential capsules. However capsules may be considered as a macro-recurrent chains at higher level of abstraction and could be merged by the same mechanism.

**Slide 11**

Further research and development of RDA and recurrent architecture will be performed in several directions.

First direction is an area of the functional operators topology that should help us to construct more effective tag transforming mechanism.

Second direction is an improvement of RDA key feature - recurrency mechanism. The most important goal is to achieve efficiency of recurrent chains merging mechanism.

Overall the solutions we can find will improve self-assembly computational paradigm and allow us to achieve goals listed in slide.