International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:9, 2008

Asynchronous Microcontroller Simulation Model in VHDL

M. Kovac

***Abstract*—**This article describes design of the 8-bit asynchronous microcontroller simulation model in VHDL. The model is created in ISE Foundation design tool and simulated in Modelsim tool. This model is a simple application example of asynchronous systems designed in synchronous design tools. The design process of creating asynchronous system with 4-phase bundled-data protocol and with matching delays is described in the article. The model is described in gate-level abstraction.

The simulation waveform of the functional construction is the result of this article. Described construction covers only the simulation model. The next step would be creating synthesizable model to FPGA.

***Keywords*—**Asynchronous, Microcontroller, VHDL, FPGA.

I. Introduction

THIS article has arisen at the beginning of research of the asynchronous systems properties. The goal of the research is the simulation of asynchronous systems by using ordinary design tools (VHDL) and ordinary target platforms (FPGA). At the beginning it started with asynchronous registers and combinational blocks. Then it leads to more complex structures like the microcontroller.

At first a simple controller without external ports and storage RAM for data was designed. Features of the simple controller will be explored and compared with classic synchronous microcontrollers by simulating and following implementation of the asynchronous circuit.

The next, introductory chapter will explain the asynchronous system theory. The basic use of bundled-data protocol and his VHDL code is shown in this chapter. The third chapter contains the microcontroller design itself. Following chapter refers to results of the simulation. The article ends with conclusion.

II. Asynchronous Theory

Synchronous logical systems are used in most cases. But the benefits of asynchronous circuits are left unnoticed. Among the benefits are the speed, modularity, low consumption and less electromagnetic emission. No global control, such as clock in synchronous circuits, is here. Every module has its own control and it communicates with each other neighboring module. Only operating modules are active. Operation speed depends only on the physical properties of semiconductor. No worst-case like in synchronous system is estimated here.

On the other side asynchronous systems have also disadvantages, namely the design complexity and lack of automated design tools.

Asynchronous systems don't have any global control, but they communicate locally with each other. This communication is called handshaking. There are two basic handshaking protocols: bundled-data and dual-rail. In the bundled-data protocol the data are packed together. Communication control is performed by request and acknowledge signals. The signal timing is shown in Fig. 3. This channel is called push channel, because the sender is active. When the receiver is active, then the channel is called pull channel [2].

Two wires per one data bit are used in the dual-rail protocol. The request signal is encoded with data in these two wires. Acknowledge signal is left alone.

There are two options in these two protocols: 4-phase or 2-phase handshaking. In the 4-phase handshaking information is encoded in signal level. The disadvantage is superfluous return to zero transitions. In the 2-phase handshaking information is encoded in signal transition. This manner is efficient but more complex. The rest of the article will reflect only the 4-phase bundled-data protocol.

Asynchronous module block with basic communication ports is shown in Fig. 1.

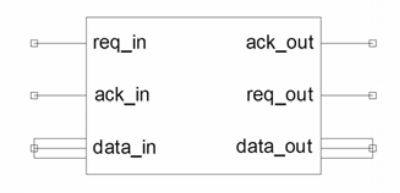

Fig. 1 Asynchronous module

Fig. 2 shows the VHDL code of this asynchronous module. This code is the headstone for building blocks of asynchronous microcontroller.

M. Kovac is with the Radioelectronics Department, Faculty of Electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic (e-mail: xkovac03@stud.feec.vutbr.cz).

```
process begin
loop
wait until req_in='1';
--waiting for request from predecessor
ack_in<='1' after delay;
--received data acknowledge
data_out<=data_in after delay;
--data transform
req_out<='1' after delay;
--sending request for successor
wait until ack_out ='1';
--waiting for acknowledge from successor
ack_in<='0' after delay;
--assert acknowledge low for predecessor (ready
for new data)
req_out<='0' after delay;
--assert request low for sucessor
end loop;
end process;
```

Fig. 2 VHDL code of handshaking

Fig. 3 shows 4-phase bundled-data communication between two modules.

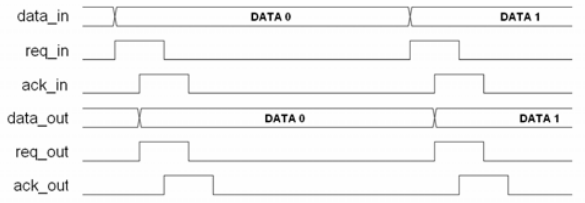

Fig. 3 Bundled-data handshaking protocol

The sender issues data and sets request high, the receiver absorbs the data and sets acknowledge high, the sender responds by taking request low (at which point data is no longer guaranteed to be valid) and the receiver acknowledges this by taking acknowledge low. At this point the sender may initiate the next communication cycle [1].

III. Microcontroller Design

The microcontroller consists of a control and a data processing parts. The control part is composed of program counter, program memory and instruction decoder. The Arithmetical-Logic unit (ALU) is the core of data processing part. Further parts are input multiplexer, Accumulator register and register file.

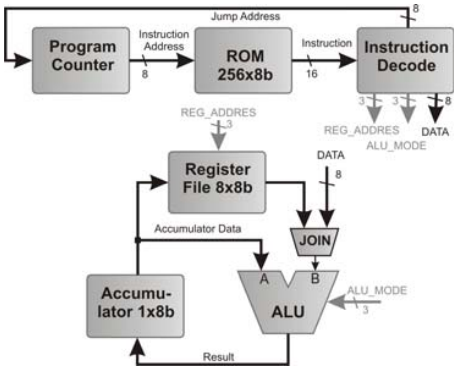

Fig. 4 Microcontroller block diagram

The program counter generates addresses for program memory. The program counter increments 1 after instruction execution. The program memory is a 256x13b ROM type memory, which is addressed by 8 bit address from program counter. Instruction word length is 13 bits. According to instruction the instruction decoder generates control signals: ALU mode, register address to register file, choice signal C and the data to data bus. The instruction is composed of following bits:

Fig. 5 Instruction word

where C is a control choice signal meaning immediate or register addressing. The word R determines one from eight registers in register file. The word A means the ALU operation [3].

There are four instruction types in figure 5: a.) ALU instruction with immediate value, b.) register addressing to ALU, c.) Accumulator value movement to the selected register in register file, d.) register loading by immediate value.

According to the most significant instruction bit the instruction decoder sends token either to ALU or to register file. For an example if ALU receives token, then the *C* control signal set the addressing mode. If addressing mode is register, then ALU sends request to register file. Register file sends data from specified register to data bus. After acknowledge from register the ALU must feed data into second data input. Then ALU speaks to Accumulator (send request). It returns the data value on A input port with acknowledgement. When the both data input ports of ALU are filled with data, ALU will execute a specified operation and result will be on output *q*. ALU then sends request to write to the Accumulator. Accumulator stores the result and returns acknowledgement. ALU sends back the acknowledgement to program counter after successful execution of the instruction. Then the program counter increments. In case of instruction, which sends token to register file, there are two options: either the value from dedicated register is written to gate B of ALU or the accumulator value is written to the dedicated register in register file. The Disadvantages of the asynchronous implementation are more complex design and handshake interconnections.

TABLE I
INSTRUCTION SET

ADD A, IMM	00111 IIIIIIII
ADD A, R	01111 XXXXXRRR
SUB A, IMM	00101 IIIIIIII
SUB A, R	01101 XXXXXRRR
AND A, IMM	00000 IIIIIIII
AND A, R	01000 XXXXXRRR
OR A, IMM	00001 IIIIIIII
OR A, R	01001 XXXXXRRR
XOR A, IMM	00010 IIIIIIII
XOR A, R	01010 XXXXXRRR
MOV A, IMM	00011 IIIIIIII
MOV R, A	11RRR XXXXXXXX
MOV R, IMM	10RRR IIIIIIII

The construction is designed by the top-bottom method. Firstly the basic blocks are defined as entities with data ports only. Then the logical structure of token transfer is created. Necessary handshaking ports (req, ack) are projected on the basis of this token transfer structure, as it is shown in the example in the previous chapter. After the detailed entity definition and considering token transfer the architecture body (block function) in each block will be completed. The simple example is a sample design in second chapter [4].

The under mentioned VHDL code is an example of the Accumulator architecture body. There are two concurrent processes, because signals *req_r* and *req_w* may occur independently from each other.

```
process begin
loop
wait until req_r='1';
dout<=reg after delay;
ack_r<='1' after delay;
wait until req_r='0';
ack_r<='0' after delay;
dout<=(others=>'Z') after delay;
end loop;
end process;
process begin
loop
wait until req_w='1';
reg <= din after delay;
ack_w<='1' after delay;
wait until req_w='0';
ack_w<='0' after delay;
end loop;
end process;
```

Fig. 6 Accumulator VHDL code of the architecture body

IV. SIMULATION RESULTS

The designed microcontroller parts in VHDL were composed to one ensemble in top-level schematic tool showed in Fig. 7.

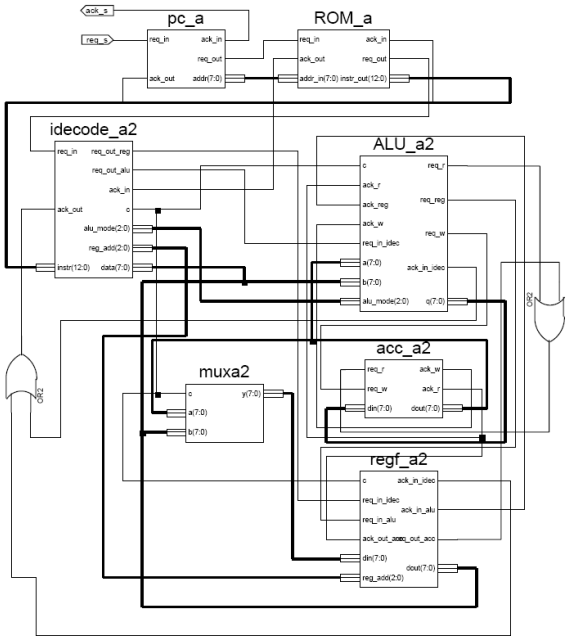

Fig. 7 Microcontroller design in ISE Foundation top- level schematic tool

The OR gates join the two mutually-exclusive output signals to one input. The 1 ns transport delay was added to all assignments in behavioral simulation to show the relationships between handshaking signals.

The following simulation in Fig. 8 shows instruction execution in memory address 01h: OR Acc, 08h. The Instruction decoder sends request to ALU. Simultaneously the immediate data are reached the port B with signal ALU mode. Then ALU sends request *req_r* in order that the Accumulator sends data to port A. Data was sent from Accumulator output with acknowledge *ack_r* to ALU. Operation OR is performed. The operation result is at port *q* and the signal *req_w* is asserted to write this result to the Accumulator. Accumulator sends the acknowledgement *ack_w* after storage the result in reg. The acknowledgement *ack_in_idec* is asserted after instruction finalization and sent to Instruction decoder.

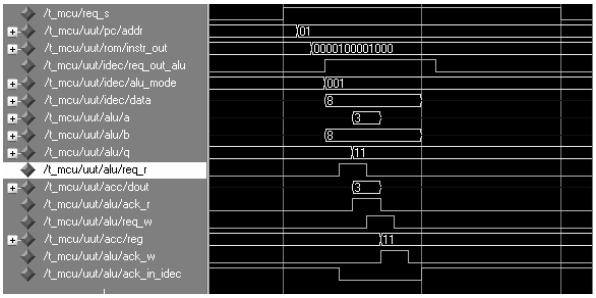

Fig. 8 Instruction process simulation waveforms in Modelsim

V. Conclusion

The simple asynchronous circuit application in the VHDL simulation model form was introduced in this article. The design process of simulation model construction was explained. The result construction was simulated by functional simulation and the results were shown in simulation diagram.

Even if the construction seems to be complex, it has global benefits of asynchronous circuits. The others implementations as Null Convention Logic will be more efficient in future.

The next task is to design functional microcontroller for synthesis and following implementation to FPGA. The parameters between synchronous and asynchronous constructions will be compared on this physical platform.

ACKNOWLEDGEMENT

Research described in the paper was financially supported by the Czech Grant Agency under grant No. GA102/08/H027 "Advanced Methods, Structures and Components of Electronic Wireless Communication", and Research Programme of Brno University of Technology MSM0021630513 "Electronic Communication Systems and New Generation Technology (ELKOM)".

REFERENCES

[1] Sparso, J. Furber, S. *Principles of Asynchronous Circuit Design - A System Perspective*. Boston: Kluwer Academic Publishers, 2001.

[2] Hauck, S. *Asynchronous Design Methodologies: An Overview*. In Proceedings of the IEEE, Vol. 83, No. 1, pp. 69-93, January, 1995.

[3] Hwang, O. E. *Digital Logic and Microprocessor Design with VHDL*. Riverside: La Sierra University, 2004. ISBN: 0-534-46593-5.

[4] Perry, L. D. VHDL: Programming by Example: Fourth Edition. McGraw-Hill, 2002.