Optimizing the Leading-Zero Counter for FPGA applications

MEEBED Abdallah
Grenoble-INP Phelma
Grenoble, France
abdallah.meebed@phelma.grenoble-inp.fr

DEVILLARD Hugo
Grenoble-INP Phelma
Grenoble, France
hugo.devillard@phelma.grenoble-inp.fr

LEVEUGLE Régis
TIMA Grenoble-INP
Grenoble, France
regis.leveugle@univ-grenoble-alpes.fr

***Abstract*—This paper explores the designs of Leading-Zero Counters adapted for FPGA synthesis of the CVA6, a RISC-V CPU, to increase slack time.**

***Index Terms*—leading-zero counter, FPGA, RISC-V**

I. Introduction

Leading-Zero Counters (LZC) are an essential component for multiple reasons, notably for floating point operations. In the RISC-V privileged specification, they are used to count trailing ones of the Physical Memory Protection (PMP) configuration register in order to calculate the boundary of memory regions. This paper analyzes the critical path of the CVA6 on the Xilinx ZyboZ7-20 FPGA in order to increase the maximum working frequency. In the following sections we explore multiple architectures of LZCs followed by an evaluation of the implementation used.

II. Preliminaries

A LZC is a circuit that is responsible for counting the leading or trailing zeroes or ones of an address. For example the binary datum 000101 has a leading zero count of 3. The RISC-V privileged architecture offers optional Physical Memory Protection (PMP) units. The user can enter multiple entries of memory regions that are reserved for certain privileged accesses, as well as read, write and execute rights for each region. Each memory check is then compared against the existing configuration. An exception is raised when the current privilege level does not have access to the memory region. The user can use multiple ways to define the range of addresses (c.f Table I). If the user configures the PMP with NAPOT, the number of trailing ones of the `pmpaddr` encode the size of the range.

A	Name	Description
0	OFF	Null region (disabled)
1	TOR	Top of range
2	NA4	Naturally aligned four-byte region
3	NAPOT	Naturally aligned power-of-two region

TABLE I
Encoding of the PMP configuration registers (Table 18 of RISC-V privileged)

For this reason, a LZC can be used to calculate bound addresses, which is the case in the CVA6. The identified critical path for the CVA6 goes from the PMP address register in the Control Status Register (CSR) and ends at the SRAM addresses. The path propagates through 28 logic levels.

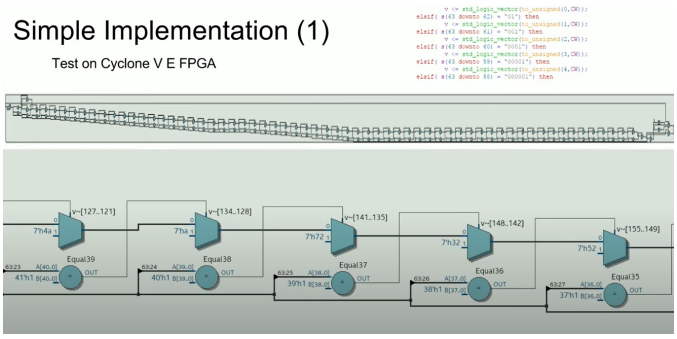

Fig. 1. FPGA synthesis of the simple implementation [1]

III. Existing Leading-Zero Counter architectures

A. Simple implementation

A simple implementation of the LZC is a long list of if statements that matches the input with the a list of predefined values [1]. For example

```
if (in[31:29] == 3'b001) out = 3;
```

The previous implementation is the most straightforward implementation but the logic scales linearly with the length of the address (c.f Figure 1).

B. Implemented architecture

The current implemented architecture of the LZC is a tree that scales logarithmically with the address length. This greatly decreases the layers of logic and by extension the critical path if the component is part of it. There is an increase in power and number of LUTs used since the tree structure introduces extra logic to combine intermediate results.

C. Suggested architecture

The following architecture is presented in [2]. Essentially, the LZC uses a modular design by splitting the number into multiple nibbles of 4 bits, followed by giving the Nibble Local Count (c.f Figure 2) for each nibble, in the form of a_i, a boolean saying if there's a 1 in the nibble and a 2-bit bus Z_i which gives the local leading zero count.

The other component is a Boundary Nibble Encoder (cf. Figure 3) which detects the first nibble containing a 1, encodes its value and passes this value to multiplexer which selects the corresponding Z_i.

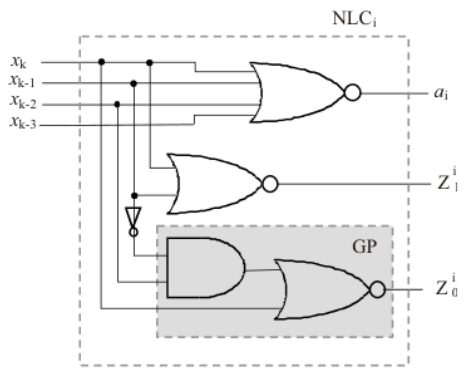

Fig. 2. Nibble Local Count [2]

The final combination of NLC and BNE are on Figure 4. The claimed advantage of this design is the better use of LUTs in FPGAs, making it possible to use less units, which implies a shorter critical path and a higher frequency.

IV. Results

The architecture suggested in [2] was implemented on a 32 bit length address, which is already used in the current configuration of CVA6. Only the Leading-Zero Counter that is implemented in the PMP was changed.

A. Original results

For this competition, Vivado version 2024.2 was used for synthesis and place and route. The original slack is $0.664\,\text{ns}$, giving a maximal frequency of $\frac{1}{25\,\text{ns}-0.664\,\text{ns}} = 41.09\,\text{MHz}$. The FPGA uses 13094 LUTs.

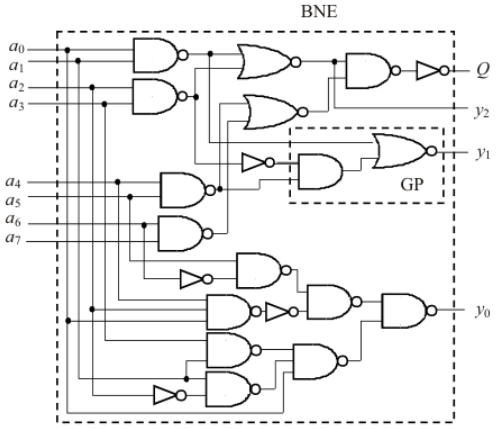

Fig. 3. Boundary Nibble Encoder [2]

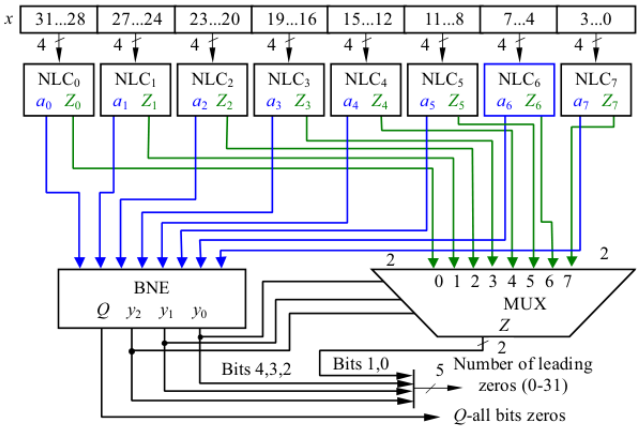

Fig. 4. Architecture of the LZC for 32 bit [2]

B. Improved results

After optimizing the LZC, the new slack is $0.732\,\text{ns}$, giving a maximal frequency of $\frac{1}{25\,\text{ns}-0.732\,\text{ns}} = 41.21\,\text{MHz}$ giving an increase of $0.29\,\%$. This increase is practically insignificant, meaning that a better solution might be to pipeline the critical path. The main advantage of this approach is the fact that this approach has no compromises on a functional level. There is no performance loss and no differences in cycles between the original and the optimized design. The design uses 13625 LUTs, which was not predicted. The claim of the implementation included also a reduction of the LUTs used, however, such results can differ between FPGAs.

C. Simulation results

As stated in the previous section, there are no differences between the simulations, as no extra stages of pipelines were added, simplifying the verification of design (Figure 5 for Coremark and 6 for MNIST results)

```
[UART]: 2K performance run parameters for coremark.
[UART]: CoreMark Size    : 666
[UART]: Total ticks      : 1151831
[UART]: Total time (secs): 1.151831
[UART]: Iterations/Sec   : 2.604549
[UART]: Iterations       : 3
[UART]: Compiler version : GCC13.1.0
[UART]: Compiler flags   :
[UART]: Memory location  : BRAM
[UART]: seedcrc          : 0xe9f5
[UART]: [0]crclist       : 0xe714
[UART]: [0]crcmatrix     : 0x1fd7
[UART]: [0]crcstate      : 0x8e3a
[UART]: [0]crcfinal      : 0x2e87
[UART]: Correct operation validated. See README.md for run and reporting rules.
[UART]: CoreMark 1.0 : 2.604549 / GCC13.1.0     / BRAM
```

Fig. 5. Coremark application results

V. Conclusion

In this competition, an essential combinatorial bloc was explored and evaluated to see if it improves considerably the maximum frequency of the CPU. The approach taken doesn't significantly improve and further optimizations are required on the critical path. This competition also highlights how the

```
# [TB]                14838 - Writing the boot address into dpc
# [TB]                20970 - Resuming the CORE
# [UART]: Expected  = 4
# [UART]: Predicted = 4
# [UART]: Result : 1/1
# [UART]: credence: 82
# [UART]: image env0003: 1760387 instructions
# [UART]: image env0003: 2790524 cycles
```

Fig. 6. MNIST application results

hardware target (FPGA or ASIC) play a role at the design and RTL stages and how different approaches should be considered in order to best use the resources. This idea also applies depending on the target technology and library within either ASICs or FPGA designs. Leveraging a target's strengths is one of the key challenges when it comes to efficient designs.

REFERENCES

[1] RTL Engineering. *Designing an Efficient Leading Zero Counter*. Youtube. 2018. URL: https://www.youtube.com/watch?v=lZ1DqG0Pn_I.

[2] Nebojsa Z Milenkovic, Vladimir V Stankovic, and Miljana Lj Milic. "Modular design of fast leading zeros counting circuit". In: *Journal of Electrical Engineering* 66.6 (2015), p. 329.