

# Power optimization of the CV32A6 RISC-V soft-core

Yasser Ajellabi<sup>1</sup>, Willy Perrin<sup>1</sup>, Adrien Stoeklin<sup>1</sup>, Hamady Ka<sup>1</sup>, Freddy Anstotz<sup>2</sup>

**Abstract**—The power optimization of processors is nowadays some of the challenging topics that need to be understood and studied in order to find effective solutions. It's a real concern for the cutting edge technologies of today. That's why we're being part of the Thales Group competition as RiscEtMorty team for the power optimization of the CVA32A6 RISV-V soft core which is implemented in an FPGA Zybo-Z7 architecture of the Xilinx Zinq-7000 family. The hardware description language is System Verilog. Some power reduction techniques have been studied but just some of them is implemented by our team.

**Index Terms**—RISC-V, ISA, FPGAs, SRAMs, Power Optimization, Reduction Techniques, Dynamic Power, Memory Read Access, Memory Splitting.

## I. INTRODUCTION

RISC-V processors are new, open and free ISA (Instruction Set Architecture) developed by the University of California and its foundation is supported by many companies. It's a RISC-based architecture that has the instructions of a standard RISC processor but with some optional extensions of this one. It's a base integer ISA with no branch delays and the support for optional variable length encodings. RISC-V are known by their fast growth and adoption and also the fact that they are supported by GCC, Clang, Linux Kernel but also several RISC-V simulators and emulators. The target Hardware is an FPGA Zybo-Z7 from Xilinx.

FPGAs (Field Programmable Gate Arrays) are semiconductor devices that are based around a matrix of configurable logic blocks connected via programmable interconnects. The key feature of FPGAs is that they are reprogrammable respecting application requirements which highlight their difference with the ASIC boards. They are

made of thousand of Configurable Logic Blocks (CLBs) connected with a network of Interconnects and they are made up also of Input Output Blocks (IOBs). CLBs can be LUTs (Look Up Tables) which are function generators based on SRAM memories which realize logic tables, they contain also Flip-Flops, Multiplexers and so on. FPGAs can also contain others components like PLLs, Memory Controllers, DSPs and High Speed Transceivers. Nowadays FPGAs can also embed Hard Processors or even Soft Processors. The fact that FPGAs can be reconfigured allow them to be used in acceleration computing or in many other applications where optimization and reconfiguration are needed frequently. There are used also for design prototype.

Despite all of these features FPGAs present some disadvantages like low frequency rates, power consumption and low performance respecting some other circuits like ASICs but this fact cannot avoid them to be used in many applications. Nowadays FPGAs are optimized in a way that they can be programmed at high speed allowing them to be used in high speed applications.

The power consumption is a headache inside FPGAs (and in large ICs in general) and a part of this power rate is caused by the SRAM memories used to control the switches for the flexibility of the FPGAs. It's in this concern that our team adopts the Memory Splitting power reduction technique in order to come over this power rate.

In fact many power reduction techniques have been studied during the researches for the competition such as: Clock Gating, Power Gating also known as Power Shut Off (PSO), Multi-VDD, Multi-Vth and RAM power reduction. Finally the RAM power reduction has been implemented through the SRAM blocks Memory Splitting that will be discussed in the next section.

<sup>1</sup>RiscEtMorty Team, Université de Strasbourg, Faculté de Physique et Ingenierie

<sup>2</sup>Freddy Anstotz, Enseignant, Université de Strasbourg, Faculté de Physique et Ingenierie

## II. MEMORY SPLITTING

SRAM-based blocks represent a large area inside FPGAs and they are also responsible of power increasing. That's why a soft implementation of the Memory Splitting reduction technique to reduce the dynamic power consumption is the solution adopted in this report. In fact the write operations consume slightly less power than the read access, so the idea is trying to reduce the number of the read accesses or the power cost of a read access. The idea behind this technique is to reduce the frequency inside the equation of the dynamic power which is:

$$P = \alpha CV^2 f \quad (1)$$

Where  $\alpha$  is the activity factor that relates to how many  $0 \rightarrow 1$  or  $1 \rightarrow 0$  transitions occur in a chip,  $V$  the Voltage Supply,  $f$  the Frequency and  $C$  the physical Capacitance.

The SRAM memory blocks are divided into smaller blocks activated only in needed ones in each access. In fact the blocks can independently transition into different power modes. A minimal data clustering can be used by Compilers and Operating Systems in just few blocks then powering down the unused ones which reduce the power.

Another advantage that make us choose this technique is that it does not require a huge architecture changing. In our case the 64k RAM has been divided in two blocks of 32k RAM. To do so we have used the description model of RAM in the architecture and then created two identical 32k SRAM. After doing that we have added a condition in order to choose in what sub-block the data is going to be read allowing the other block to be shut off during the read operation. Here below is the architecture of the Memory Splitting using two blocks of 32k RAM.

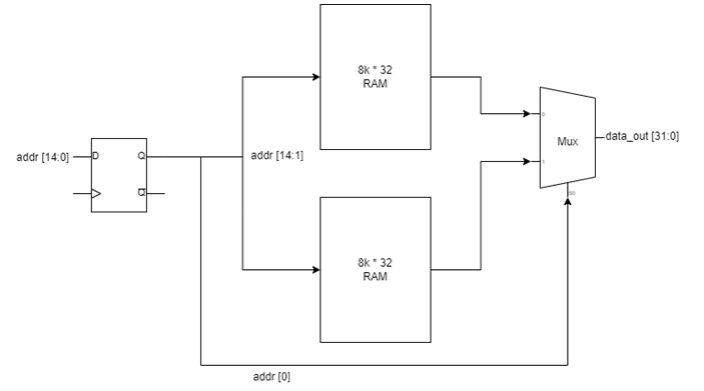


Fig1: Memory Splitting Architecture (2X32k Block RAM).

## III. RESULTS BEFORE IMPLEMENTATION: REFERENCE

Here below we have the results with the reference project before our architecture implementation.

Before implementation	Values
Total Power as reported by Vivado (mw)	307
Number of Cycles	2098749
Actual Clock Period(ns)	22.2
Reference Processing Time (ms)	49.59
FPGA Resources	14675 LUTs, 9292 FFs, 36 RAM36
Credence	82

Fig1: Reference

## IV. RESULTS AFTER IMPLEMENTATION

After the implementation of the Memory Splitting, a little power gain has been noticed by the diminution of 0.001 in the operand reads. At energy level we have the following results after calculation. A diminution of 0.07% has been noticed on the number of cycles.

- Reference Energy: **14.30 mJ**
- Energy after implementation: **14.24 mJ**
- Energy Gain: **- 0.41 %**
- Time of Simulation: **46.5585 ms**

Fig2: Energy Gain Report.

After implementation	Values
Total Power as reported by Vivado (mw)	306
Number of Cycles	2097231
Actual Clock Period(ns)	22.2
Reference Processing Time (ms)	46.5585
FPGA Resources	14675 LUTs, 9292 FFs, 36 RAM36
Credence	82

Fig3: Table after implementation

Name	Power (W)	Name	Power (W)
core_clk_20	0.131	core_clk_20	0.131
l_arbiter	0.003	l_arbiter	0.003
l_arbiter_1	0.003	l_arbiter_1	0.003
l_arbiter_2	0.003	l_arbiter_2	0.003
l_arbiter_3	0.003	l_arbiter_3	0.003
l_arbiter_4	0.003	l_arbiter_4	0.003
l_arbiter_5	0.003	l_arbiter_5	0.003
l_arbiter_6	0.003	l_arbiter_6	0.003
l_arbiter_7	0.003	l_arbiter_7	0.003
l_arbiter_8	0.003	l_arbiter_8	0.003
l_arbiter_9	0.003	l_arbiter_9	0.003
l_arbiter_10	0.003	l_arbiter_10	0.003
l_arbiter_11	0.003	l_arbiter_11	0.003
l_arbiter_12	0.003	l_arbiter_12	0.003
l_arbiter_13	0.003	l_arbiter_13	0.003
l_arbiter_14	0.003	l_arbiter_14	0.003
l_arbiter_15	0.003	l_arbiter_15	0.003
l_arbiter_16	0.003	l_arbiter_16	0.003
l_arbiter_17	0.003	l_arbiter_17	0.003
l_arbiter_18	0.003	l_arbiter_18	0.003
l_arbiter_19	0.003	l_arbiter_19	0.003
l_arbiter_20	0.003	l_arbiter_20	0.003
l_arbiter_21	0.003	l_arbiter_21	0.003
l_arbiter_22	0.003	l_arbiter_22	0.003
l_arbiter_23	0.003	l_arbiter_23	0.003
l_arbiter_24	0.003	l_arbiter_24	0.003
l_arbiter_25	0.003	l_arbiter_25	0.003
l_arbiter_26	0.003	l_arbiter_26	0.003
l_arbiter_27	0.003	l_arbiter_27	0.003
l_arbiter_28	0.003	l_arbiter_28	0.003
l_arbiter_29	0.003	l_arbiter_29	0.003
l_arbiter_30	0.003	l_arbiter_30	0.003
l_arbiter_31	0.003	l_arbiter_31	0.003
l_arbiter_32	0.003	l_arbiter_32	0.003
l_arbiter_33	0.003	l_arbiter_33	0.003
l_arbiter_34	0.003	l_arbiter_34	0.003
l_arbiter_35	0.003	l_arbiter_35	0.003
l_arbiter_36	0.003	l_arbiter_36	0.003
l_arbiter_37	0.003	l_arbiter_37	0.003
l_arbiter_38	0.003	l_arbiter_38	0.003
l_arbiter_39	0.003	l_arbiter_39	0.003
l_arbiter_40	0.003	l_arbiter_40	0.003
l_arbiter_41	0.003	l_arbiter_41	0.003
l_arbiter_42	0.003	l_arbiter_42	0.003
l_arbiter_43	0.003	l_arbiter_43	0.003
l_arbiter_44	0.003	l_arbiter_44	0.003
l_arbiter_45	0.003	l_arbiter_45	0.003
l_arbiter_46	0.003	l_arbiter_46	0.003
l_arbiter_47	0.003	l_arbiter_47	0.003
l_arbiter_48	0.003	l_arbiter_48	0.003
l_arbiter_49	0.003	l_arbiter_49	0.003
l_arbiter_50	0.003	l_arbiter_50	0.003
l_arbiter_51	0.003	l_arbiter_51	0.003
l_arbiter_52	0.003	l_arbiter_52	0.003
l_arbiter_53	0.003	l_arbiter_53	0.003
l_arbiter_54	0.003	l_arbiter_54	0.003
l_arbiter_55	0.003	l_arbiter_55	0.003
l_arbiter_56	0.003	l_arbiter_56	0.003
l_arbiter_57	0.003	l_arbiter_57	0.003
l_arbiter_58	0.003	l_arbiter_58	0.003
l_arbiter_59	0.003	l_arbiter_59	0.003
l_arbiter_60	0.003	l_arbiter_60	0.003
l_arbiter_61	0.003	l_arbiter_61	0.003
l_arbiter_62	0.003	l_arbiter_62	0.003
l_arbiter_63	0.003	l_arbiter_63	0.003
l_arbiter_64	0.003	l_arbiter_64	0.003
l_arbiter_65	0.003	l_arbiter_65	0.003
l_arbiter_66	0.003	l_arbiter_66	0.003
l_arbiter_67	0.003	l_arbiter_67	0.003
l_arbiter_68	0.003	l_arbiter_68	0.003
l_arbiter_69	0.003	l_arbiter_69	0.003
l_arbiter_70	0.003	l_arbiter_70	0.003
l_arbiter_71	0.003	l_arbiter_71	0.003
l_arbiter_72	0.003	l_arbiter_72	0.003
l_arbiter_73	0.003	l_arbiter_73	0.003
l_arbiter_74	0.003	l_arbiter_74	0.003
l_arbiter_75	0.003	l_arbiter_75	0.003
l_arbiter_76	0.003	l_arbiter_76	0.003
l_arbiter_77	0.003	l_arbiter_77	0.003
l_arbiter_78	0.003	l_arbiter_78	0.003
l_arbiter_79	0.003	l_arbiter_79	0.003
l_arbiter_80	0.003	l_arbiter_80	0.003
l_arbiter_81	0.003	l_arbiter_81	0.003
l_arbiter_82	0.003	l_arbiter_82	0.003
l_arbiter_83	0.003	l_arbiter_83	0.003
l_arbiter_84	0.003	l_arbiter_84	0.003
l_arbiter_85	0.003	l_arbiter_85	0.003
l_arbiter_86	0.003	l_arbiter_86	0.003
l_arbiter_87	0.003	l_arbiter_87	0.003
l_arbiter_88	0.003	l_arbiter_88	0.003
l_arbiter_89	0.003	l_arbiter_89	0.003
l_arbiter_90	0.003	l_arbiter_90	0.003
l_arbiter_91	0.003	l_arbiter_91	0.003
l_arbiter_92	0.003	l_arbiter_92	0.003
l_arbiter_93	0.003	l_arbiter_93	0.003
l_arbiter_94	0.003	l_arbiter_94	0.003
l_arbiter_95	0.003	l_arbiter_95	0.003
l_arbiter_96	0.003	l_arbiter_96	0.003
l_arbiter_97	0.003	l_arbiter_97	0.003
l_arbiter_98	0.003	l_arbiter_98	0.003
l_arbiter_99	0.003	l_arbiter_99	0.003
l_arbiter_100	0.003	l_arbiter_100	0.003

Fig5: Power diminution in the read operands.

## V. CONCLUSION

During the competition, many techniques of power optimization have been studied and two of them have been implemented. In fact we have tried the Memory Splitting method and the Clock Gating method. The second one didn't give amelioration in the power. The first one has given at least a little diminution of 0.33 % in the energy consumption.

We thank the Thales Group for giving us the opportunity to be part of this challenge. We have learnt a lot regarding power optimization techniques and how to implement them efficiently. We thank also our Professor Freddy Anstotz for being our mentor and make us able to be part of this journey too.

## REFERENCES

- [1] 1. Naresh Grover; Dr. M.K.Soni; (2012). "Reduction of Power Consumption in FPGAs - An Overview".
- [2] Shashi Kumar V; Gurusiddhaya Hiremath; (2016). "Low Power Implementation of RISC-V Processor".
- [3] VASANTH VENKATACHALAM; MICHAEL FRANZ; (2005). "Power Reduction Techniques For Microprocessor Systems".
- [4] M.Sathish Kumar, S.K.Senthil Kumar, R.Venkatesh, (2016). "Design and Implementation of Clock Gated ALU performing 16 functions using DeMux and AND Gate".
- [5] L. Sterpone; L. Carro; D. Matos; S. Wong; F. Fakhar; (2011). "A New Reconfigurable Clock-Gating Technique for Low Power SRAM-based FPGAs".
- [6] Ajay Kumar Singh; (2017). "Power Efficient Data-Aware SRAM Cell for SRAM-Based FPGA Architecture".
- [7] X.S. Zhang, Washington University "Power Optimization (Part 1 and Part 2)".
- [8] Smitha Sundaresan, Frederic Rivoallon,(2012) "Analysis of Power Savings from Intelligent Clock Gating".
- [9] M. Gautschi et al.,(2017), "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices".