Design Models of Pipelined Units for Digital Signal Processing

Iryna Hahanova, Yaroslav Miroshnychenko, Irina Pobegenko, Oleksandr Savvutin

Abstract - **In this paper the architectural models of pipelined computing units with system-level description, those essentially decrease the design cycle for digital signal processing products, are offered. Practical realization of the filter, that confirms developed design flow effectiveness with software products Simulink (Mathlab) and Active HDL, Aldec Inc., is given.**

Keywords – **pipelined computing unit, digital signal processing, finite state machine, digital image processing.**

I. Introduction

Popularity of specialized digital products on the market in comparison with multipurpose computers is defined by: high performance in functionalities and operations execution, low power consumption and cost, parallel computational procedures on large size registers. Specialized devices, those are belong to DSP (Digital Signal Processing) group, are very popular due to their mass usage for tasks of image transmission in mobile communications by Wi-Fi, Wi-Max, UWB, RFID protocols. For the speed increase in DSP it is common to use a pipelined architecture, which allows to process big input data flows. Thus, it is necessary to have a tool, which could design a pipeline, could distribute its functions by time, could perform detailing of information transfer between cycles, could track the current pipeline state and could generate control signals.

Goal – essential decrease (by 30-70%) of DSP specialized pipelined units design cycle, by means of creation new architectural system-level model by given specification.

To reach the formulated goal it is necessary to solve the next tasks: 1) development of the architectural system-level description model for the pipelined computing unit; 2) development of the modified finite state machine model to control the process of pipelining; 3) creation of the model for the automatic generation of VHDL code for the control unit; 4) models testing and verification.

Article structure: 1) review of models, those are used to represent specialized pipelined systems for the digital data processing; 2) description of pipelined device macro-automaton model; 3) example of macro-automaton model of 2D pass digital image processing; 4) model of the process of control unit macro model transformation to synthesized RTL VHDL code; 5) assessment of the pipelined device design cycle time parameters.

II. Models of Pipelined Systems for DSP

Pipelined computing unit (PCU) is CU, that executes one or several periodic algorithms on the input data flow, where in the one cycle it is processed several data chunks, situated on the different process stages [1]. For the pipeline data processing there are exist several approaches, listed below.

1. Data-flow graph (DFG) proposes generalized model of data flow processing algorithm, where the graph nodes (actors) correspond to the particular computational processes, and the graph edges correspond to the communication buses between them. At that, an edge corresponds to FIFO buffer, which delays data flow transmission for defined number of receiver process starting cycles. Actor executes after the moment, when its inputs have information, which is ready to processing. Input data flow can be infinite, but there are two constraints in place: depth of any FIFO buffer is finite and the algorithm executes in such way, that buffers is never empty. Those constraints are called DFG model consistency constraints. When we have a consistent DFG, it is possible to make a formal transform of model to hardware or software implementation, that works in pipeline mode with minimized buffer memory capacity [2].

2. Synchronous data-flow graph (SDFG) is a simple and the most commonly used model of data processing algorithm. SDFG in contrast to DFG, has a constant data size, that is issued by the one from actors to its output and is received to its input during each run. Therefore, in SDFG every edge has two additional attributes: amount of data, that is transmitted to the edge by the actor-source and amount of data, that is received by input of the actor-receiver during each its run. Thanks to such constraints, it is significantly easier to perform SDFG model consistency check and to find its effective representation into hardware [1,2].

3. Signal graph or signal-flow graph is frequently used for signal processing algorithm definition. Transfer function of processing system expresses with Z-transform, so it in a formal way transforms into the CU structural model for signal processing or into the signal graph [1,3]. For example, transfer function

$$H(Z) = (1 + Z^{-1})/1 + Z^{-1} - Z^{-2})$$

transforms into signal graph on Fig.1. Operator Z^{-k} corresponds to delay for k iterations.

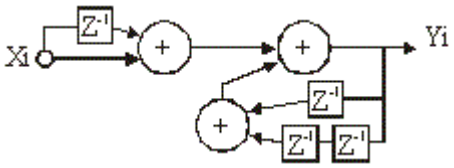

Fig.1 Signal graph

Described approaches allow define algorithm work, but do not give information about design components and how to control them upon process of functioning. They also do not allow creation of the device architecture in the form of operational and control automatons.

4. Models, those represent pipeline working process in multidimensional space [1]. They allow describe in detail pipeline work in space and perform its optimization. Though,

pipeline for image processing units can include tens or hundreds of thousands stages, that makes difficult detailing by cycles and makes application of such models ineffective.

5. For design of pipeline and its steering circuits it is inexpediently to use classical design approaches for Mealy and Moore automatons, because such automatons can include up to hundreds of stages. As a rule, control block for pipelined units bases on counters, those save the control automaton state. All output control signals are generated on basis of those counters' values.

Pipelined device model can be developed with help of hardware description languages, like VHDL or Verilog. Though, these languages are well-fit for description and future implementation of the model, but development of the device's architecture represents a certain challenge, that cannot be solved only in the language environment. At that level errors in algorithm implementation and pipeline construction will result in significant design time for model rebuilding and debugging.

Thus, it is desirable to have a tool, which could allow create pipelined device architecture at abstract level, perform its verification and only after that pass to model coding with HDL languages. For classical control automatons there are developed and used many graphical tools, those allow perform device creation by the means of state graph, from which HDL model is generated later. But existing design approaches for pipelined devices are intended to small length pipelines, which have one or few cycles for execution of block operations.

III. Macro-automaton model

For complex devices design with pipeline length of tens or hundreds synchrocycles, starting from system level and up to HDL model hardware implementation it is suggested to use Simulink software tool, which is part of Matlab software package.

Generalized model of pipelined device is shown on Fig. 2.

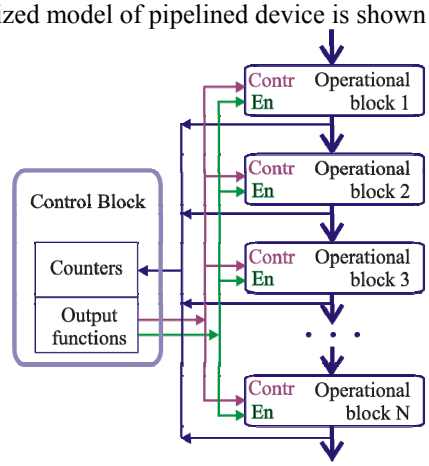

Fig.2 Pipelined schema, based on macro-automaton

To create such model it is necessary to perform the assignments: 1) universal control unit (UNU), shown on Fig. 3, should be described with the next parameters: number of counters, their capacity, range and counting direction; output control signals and their functions, those are formed by unit; input signals, those are connected to counters' inputs – 'clock enable' and 'set/reset'; 2) operational units should have defined inputs, output functions, number of cycles, needed to form values on each output; 3) control signals, generated by UNU, should be connected to controlling inputs (CI) and 'enable' inputs (EnI) of each operational unit.

In case when upper level model operational units are controlled by the clock edge, in future detailing of such project they can be changed to their HDL models and can be processed with simulation software Active-HDL or Modelsim.

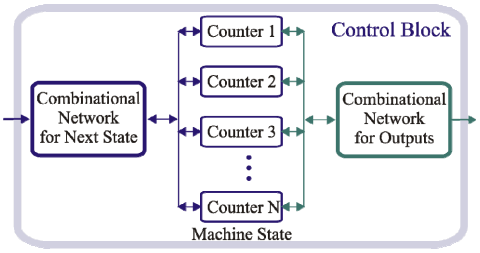

Fig.3 Macro-automaton

Thus, complex pipelined digital devices design using system-level models in Simulink will contain the next stages: 1) system model creation in Simulink environment, where for each operational unit will be defined functions, control signals, number of cycles to generate an output value. Also, the simplified models of ready IP cores can be used; 2) control unit creation, where we define counters number and their parameters, output functions, control signals for counters; 3) device's system model debugging, creation of test sequences and patterns; 4) development of models for each unit in HDL, that can be performed simultaneously for different units. HDL models debugging for each unit. Possible automated generation of control unit HDL model; 5) testbench generation for HDL model.

Simulink software has an interface to HDL simulation software: Active-HDL and Modelsim. Therefore, in offered schema we can add HDL units and Simulink modules simulation stage. Though, in that case system schema should be detailed down to clock edge control.

IV. Macro-automaton model of 2D-pass for digital image processing

For example we describe 2D-pass for halftone image processing with block diagram on Fig.4.

Kernel coefficients, those are described by 3x3 matrix, define the result of image filtering process[4]:

$$M = \begin{bmatrix} a & b & a \\ d & e & d \\ g & h & g \end{bmatrix}.$$

In that case, every pixel's value can be described with expression:

$$\begin{aligned} P_{ij} &= a \times x_{i-1,j-1} + b \times x_{i,j-1} + \\ &+ c \times x_{i+1,j-1} + d \times x_{i-1,j} + \\ &+ e \times x_{i,j} + f \times x_{i+1,j} + g \times x_{i-1,j+1} + \\ &+ h \times x_{i,j+1} + k \times x_{i+1,j+1} + weight, \end{aligned} \quad (1)$$

where P(i,j) – current pixel's coordinate.

For example, for kernel of the pass

$$A = \begin{bmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{bmatrix}$$

each pixel values are calculated by the next expression:

$$P_{ij} = \frac{1}{16}(x_{i-1,j-1} + x_{i,j-1} + x_{i+1,j-1} + x_{i-1,j} + x_{ij} + x_{i+1,j} + x_{i-1,j+1} + x_{i,j+1} + x_{i+1,j+1}).$$

Pass scheme includes two operational units (Memory_buffer and DSP_arith_block) and one control unit Macro_control.

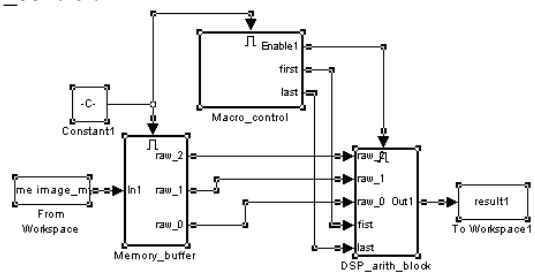

Fig. 4 Block diagram of image pass in Simulink

Memory block (Memory buffer), shown on Fig. 5, takes an input sequence, that in each cycle delivers a new value for input signal Di and generates simultaneously three values from three different rows: raw_2, raw_1 and raw_0.

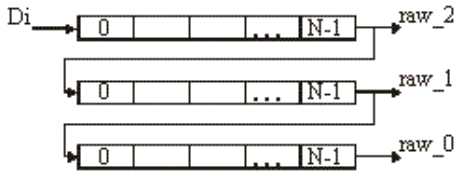

Fig.5 Shift registers unit architecture

That data proceeds to the arithmetical unit DSP_arith_block. Delay between feed of the fist value on the unit's input and the first generated output sequence equals 2N, where N is an image row's length. Thus, that value for an image width of 250 pixels will equal 500 cycles. Fig. 6 describes memory schema in Memory_buffer in Simulink for the row of 6 elements.

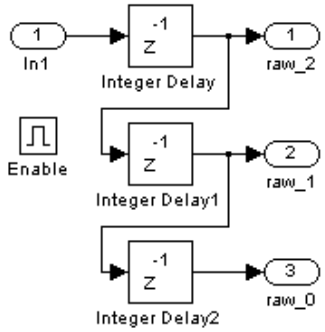

Fig. 6 Memory_buffer unit model in Simulink

To implement row shift registers (Fig. 4) it were used Integer Delay elements.

Thus, Memory_buffer unit has data input and read enable input, and three outputs, which values are copies of the input signal, those are delayed by N, 2N and 3N cycles:

Output	Function
raw_0	$Di*z^{-N}$
raw_1	$Di*z^{-2N}$
raw_2	$Di*z^{-3N}$

Arithmetic unit implements calculation of pixel's value, based of 9 values of the source image (1). At the same time on the unit's input is fed with values from each of three rows. Thus, taking into account delays, Eq. (1) can be written in the next way:

$$P_{ij} = a \cdot raw\_0 \cdot z^{0} + d \cdot raw\_1 \cdot z^{0} + g \cdot raw\_2 \cdot z^{0} + b \cdot raw\_0 \cdot z^{-1} + e \cdot raw\_1 \cdot z^{-1} + h \cdot raw\_2 \cdot z^{-1} + c \cdot raw\_0 \cdot z^{-2} + f \cdot raw\_1 \cdot z^{-2} + k \cdot raw\_2 \cdot z^{-2}.$$

Delay between feed of the input value and reaction on the output equals to the one cycle. Block diagram of DSP_arith_block (arithmetic unit) is shown on Fig. 7. It has three data inputs, one output, clock enable input (Enable). Two controlling inputs – First and Last are used during the first and the last rows' coefficients calculation, because these rows do not have enough input information. In that case value of the first or the last row is duplicated.

Fig. 8 shows UCU for image filtering process controlling.

According to proposed approach, to describe a such unit it is needed to define:

1) number of counters, their capacity, range and counters direction. Device includes two counters – count1 and counter3. The first one counts number of elements in a row, which is defined by a constant, and the last one counts a number of processed rows;

2) output control signals and and their functions, those are formed by the unit. There three control signals are formed: Enable, First and Last, those are pass to arithmetic unit. Their functions are:

$$First = \begin{cases} 1, & counter3 - 2; \\ 2, & else. \end{cases}$$

$$Last = \begin{cases} 1, & counter3 = raw + 1; \\ 2, & else. \end{cases}$$

$$Enable = \begin{cases} 1, & (counter3 > 1) \,\&\, (counter3 <= raw + 1); \\ 1, & (counter3 = raw + 2) \,\&\, (count1 = 1); \\ 0, & else. \end{cases}$$

3) counters' control signals: enable (clock enable) and set/reset. Unit should have a general reset to set the device into the initial state and 'enable' input, that is external for the whole device, and which sets, when the data input is fed with a new value.

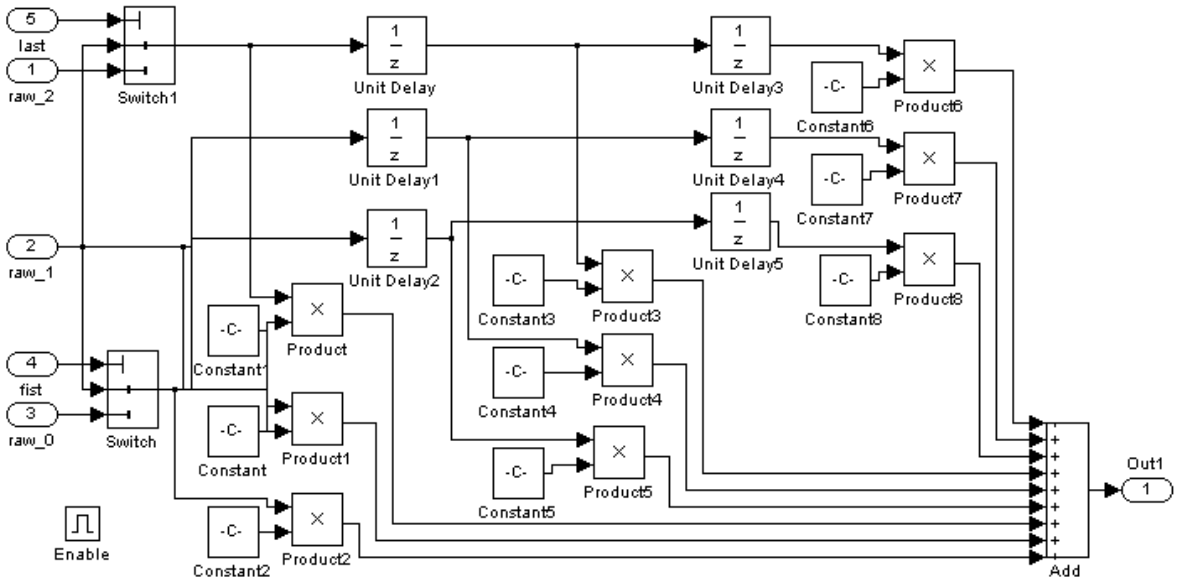

Fig. 7 DSP_arith_block unit model in Simulink

For the testing process in Matlab it was created script, that simulates work of the pass and generates standard output sequences. Then, the same test was fed to the device model in Simulink, and the test results were compared to the standard ones.

V. Transformation Process of Control Unit Macromodel into Synthesized RTL VHDL Code.

All control unit parameters, those are shown on Fig. 8, presented in Table 1, so it is easy to create a VHDL model, based on it.

Constant values raw_image_size and column_image_size are transformed to generics in VHDL. Counters capacity is calculated, basing on raw_image_size and column_image_size values [5].

```
generic ( raw_image_size:integer:=256;
column_image_size:integer:=256;
count_size: integer:=8);
```

Unit contain ports: clock enable input ‘Enable1’, clock input ‘clk’ and reset input. Output ports are defined by control unit output functions.

```
port( Enable : in STD_LOGIC;
Clk, reset : in STD_LOGIC;
First : out STD_LOGIC;
Last : out STD_LOGIC;
Enable1 : out STD_LOGIC );
```

Every counter is implemented with a standalone process operator [6]. For example, process for ‘count1’ counter looks like that:

```
cnt1: process(reset, enable, clk)
begin
if reset='1' then
-- na.in n.ao.eea a ia.aeuiia ninoiyiea
count1 <= (others=>'0');
elsif clk='1'and clk'event then
if enable='1' then
-- i.iaa.ea aa.oiae a.aieou n.aoa
if CONV_INTEGER(count1) = raw_image_size +1
then count1 <= "000000001";
else count1 <= count1 + '1';
end if;
end if;
end if;
end process;
```

TABLE 1

CONTROL UNIT OF THE PASS DESCRIPTION

Constants			
raw_image_size	Number of rows		
column_image_size	Number of columns		
Counters			
Number	2		
count1			
Initial state	0		
Range	1	raw_image_size+1	
Control signals	enable	external input	comb.
count2			
Initial state	0		
Range	0	max	
Control signals	enable	count1=raw_image_size+1	reg.
Output functions			
first	if (count2==2) first=1 else first=0		comb.
last	if count2==(column_image_size+1) last=1 else last=0		comb.
enable1	if(count2>1)&(count2<=column_image_size+1) enable1=1 elseif		comb.

	(count2==(column_image_size+2))&(cou nt1==1) enable1=1 else y=0	

Process and control signal (enable_cnt2) generation for the second counter:

```
cnt2: process(reset, enable_cnt2, clk, count1)
begin
if reset='1' then
count2 <= (others=>'0');
elsif clk='1'and clk'event then
if enable_cnt2='1' then
if CONV_INTEGER(count1)=raw_image_size+1 then
count2 <= count2 +1;
end if;
end if;
end if;
end process;
en_cnt2: process(reset, clk)
begin
if reset='1' then
enable_cnt2 <= '0';
elsif clk='1'and clk'event then if
CONV_INTEGER(count1)=raw_image_size+1
then
enable_cnt2<='1';
end if;
end if;
end process;
```

Combinational output functions of UCU could be implemented in VHDL with help of the concurrent case statements:

```
first <='1' when count2=x"02"
else '0';
last <='1' when
CONV_INTEGER (count2)=(column_image_size+1)
else '0';
enable1 <= '1' when (CONV_INTEGER(count2)>1)and
(CONV_INTEGER(count2)<=column_image_size+1)
else '1' when
(CONV_INTEGER(count2)=(column_image_size+2)) and
(count1=x"01")
else '0';
```

In case of register unit outputs, maximum allowable work frequency of the device will be increased. Thanks to the fact, that every construction in Table 1, which describes particular characteristic of CU, has corresponding construction in VHDL, we can implement automated generation of the VHDL model.

VI. Conclusion

Scientific novelty is in offered architectural system-level models of pipelined devices for digital data processing, those make automatic VHDL-code generation by the device specification considerably easier.

Practical significance is in essential decrease (by 30-70%) of design cycle of digital signal processing pipelined device, by means of preliminary development of its architectural system-level model by the given specification.

Advantages of the offered hierarchical design models:

1) device's system description clearness in the architectural form and the possibility of simulation of its behavior up to RTL model creation in HDL;

2) technological effectiveness in VHDL code creation, simplicity of testing, diagnosis and fixing of data transmission or control signals generation errors, that is essentially simplifies design process and reduces development time for pipelined computing units.

References

[1] A. M. Sergienko. VHDL for computing units design. K.: PP Korneychuk. 2003. 208 p.

[2] E. A. Lee, D. G. Messerschmitt. Calculations with synchronous data flows. TIIER. 1987. Book 75. №9. pp.107-119

[3] S. K. Rao, T. Kaylat. Regular iterative algorithms and their implementation on processing matrices. TIIER. 1988. Book 76. №3. pp.58-69.

[4] Steven W. Smith. The Scientist and Engineer's
Guide to Digital Signal Processing. California Technical Publishing. San Diego. California. 1997. 645 p.

[5] Ashenden, Peter J. The designer's guide to VHDL. San
Francisco: Morgan Kaufmann Publishers. 1996. 688 p.

[6] Bhasker, J. A VHDL Synthesis Primer. Allentown: Star Galaxy Publishing. 1998. 296 p.

Reviewer: ScD, professor G. F. Krivoulya.

Iryna Hahanova – person, working for doctor's degree, DAD Department, Kharkiv National University of Radio Electronics (KNURE), 14, Lenina ave, Kharkiv, 61166, UKRAINE
E-mail: hahanova@mail.ru
Yaroslav Miroshnychenko – engineer, DAD Department, KNURE.
E-mail: miroshnychenko@kture.kharkov.ua
Iryna Pobegenko – PhD student, DAD Department, KNURE.
E-mail: irina_pob@ukr.net
Oleksandr Savvutin – 4th course student, KNURE.
E-mail: alex-svx@mail.ru