MODELLING AND ANALYSIS OF AN ARITHMETIC-LOGIC UNIT WITH INTEGRATED DEVELOPMENT ENVIRONMENT WEBPACK

Valentina Stoyanova Kukenska, Ivan Simeonov Simeonov

Computer Systems and Technologies, TU - Gabrovo, 4 Hadji Dimitar St., 5300 Gabrovo, Bulgaria, tel.:+359 66 223 456, e-mail: vally@tugab.bg, tel.:+359 66 223 479, e-mail: isim@tugab.bg

One of the methods for designing digital devices is through the use of programmable logics. What is necessary for the separate stages of designing is both constructing and using different descriptions. For this purpose, languages for hardware description like VHDL (Very Hardware Description Language), ABEL, Verilog and others are used. With their help we construct structural and behavioral descriptions of the objects designed. These sets of descriptions are then used for drawing up the models of devices we're going to be in need of for their designing.

This paper presents developed VHDL descriptions of arithmetic-logistic units. We have both structural and behavioral models described, the latter being simulated in WebPack. The paper also features final results and analyses.

The paper is an element of a database of project decisions for digital schemes and units on the basis of programmable logics, currently developed by our team.

Keywords: modeling, analysis, ALU, VHDL, WebPack

1. INTRODUCTION

One of the methods for designing digital schemes and devices is through the use of programmable logics. To describe the objects of design one should use either graphical or symbol editor. It is convenient to use graphical editor in case of provided principle scheme of the device.

What is necessary for the separate stages of designing is both constructing and using different descriptions. For this purpose, languages for hardware description like VHDL (Very Hardware Description Language), ABEL, Verilog and others are used. With their help we construct structural and behavioral descriptions of the objects designed. These sets of descriptions are then used for drawing up the models of devices we're going to be in need of for their designing.

This paper presents developed VHDL descriptions of arithmetic-logistic units. We have both structural and behavioral models described, the latter being simulated in WebPack. The paper also features final results and analyses.

The paper is an element of a database of project decisions for digital schemes and units on the basis of programmable logics, currently developed by our team.

2. HOW IT WORKS ALU

The Arithmetic-Logistic Unit (ALU) performs arithmetic (addition, subtraction, multiplication and division in two, changing signs) and/or logical operations

(displacement, rotation, bit conjunctions, disjunctions and others). This unit has two n-bit inputs and a single n-bit output, as well as with inputs and outputs for each bit for carrying, plus an extra input allowing the option of the operations demanded (Fig.1).

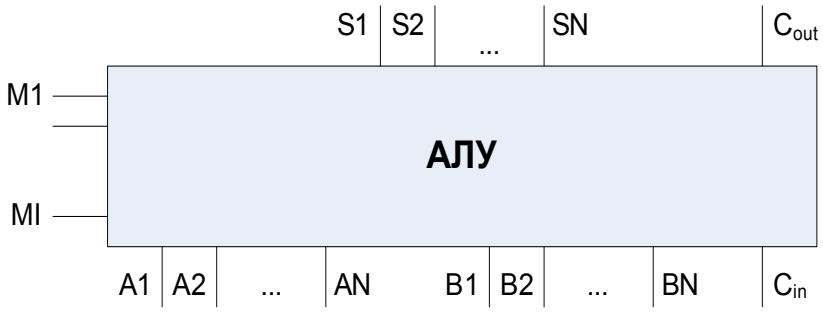


Fig. 1. Block scheme of an arithmetic-logical unit (ALU)

Our main purposes here are: getting to know the way ALU works, drawing up a simple behavioral description of the object; building up of behavioral specification by representing the interim calculations as local signals and implementation of a structural description of the ALU's VHDL model.

Fig. 2. presents the principle scheme of an ALU, visualizing four inputs, two transferring bits, a bit for choosing regime and two bits for the output.

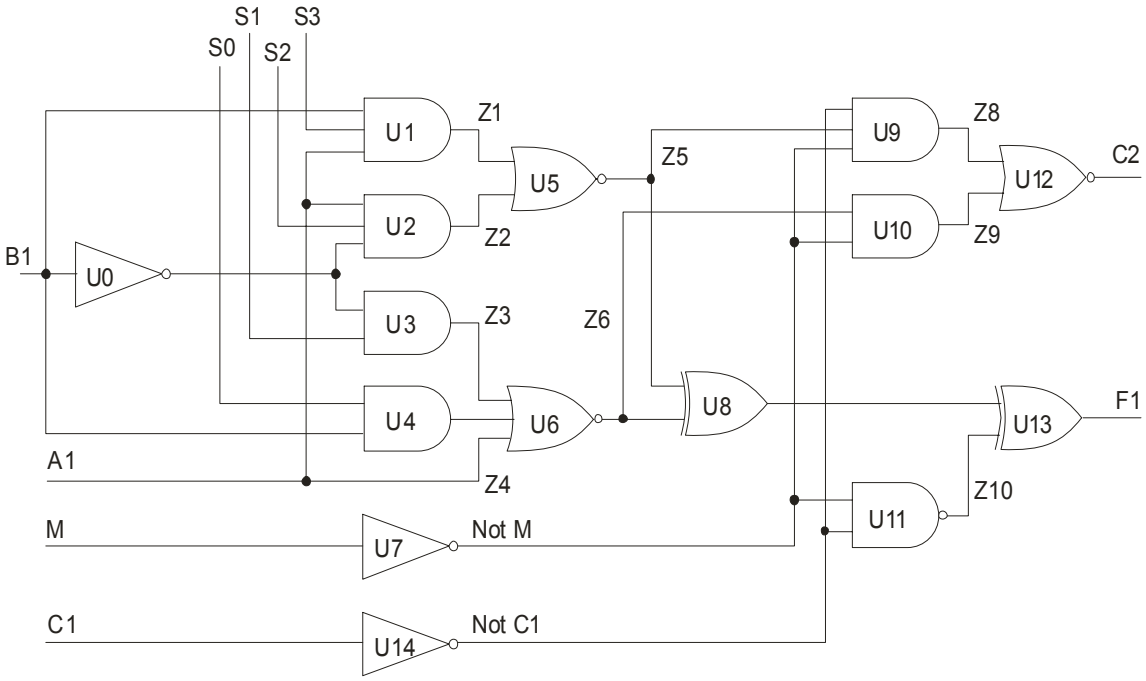


Fig. 2. Principle scheme of an arithmetic-logical unit

ALU has four inputs: four option lines S3, S2, S1 and S0; an inverted transferring bit C1; a bit for choosing regime M; two bits for operandi A1 and B1. The outputs are either the arithmetic or the logical outcome F1 and the final transferring bit C2.

In case M=T the system performs logical functions only, keeping in mind that when M=T, C2 always equals '1'.

if M = 1

"0000"	not A1
"0011"	'0'
"0101"	not B1
"0110"	A1 xor B1
"1010"	B1
"1011"	A1 and B1
"1100"	'1'
"1110"	A1 or B1
"1111"	A1

When M='0' the system performs arithmetical functions only. For example:

if M = 0

"0000"	A1 + C1
"0011"	not C1
"0110"	A1 + (1's compliment B1) + C1
"1001"	A1 + B1 + C1
"1111"	A1 + 1 + C1

```
entity ALU_stage is
        port (S3, S2, S1, S0, A1,
B1, C1, M : Bit ; C2, F1 : out Bit);
end ALU_stage;
```

3. VHDL MODELING

Computer designing represents a sequence of phases for solving specific problems in creating a reliable model for object examination**.**

The technological means for computer design (modeling) encompasses both language and operational regimes for realization of models and model experiments, as their data setup provides for a simple and efficient designing, as well as for an easy description of structures and connections within the object. They are convenient resources for formalization and description of process dynamics.

The working out of a computer model is represented as a consolidated sequence of abstract steps specifying the methodological schemes of computer designing. The very elaboration goes through the following main stages: formulating a conceptual model, designing a computer model, specifying the model adequacy, realizing model experiments and drawing up assessments and conclusions.

What follows after a while is a behavioral description of an arithmetic-logical unit (ALU) representing interim calculations as local signals.

```
        architecture Pure_behavior of ALU_stage is
        begin
C2 <=
    ( ( not C1 ) and ( ( B1 and S3 and A1 ) nor ( A1 and S2 and
        ( not B1 ) ) ) and ( not M ) )
        nor ( ( not ( ( ( not B1 ) and S1 ) or ( S0 and B1 ) or A1 ) )
        and ( not M ) ) after 20 ns;
F1 <=
    ( ( B1 and S3 and A1) nor ( A1 and S2 and ( not B1 ) ) ) xor
    ( not ( ( ( not B1) and S1) or ( S0 and B1) or A1 ) ) xor
```

```
        ( (not C1) nand (not M) ) after 20 ns;
    end Pure_behavior;
        architecture Behavior of ALU_stage is
        signal NotC1, NotB1, NotM : Bit ;
            signal Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8, Z9, Z10  : Bit ;
            constant Delay : Time := 5 ns ;
    begin
            U0  : NotB1 <=  not B1 after Delay;
            U1  : Z1      <=  B1 and S3 and A1 after Delay;
            U2  : Z2      <=  A1 and S2 and NotB1 after Delay;
            U3  : Z3      <=  NotB1 and S1 after Delay;
            U4  : Z4      <=  S0 and B1 after Delay;
            U5 :  Z5      <=  Z1 nor Z2 after Delay;
            U6 :  Z6      <=  not ( Z3 or Z4 or A1 );
            U7  : NotM  <=  not M after Delay;
            U8  : Z7      <=  Z5 xor Z6 after Delay;
            U9  : Z8       <=  NotC1 and Z5 and NotM after Delay;
            U10 : Z9       <=  Z6 and NotM after Delay;
            U11 : Z10     <=  NotC1 nand NotM after Delay;
            U12 : C2       <=  Z8 nor Z9 after Delay;
            U13 : F1        <=  Z7 xor Z10 after Delay;
            U14 : NotC1 <=  not C1 after Delay;
        end Behavior;
```

And for the behavioral description above, the following VHDL structural model is put forward.

Each of the specified competitive signals using built-in operators is replaced by a sign that instantiates given components. This sign, on its turn, creates a gate-level primitive that acts like the inbuilt operator.

```
architecture Structure of ALU_stage is
signal NotC1, NotB1, NotM : Bit;
signal Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8,
Z9, Z10 : Bit;
component And2
port( I1, I2 : Bit; O1 : out Bit );
end component;
component And3
port( I1, I2, I3 : Bit; O1 : out Bit );
end component;
component Nor2
port( I1, I2, I3 : Bit; O1 : out Bit );
end component;
component Nor3
port( I1, I2, I3 : Bit; O1 : out Bit );
end component;
component Inverter
port( I1 : Bit; O1 : out Bit );
end component;
component Xor2
port( I1, I2 : Bit; O1 : out Bit );
end component;
component Nand2
port( I1, I2 : Bit; O1 : out Bit );
end component;
begin
  U0  : Inverter port map ( B1, NotB1 );
U1  : And3 port map (B1, S3, A1, Z1 );
```

```
U2  : And3 port map ( A1, S2, NotB1,
Z2 );
U3  : And2 port map ( NotB1, S1, Z3 );
U4  : And2 port map (S0, B1, Z4 );
U5  : Nor2 port map ( Z1, Z2, Z5 );
U6  : Nor3 port map ( Z3, Z4, A1, Z6 );
U7  : Inverter port map ( M, NotM );
U8  : Xor2 port map ( Z5, Z6, Z7 );
U9  : And3 port map ( NotC1, Z5,
NotM, Z8 );
U10 : And2 port map ( Z6, NotM, Z10
);
U11 : Nand2 port map ( NotC1, NotM,
Z10 );
U12 : Nor2 port map ( Z8, Z9, C2 );
U13 : Xor2 port map ( Z7, Z10, F1 );
U14 : Inverter port map ( C1, NotC1 );
end Structure ;
```

4. TESTING THE VHDL MODELS OF ALU

The simulation of VHDL models of ALU has been done under WebPACK, as it has been carried out, following the steps below:

- introducing the description of the designed object, using the high-level description language VHDL;
- Transforming the language into a list of connections (netlist). The very netlist is a description of both logical schemes in designing and the way they interrelate;
- Representation of logical schemes and inside connections of functional blocks, which can be decomposed to their composite parts in macro cells;
- Charging a stream of bits. After the end of charging, operations specified by the HDL code are performed.

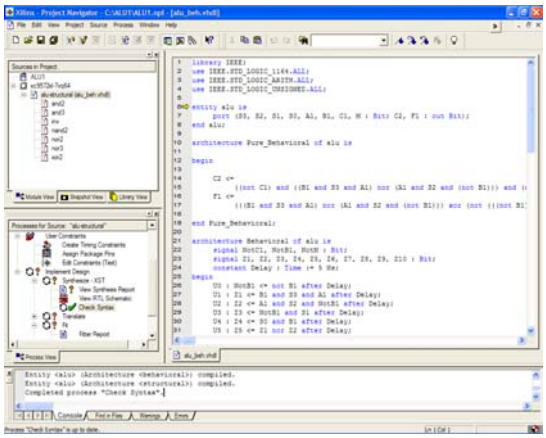
This action has been made quite easier due to the fact that Xilinx WebPack disposes of an HDL editor, generator of logics and a generator provided with programs.

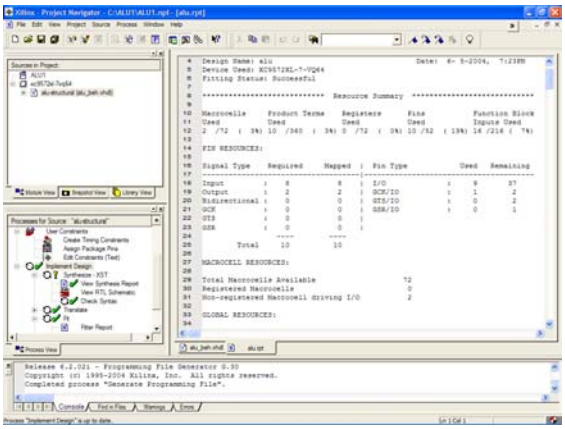
After filling in the fields in the New Protect window, we go for the next step, where the actual designing of an ALU is carried out. We move on by choosing the New Source button.

Both inputs and outputs of the scheme for ALU are declared in the same window. A set of operations is performed, leading to the creation of a 7-bit S bus, providing the object with information. We should then choose the Direction field for the S bus, a menu appears where we should arrange output signals for the bus.

The structure of the VHDL model of ALU is represented below, as from the first to the fourth row there is an establishment of links containing useful definitions of a project description. ALU inputs and outputs are declared between rows 6 and 9, while logic operations in the architectural section are found in the range between row 13 and row 16.

After the ALU logical scheme has been created, it should be adapted to CPLD. In order to do so, we chose the alu1 object in the starting window and then double-click on Implement Design. This step is shown below.





The Translation process transforms the initial netlist into Xilinx, thus creating a commentary on all actions that emerged during the process of designing. Elements of the scheme found in the unit and the netlist get connected through the process of adaptation.

To mark off input from output pin, we should use the Edit Constraints function. The Chip Viewer window displays the list of signals that are divided into input and output pin.

The Functional Blocks (FB) contains macrocells used within the process of elaboration. Here we see the input signals coming trough the chips connected to all functional blocks (FB).

The Resource Summary shows that just 2 out of all 72 macrocells present have been used by the object in XC9572XL CPLD. Ten input-output chips have been chosen, eight of them used for input chips and three for outcome ones.

5. CONCLUSION

The VHDL models of ALU shown here are elements of a library, elaborated by the authors, featuring project variants for digital schemes and devices on the basis of programmable logics.

The elaborated models, as well as the very methodic are used for designing more complex devices. The decomposition of the latter results in obtaining library modules.

Designing decisions are use also used in the training process of students from 'Computer systems and technologies', in their disciplines "Digital schematics", "Automated design and systems for automated design", "Designing with programmable logics" and "Designing of computer systems".

6. REFERENCES

[1]. Armstrong J., Chip-level modeling with VHDL, 1991.

[2]. Xilinx Design Reuse Methodology., System-on-a-chip designs reuse solutions., Reuse Methodology Manual, 1998.

[3]. Wayne W., Modern VLSI Design, Prentice Hall, 1994.