RISC-V HARDWARE MODIFICATION FOR M-SEQUENCES GENERATION

Petrovsky D., Ivaniuk A.
Faculty of Computer Systems and Networks, Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus
E-mail: petrovsky.dmitr@gmail.com, ivaniuk@bsuir.by

A hardware modification of the soft processor core of the open RISC-V architecture to accelerate the generation of M-sequences is being considered. The results of a comparative analysis of the performance of completely software algorithms and an algorithm with support for hardware modification are shown, and the hardware costs for implementation in a Xilinx-7 FPGA chip are calculated.

Introduction

The main advantage of processor systems is the ability to perform a wide range of tasks, such as various mathematical algorithms and control systems. To speed up the execution of which, the following approaches can be used: software optimization, adding specialized co-processors, and a combination of hardware and software modifications. This article will discuss the third approach.

I. Problem statement

As a processor core we will use the currently popular RISC-V[1] architecture, into the structure of which hardware changes will be introduced. This architecture supports instruction sets with different bit depths as shown in Table 1.

Table 1 – Basic instruction sets

Abbreviation	Name
RVWMO	Basic memory consistency model
RV32I	Basic set with integer operations, 32-bit
RV64I	Basic set with integer operations, 64-bit
RV128I	Basic set with integer operations, 128-bit

In addition to the basic sets, this architecture also contains additional sets that expand the functionality, as shown in Table 2.

Table 2 – Standard unprivileged command sets

Abbreviation	Name
M	Integer Multiplication and Division
F	Single-Precision Floating-Point
D	Double-Precision Floating-Point
V	Vector Operations

A pseudo-random sequence (PRS) is a sequence of numbers that was calculated according to some arithmetic rule, but has all the properties of a random sequence. One of the devices for generating PRS is a linear feedback shift register (LFSR), in which the value of the input (pushed) bit is equal to a linear Boolean function of the values of the remaining bits of the register before the shift. There are two types of feedback in LFSR, external feedback (Fibonacci configuration Figure 1.a) and internal feedback (Galois configuration Figure 1.b). Both implementations have the same generator functions, but the Galois configuration, due to the ability to parallelize XOR operations, allows for greater performance in both software and hardware implementations.

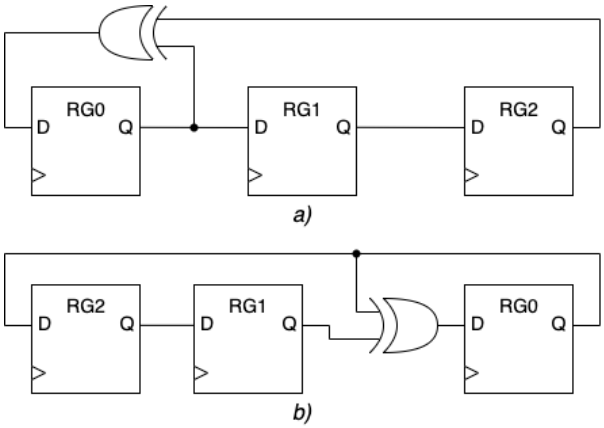

Figure 1 – Shift register with linear feedback a) Fibonacci configuration b) Galois configuration

The PRS with the maximum length is called an M-sequence and has the following properties:

- M-sequences are periodic with period $N = 2^n - 1$;
- the number of symbols taking the value one, over the length of one period of the M-sequence, is one more than the number of symbols taking the value zero;
- any combination of symbols of length n in one period of the M-sequence, with the exception of a combination of n zeros, occurs no more than once;
- a combination of n zeros is prohibited: only a sequence of only zeros can be generated on its basis.

The extensive use of M-sequences in cryptography problems, communication systems and digital signal processing algorithms makes it urgent to solve issues of optimization and acceleration of PRS generation algorithms.

II. Research results

A pipeline implementation of a soft processor of the RISC-V architecture with support for the

RV32I[2] instruction set was taken. This implementation contains a set of 32 registers and includes 39 instructions: fetch-store, logical and arithmetic operations, conditional and unconditional jump instructions.

To speed up M-sequence calculations, it is proposed to add additional logic to the general-purpose register and modify the command decoding device.

The additional logic for the general-purpose register will allow it to perform both the LFSR and general-purpose register functions. To make the general-purpose register work as a shift register between triggers we will place a two-input multiplexer, the selecting signal of which provides a choice between parallel write or shift operation. In the resulting shift register input bit is directed from the stage of elements AND/XOR allowing the implementation of external feedback in the LFSR. This structure(Figure 2) allows you to implement any polynomial to form a PSP with a dimension of up to 32-bit. The polynomial is specified in the 32-bit status control register (SCR), so a one set in bit number N adds the N+1 term to the polynomial. For example, a polynomial of the form « $x^{32}+x^{22}+x^2+x^1+1$» will correspond to the value of register 0x8020003. If this register takes a non-zero value, then the general-purpose register switches its mode to LFSR, if this SCR register took a zero value, the register functions as a normal general-purpose register.

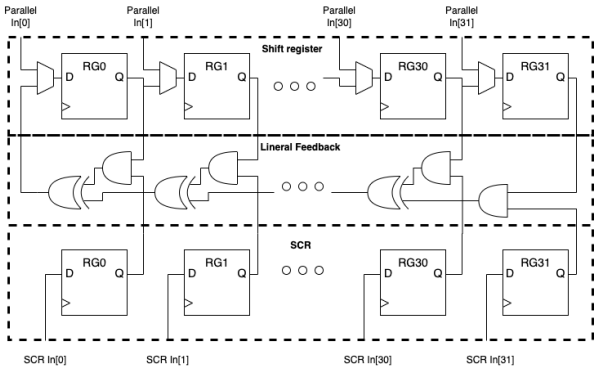

Figure 2 – LFSR block structure

Modification of the control device is an additional reading signal. This signal is generated for the register we have selected while executing commands that use this register.

This hardware modification allows the register with LFSR function to be used for operation with standard processor instructions. Thus, all logical operations, arithmetic operations and memory write operations will cause the next LFSR value to be calculated, so that a new value can be obtained every processor clock.

Here are the results of comparison of software implementations with our hardware modification. The Prog.1 fully corresponds to our hardware modification, it checks all bits of the polynomial in a loop and when the value of the bit "1" to perform the XOR operation, the shift always occurs. The Prog.2 computes only one polynomial, runs without a loop and without unnecessary shift operations. Table 3 shows for comparison the number of CPU cycles required to calculate one element of the M-sequence of the following polynomials «0x8020003, 0x8579D037, 0xFFFFFFFEE». The general graph of clock cycles consumed for all polynomial lengths is shown in Figure 3.

Table 3 – Comparison CPU cycles

Polynomial	Number of terms	Prog.1	Prog.2	Prog.3 with mod.
0x80200003	4	312	12	1
0x8579D037	16	309	36	1
0xFFFFFFEE	30	295	64	1

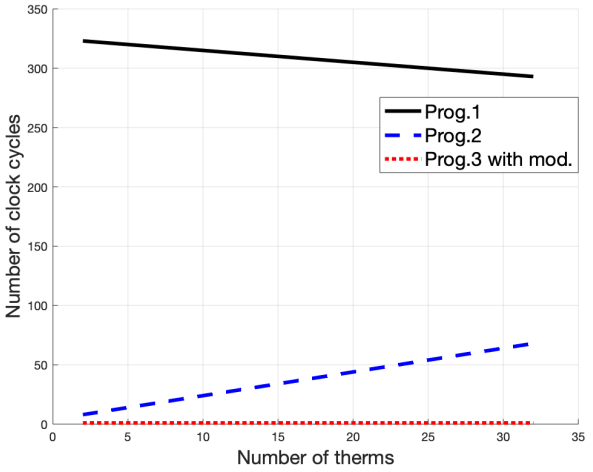

Figure 3 – Graph of the clock cycles

The RISC-V soft processor was implemented in a Xilinx-7 series FPGA chip, Table 4 shows the hardware overheads of the implementation without and with hardware modification, and the difference in their expenses.

Table 4 – table

Implementation	Slice	LUT's	Registers
Original RISC-V	704	1489	1433
Mod. RISC-V	789	1638	1465
Difference	85	149	32

The data in Tables 3-4 show that fully software implementations are inferior to hardware-software implementations in performance by a factor of 8 to 320. Hardware costs are not significant compared to the resources occupied by the soft processor core.

1. Official site RISC-V International [Electronic resource] – Mode of access: https://riscv.org. – Date of access: 26.10.2023.
2. Harris. S. L. Digital Design and Computer Architecture, RISC-V / S. L. Harris, D. Harris; – Morgan Kaufmann. Press, 2021. - 592 p