Łukasz Broll 225972

Krzysztof Dombek 226093

Organizacja i architektura komputerów

**Projekt:**

Skalowanie w systemie RNS

Prowadzący: dr inż. Piotr Patronik

Grupa: wtorek/TP godz. 13.15-15.00

POLITECHNIKA WROCŁAWSKA – CZERWIEC 2017

WYDZIAŁ ELEKTORNIKI

Spis treści

**Nie znaleziono żadnych pozycji spisu treści.**

# Wstęp

## teoria

Projekt skalowania w systemie RNS opierał się na dokumencie ‘2n RNS Scalers for Extended 4-Moduli Sets’ autorstwa Leonel Sousa. Najważniejszym zadaniem na tym etapie projektu było dokładne zrozumienie treści artykułu. Naukowy język oraz fachowość nazewnictwa w języku angielskim sprawiły nam duże problemy w interpretacji treści, a bez dokładnego zrozumienia całej idei skalowania nie bylibyśmy w stanie zrealizować dalszych kroków.

## Idea skalowania

Skalowanie liczb binarnych, czyli dzielenie przez potęgę 2 tj., podstawy systemu, nie może być zrealizowane wprost na liczbach dziesiętnych. Algorytm, na podstawie którego został zrealizowany projekt opiera się na resztowym systemie zapisu liczb. W procesie skalowania można wyróżnić następujące punkty:

1. Przekonwertowanie dzielnej na system **R**esidue **N**umber **S**ystem (resztowy).
2. Osobne przeskalowanie każdej z reszt R1-R4 w RNS przez dzielnik.
3. Przekonwertowanie wyniku z powrotem na system dziesiętny.

## Resztowy system zapisu liczb

Zgodnie z teoriąjest to system liczbowy służący do reprezentacji liczb całkowitych za pomocą wektora reszt z dzielenia względem ustalonego wektora wzajemnie względnie pierwszych modułów. W naszym przypadku ustalony wektor modułów wynosił: { 2n -1, 2n, 2n+1, 2n+1-1}. Chińskie twierdzenie o resztach gwarantuje, że taka reprezentacja jest jednoznaczna dla liczb całkowitych ze zbioru [0,M], gdzie M jest iloczynem wszystkich modułów.

## środowiska symulacji

### PARI/GP

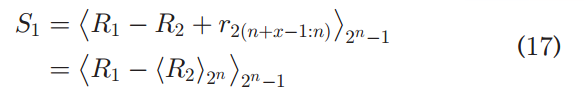
Do obliczeń czysto teoretyczno-matematycznych wykorzystaliśmy pogram Pari/GP – system komputerowy przeznaczony do szybkich obliczeń algebraicznych. Prosty interfejs konsolowy, rozbudowana dokumentacja i intuicyjna obsługa pozwoliły nam w miarę szybko zrozumieć jak program działa oraz jak korzystać z funkcji, które potrzebowaliśmy zaimplementować: tworzenie procedury, pętli, drukowanie wyników i operacje arytmetyczne. Aplikacja PARI/GP została przygotowana również pod systemy z rodziny Microsoft Windows i nasz program pisaliśmy na systemie Windows 10.

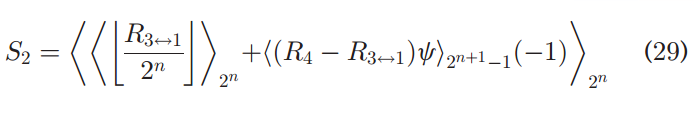
### Verilog

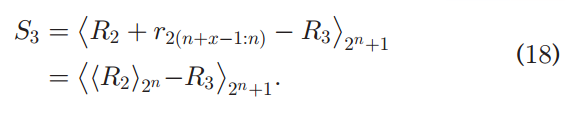
Język Veriloga wykorzystuje się do opisu sprzętu w projektowaniu oraz symulacji układów cyfrowych. Największą trudnością dla nas było zrozumienie całej idei języku opisu sprzętu. Do tej pory proste układy implementowaliśmy bezpośrednio na sprzęcie, bez komputerowego przygotowania algorytmu i symulacji. Skrócone kursy uczelniane, przykłady i dokumentacja również w tym przypadku wprowadziły nas w podstawy programowania językiem Verilog, co wystarczyło w naszym symulatorze na zadeklarowanie modułów, sygnałów, pętli, obliczeń arytmetycznych oraz modułu test\_bench. Dla potrzeb naszego projektu korzystaliśmy z internetowego środowiska kompilatora firmy Codingground dostępnego pod adresem https://www.tutorialspoint.com/compile\_verilog\_online.php

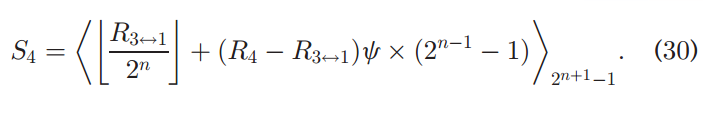
## Implementacja

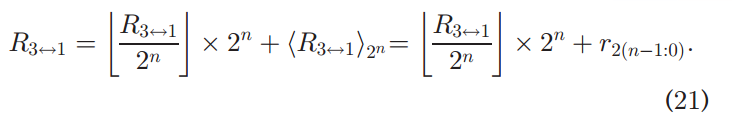
W naszej implementacji użyliśmy poniższych wyprowadzeń zastosowanych twierdzeń oraz wzorów z artykułu:

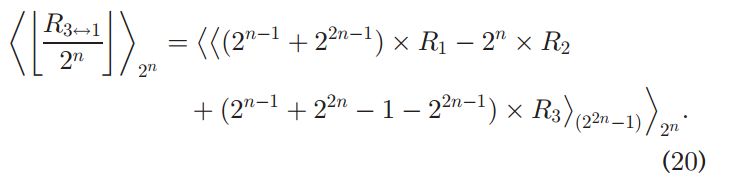


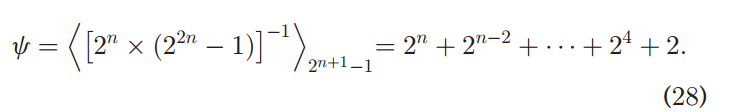












Kod źródłowy implementacji w Verilogu:

<https://pastebin.com/H45PZVZw>

W naszym symulatorze podajemy dowolny X oraz skalar, który potrzebny jest do wyliczenia poszczególnych modułów, jednak tak naprawdę na wejściu naszego układu podawane są wyliczone wcześniej reszty, na których operujemy, uzyskując ostateczny wynik. Każda z reszt S1-4 rozpisana jest w osobnym module, gdzie zadeklarowane są wykorzystywane porty. Dla każdego portu określamy jaką funkcję pełni – input, output, inout – wejścia, wyjścia oraz dwukierunkowy. Korzystamy również z magistral wewnętrznych, które bezpośrednio związane są z poleceniem „wire”. Wywołując nasz program korzystamy z instancji, które wywołują poszczególne moduły. Dla każdego portu przewidzieliśmy zapas, jeżeli chodzi o rozmiar sygnału, który będzie nim wysyłany, aby umożliwić testowanie naprawdę dużych liczb.

Przykład – w celach lepszego zrozumienia systemu skalowania rozpisaliśmy swój przykład.

Table 1

Example of Scaling in RNS the Integer X = 2^42-1 by 2^n for the 4-Moduli A and n=8 (2^8 = 256)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Moduli | 255 | 256 | 257 | 511 |
| Original residues | 3 | 255 | 252 | 63 |
| Reverse conv. To obtain X |  | | | |
| Forward conv. – Residues of scaled result Si | S1 = 252 | S2 = 255 | S3 = 3 | S4 = 127 |
| S1 = wzor |  | | | |
| S2 = wzor |  | | | |
| S3 = wzor |  | | | |
| S4 = wzor |  | | | |