FBU-CPU

Veysel ÖZASLAN, Nisa AKSOY, Miray PINARBAŞI, İrfan KARAŞOĞLU

Istanbul Türkiye

e-mail: { [veysel.ozaslan@stu.fbu.edu.tr](mailto:veysel.ozaslan@stu.fbu.edu.tr), [nisa.aksoy@stu.fbu.edu.tr](mailto:nisa.aksoy@stu.fbu.edu.tr), [miray.pinarbasi@stu.fbu.edu.tr](mailto:miray.pinarbasi@stu.fbu.edu.tr), [irfan.karasoglu@stu.fbu.edu.tr](mailto:irfan.karasoglu@stu.fbu.edu.tr) },

*Abstract* — FBU-CPU processor design is based on Vonn Neumann architecture and implemented with state machine approach. This processor design was tested with assembly language test codes and includes Control Units and Arithmetic Logic Unit (ALU) system. This processor design was implemented and examined in 3 different cases. As a result of all these, our algorithmic thinking ability improved.

Keywords — FPGA, CPU.

# Introductıon

This FBU-CPU processor design aims to design a processor that follows the principles based on the Vonn Neumann architecture. This architecture integrates memory, processing and control units into a unified system. The initial goal is to develop a robust and simple processor that processes basic arithmetic and logical operations while demonstrating efficient control flow in the state machine.

# System Archıtecture

The system architecture of FBU-CPU design consists of 4 elements: Custodians, Memory (RAM), Processing Unit (ALU), Control Unit. Memory holds the opetaring instructions and variables in the memory. Processor unit is required to decode the instructions. Storers are storage areas used in various tasks.

# Software Used

We coded this processor design as a state machine. The case in the code looks at the state and progresses according to the structures in the state. Also, the reason we use the status register is that the register, which specifies when the mechanisms in the system will work, works according to these steps. In Case 0, the PC value is transferred to MAR for fetching the instruction from RAM and the state is moved to Case 1. In Case 1, the received instruction is written to IR and PC is incremented to prepare for the next instruction. Case 2 checks the opcode to determine the type of operation. If the opcode indicates an opcode bound to less than 6 addresses, the corresponding address is retrieved and the control flows for the special instructions are executed accordingly. In Case 3, arithmetic or memory operations such as LOD, STO, ADD, SUB, and MUL are performed and updated in ACC. Finally, Case 4 halts the processor and puts it into an idle loop that exits only when the system resets.

# Results

The operations supported by the developed processor are taken from the FBU-CPU ISA (Instruction Set Architecture) table. When expressing the FBU processor in the Verilog language, the ALU/ arithmetic operation unit, is the section where arithmetic operations are performed. There are: arithmetic operations in the FBU-CPU. These are addition, subtraction and multiplication. Operations are performed according to the incoming operation code and written to the ACC register. Test software, in all cases were tried one by one and examined on the simulation on Vivado. Our algorithmic thinking ability has improved and our command of the verilog language has increased from this project.

##### Project team

Veysel ÖZASLAN: 220304039

Nisa AKSOY: 230304047

Miray PINARBAŞI: 220304044

İrfan KARAŞOĞLU: 220304051

OUR PROJECT LINK

Presentation: <https://youtu.be/yfMe_w1z8J4?si=GkTSvozOQ78wqC_j>

Files: <https://github.com/OzaslanVEYSEL/FBU-CPU-RTL>

##### Reference Files

<https://github.com/OCagy/FB_CPU/blob/main/FB_CPU.v> ,<https://github.com/Azizburak2/FBU-CPU-PROJECT/blob/main/FBU-CPU-PROJECT/CODE.txt> , <https://youtu.be/vm8iiVlP2hs?si=MRDXFNPyvX4ZI-5z> ,<https://youtu.be/_GoEawF9GOA?si=76KPp4TqI9oECrvn> ,<https://github.com/hilala7/FBU-CPU/blob/main/CODE.txt>