NYU-6463 PROCESSOR

ACHD

Aditya Kaushal(ak5656), Parteek Singh Khushdil(psk287), Shashank Garg(sg4437), Varun Sharma(vs1638)

INTRODUCTION

At present, the processing technology that we majorly use has seen tremendous improvement at a very fast pace in past years. However, it is believed to soon saturate in performance. Processors core implement sequential execution of programs which proves to be a bottleneck in the fast growing high speed computing applications like image processing, data science and various other machine learning applications. On the other hand, FPGA makes use of the hardware parallelism to execute millions of instructions concurrently. This makes FPGA a strong candidate to replace the traditional processors. A clear example of this can be seen in the industry as big companies like amazon have started replacing their data centers with FPGA technologies.

In our project we aim to do a similar task at a low scale. We intend to implement a MIPS based processor(NYU-6463 Processor) on FPGA hardware and validate the performance improvements. We also intend to run the algorithm of complete RC5 encryption on the designed processor using assembly instruction set of the designed processor.

IMPLEMENTATION

The NYU-6463 processor has a modular design, containing the following modules:

• **Control Unit:** It controls all the modules by generating the required control signals like Instruction type, ALU operation and maintains the data flow through **Program Counter (PC)** that contains the address of the next instruction to be executed by the processor. PC value is calculated differently if it's a Branch or Jump instruction and incremented by one for rest of the instructions.

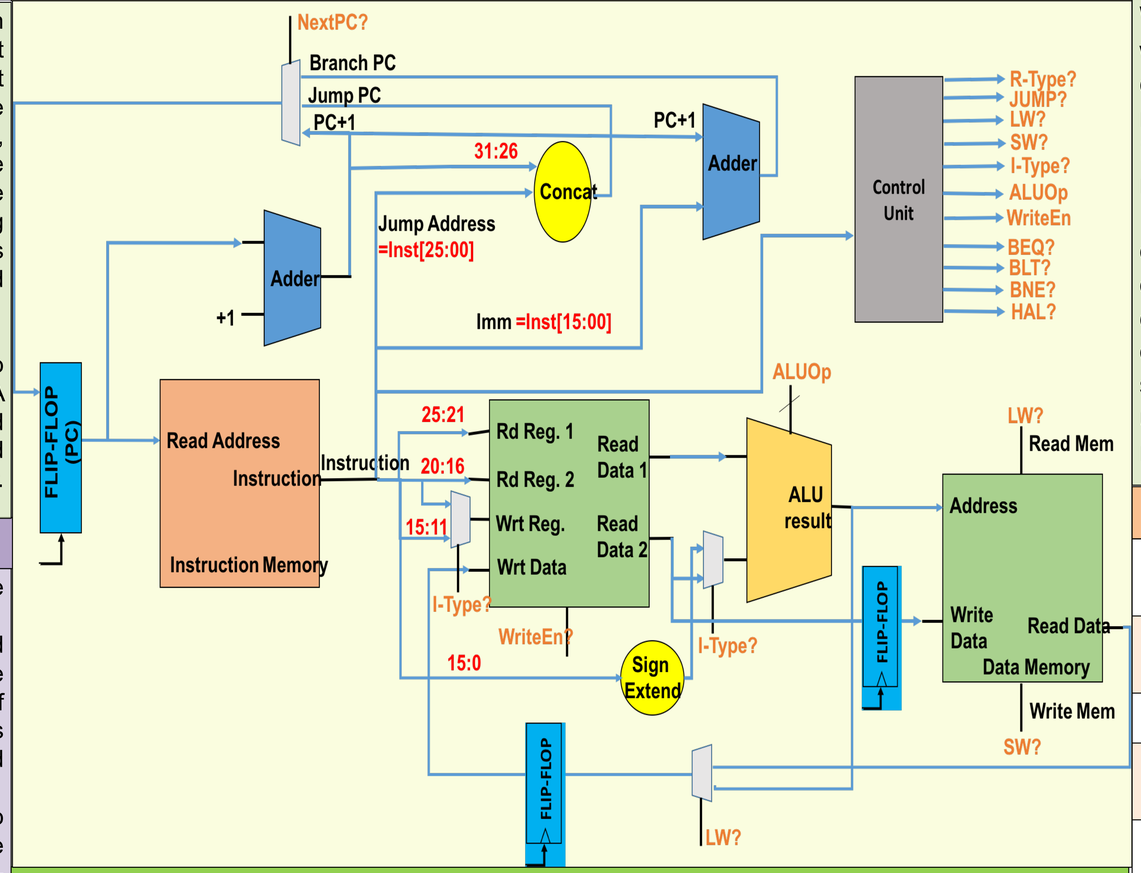
• **Instruction and Data Memory**: The instruction memory is initialized to contain the complete RC5 program to be executed and instruction are fetched based on the value of PC. The data memory is initialized with S-array and input for encryption and is accessed using load instructions. The Skey, encrypted and decrypted output are stored in the data memory after the execution using the store instruction.

• **Decode Unit & Register File:** It contains 32, 32-bit registers. It takes 32-bits of the instruction and decode it to find the 5-bit address of the operand registers and destination register the 32 registers. The register file supports two independent register reads and result is written to the destination register or data memory in next clock cycle.

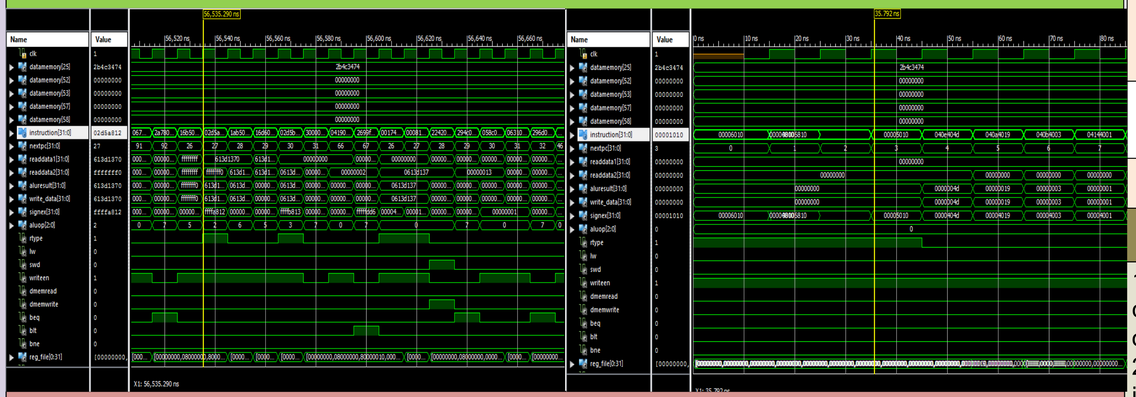
• **ALU:** This block performs operations such as ADD, SUB and other logic operations. It uses the control signals generated by the Control Unit, as well as the data from the registers or from the instruction directly. It computes data that can be written into one of the registers (including PC).

The designed processor utilizes the hardware parallelism as multiple operations are performed in parallel in different modules like Instruction Memory, Decode Unit(Register File), ALU, Control Unit and Data Memory to completely execute a single instruction. The results of those operations are transferred simultaneously from one module to other to further execute the instruction based on control signals.

BLOCK DIAGRAM



SIMULATION



CONCLUSION

We ran the RC5 encryption algorithm on a simulator designed in C++ and it took around 3 seconds to complete the execution. Whereas, on the designed processor, the algorithm took 90 μsecs on FPGA board. It is clear that there were tremendous performance improvements due to hardware parallelism.

DESIGN VERIFICATION

We verified every part of the project individually and when tied together as well. To verify the assembly code we developed a simulator on C++ platform that reads assembly instruction line by line and interprets it as a processor. We validated the C++ simulator by running various Assembly instructions of the processor and checking the result. We also verified every VHDL module like ALU, Control Unit, Decoder etc. individually before tying them all. To verify the complete design we matched the output for the RC5 encryption algorithm running on the processor for some predefined inputs for which the outputs are universally known. The processor gave the correct result every time which validated our design.

PERFORMANCE METRIC

Parameters	After Synthesis	Post Route
Max Combinational Path Delay	0.392 ns	--
Critical Path Delay	2.489 ns	--
Speed of operation	401.816 MHz(Syn. Report)	
No. of Sliced LUTS	278/63400	237/63400(1%)
No. of sliced registers	61/126800	61/126800(1%)
No. of fully used LUT-FF Pairs	60/279	55/243(22 %)
No. of bonded IOBs	54/210	54/210

FUTURE WORKS

1. This design can be extended by implementing dynamically changing of Instructions and updating the Instruction memory on run time.
2. The performance of the processor can be enhanced by implementing the pipelined based design.
3. Programming interface can be improved by developing a higher level language compiler that can convert the C/C++ code to the Assembly Instruction set of the processor and finally to the machine code.