SYKOM Projekt

Mateusz Gawlik 324914

Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych

10 maja 2024

Spis treści

1. Zadanie projektowe

Projekt

Politechnika Warszawska, Instytut Telekomunikacji

Prowadzący: Aleksander Pruszkowski

Organizacja projektu:

- Osoba dla której przygotowano ten dokument: Gawlik Mateusz

Przed przystąpieniem do realizacji zadania projektowego należy: Podobnie jak w zajeciach laboratoryjnych, pobrać plik z rozszerzeniem OVPN z serwera WWW `https://resrepo.tele.pw.edu.pl`. Zarezerwowac sobie tzw. wirtualny komputer za pomoca serwera WWW `https://resrepo.tele.pw.edu.pl`. Dla potrzeb zajęć projektowych rezerwacja jest dokonywana wyłącznie w slotach 2h. Proszę pamietać, że liczba rezerwacji dla poszczególnych etapów jest ograniczona choć rezerwować maszyny wirtualne można wielokrotnie w semestrze.
Projekt realizowany jest idywidualnie (grupa jedno osobowa)!

Oczekiwane wyniki - Zgodnie z opisem wprowadzającym `SYKOM_proj.pdf` proszę utworzyć:

- Moduł verilog w pliku: gpioemu.v, który ma realizować operację wyznaczenia N'tej liczby pierwszej - jej numer podawany ma być przez rejestr - argument (A). Podawany argument proszę założyć, że będzie nie większy od 1000. Proszę przyjąć, że moment wpisania wartości do tego rejestru uruchamia automat wyznaczający N'tą liczbę pierwszą.

  Aktualny stan automatu wyznaczającego liczbę pierwszą ma być dostępny przez rejestr - status (S). Samodzielnie zaproponuj jakie wartości powinien on udostępniać w zależoności od aktualnego stanu tego automatu.

  Gdy zadana N'ta liczba pierwsza zostanie znaleziona, jej wartość będzie dostępna w 32 bitowym rejestrze - wynik (W).

  Przyjmij, że na wyprowadzeniu GPIO modułu gpioemu ma pojawiać się liczba wszystkich znalezionych liczb pierwszych od włączenia systemu.

- Pliki źródłowe modułu jądra systemu Linux komunikującego moduł verilogowy gpioemu z aplikacją użytkownika. Przyjmij, że dane przekazywane między aplikacją użytkownika a modułem jądra bedą w tekstowym formacie: HEX.

- Aplikację użytkownika testującą poprawność działania całego systemu w możliwie wielu przypadkach jego użycia. Aplikacja ta ma być podczas testów wbudowana w docelowy filesystem (czyli w plik: rootfs.ext2).

Adresy przestrzeni GPIO i rejestrów udostępnianych przez moduł gpioemu ($SYKT\_GPIO\_ADDR\_SPACE$, A, S, W) a widocznych przez CPU powiny być następujące:
$SYKT\_GPIO\_ADDR\_SPACE$ - ustalony na podstawie konfiguracji wewnętrznej QEMU - zgodnie z opisem dla lab1,
A - wyznaczony jako $SYKT\_GPIO\_ADDR\_SPACE + 0xEC$ (dostępny przez plik: /sys/kernel/sykom/rejAgawmat),
S - wyznaczony jako $SYKT\_GPIO\_ADDR\_SPACE + 0x104$ (dostępny przez plik: /sys/kernel/sykom/rejSgawmat),
W - wyznaczony jako $SYKT\_GPIO\_ADDR\_SPACE + 0xFC$ (dostępny przez plik: /sys/kernel/sykom/rejWgawmat),
Prosze także pamiętać aby w kodzie (modułu jądra i aplikacji testowej) zadbać o sprawdzanie poprawności przekazywanych i zwracanych wartości.

Zawartość raportu: Raport powienien ukazywać na zamieszczonych w nim ilustracjach lub wycinkach raportów wyświetlanych na ekranie QEMU, działanie systemu w różnych a zarazem ważnych(!) i sensownie wybranych chwilach - sensowność doboru tych infromacji także będzie oceniana, jest ona dowodem, że autor jest pewien poprawności działania utworzonego przez siebie systemu i potrafi wskazać na infromacje odpowiadające tej poprawności.
Raport (jako forma sprawozdania) prosimy aby był utworzony w dowolnym edytorze tekstowym, ale po jego przygotowaniu należy raport taki skonwertować do formatu PDF. Żadne inne formaty dokumentów elektornicznych np.: DOC, DOCX, ... nie będą przyjmowane. Fianlnie raport oraz wszelkie pliki źródłowe będące wynikiem prac nad projektem proszę umieść w przydzielonym Tobie indywidualnym repozytorium GIT w jego katalogu projektowym - z tego (i tylko z tego) miejsca prowadzacy będzie pobierał te pliki do późniejszego ocenienia i wystawienia oceny.

Uwaga! Proszę nie umieszczać w przydzielonym repozytorium GIT plików generowanych automatycznie, czyli: qemu-system-riscv32-sykt, Image, rootfs.ext2, natomiast zadbać aby do tego repozytorium trafiały wyłącznie ważne pliki wytworzone przez Ciebie a nie elementy wygenerowane innymi narzędziami.

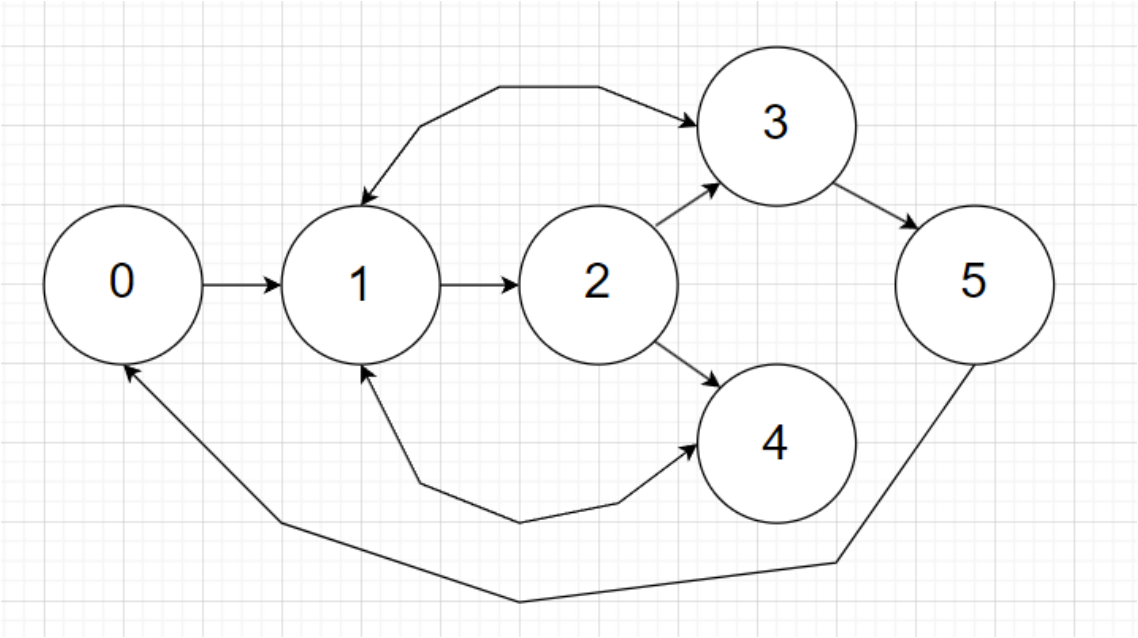
Rys. 1. Zadanie projektowe

2. Moduł verilog

2.1. Stany automatu

Moduł verilog *gpioemu.v* realizujący operacje wyznaczenia N'tej liczby pierwszej bazuje na 6 stanach:

- stan 0 - **INIT**: stan początkowy modułu, w przypadku otrzymania flagi *start* inicjalizuje zmienne i przechodzi do stanu CHECKING_DIV.
- stan 1 - **CHECKING_DIV**: sprawdza czy aktualnie badana pod kątem pierwszości liczba jest równa 2 lub 3 (wtedy przechodzi do stanu PRIME), czy jest podzielna przez 3 (przez 2 nie sprawdzamy, w celach optymalizacji sprawdzamy tylko liczby nieparzyste) oraz, jeśli żadne z poprzednich warunków nie jest prawdziwe, definiuje zakresy poszukiwań dzielników dla sprawdzanej liczby i przechodzi do stanu CHECKING_PRIME.
- stan 2 - **CHECKING_PRIME**: sprawdza czy liczba jest pierwsza, szukając nieparzystych dzielników w zakresie ustalonym w stanie 1. Jeśli taki znajdzie, przechodzi do stanu NOT_PRIME, w przeciwnym wypadku, przechodzi do stanu PRIME.
- stan 3 - **PRIME**: sprawdza, czy znaleziona liczba pierwsza jest N'ta. Jeśli tak, przechodzi do stanu DONE, w przeciwnym wypadku do CHECKING_DIV, sprawdzając kolejną liczbę.
- stan 4 - **NOT_PRIME**: ustawia kolejną liczbę do sprawdzenia i przechodzi do stanu CHECKING_DIV.
- stan 5 - **DONE**: rejestruje wynik do zmiennej W, oraz ustawia flage start na 0.



Rys. 2. Diagram stanów

2.2. Kod

```
/* verilator lint_off UNUSED */
/* verilator lint_off MULTIDRIVEN */
/* verilator lint_off WIDTH */
/* verilator lint_off INITIALDLY */
module gpioemu (
    n_reset,
    saddress[15:0], srd, swr,
    sdata_in[31:0], sdata_out[31:0],
    gpio_in[31:0], gpio_latch,
    gpio_out[31:0],
    clk,
    gpio_in_s_insp[31:0]

);

input clk;
input n_reset;
reg [9:0] A;            // Numer n-tej liczby pierwszej
reg [12:0] W;    // Wynik - N-ta liczba pierwsza
reg [2:0] S;     // Status automatu
reg start;
reg [12:0] divisor;       // Dzielnik liczby - sluzy do sprawdzania czy liczba
↪ pierwsza
reg [12:0] number;        // Liczba ktorej pierwszosc sprawdzamy
reg [9:0] counter;        // Sprawdza ktora dana liczba pierwsza jest w
↪ kolejnosci
reg [3:0] prime_found = 0;    // Liczba znalezionych liczb pierwszych od
↪ wlaczenia systemu

input [15:0] saddress;
input srd;
input swr;
input gpio_latch;
input [31:0] sdata_in;
input [31:0] gpio_in;
output [31:0] gpio_out;
output [31:0] gpio_in_s_insp;
output [31:0] sdata_out;

reg [31:0] sdata_out;
reg [31:0] gpio_in_s;
reg [31:0] gpio_out_s;

// Definicje stanow automatu
localparam INIT = 3'b000;
localparam CHECKING_DIV = 3'b001;
localparam CHECKING_PRIME = 3'b010;
localparam PRIME = 3'b011;
localparam NOT_PRIME = 3'b100;
localparam DONE = 3'b101;

//odpowiedz na reset (aktywowane przejsciem 1->0)
always @(negedge n_reset)
begin
```

```
52      gpio_in_s <= 0;
53      gpio_out_s <= 0;
54      sdata_out <= 0;
55      start <= 0;
56      counter <= 0;
57      number <= 0;
58      divisor <= 0;
59      prime_found <= 0;
60      S <= INIT; // Stan poczatkowy: oczekiwanie na zadanie
61  end
62
63  //Obsluga odczytu z GPIO
64  always @(posedge gpio_latch)
65  begin
66      gpio_in_s <= gpio_in; // Zatrzask w rejestrze wejsciowym
67  end
68
69
70
71  // odczyt z gpio
72  always @(posedge srd)
73  begin
74      // S - 0x104
75      if (saddress == 16'h104)
76          sdata_out <= S;
77      // W - 0xfc
78      else if (saddress == 16'hfc) begin
79          if(S == DONE) begin
80              sdata_out <= (W << 4) + prime_found; //teraz dane sa na [16:4]; na
                ↪  [3:0] liczba znalezionych liczb pierszych od wlaczenia systemu
81          end
82          else
83              sdata_out <= 0;
84      end
85      else
86          sdata_out <= sdata_out;
87  end
88
89  // zapis do gpio
90  always @(posedge swr) begin
91      // A - 0xec
92      if (saddress == 16'hec) begin
93          S <= INIT;
94          A <= sdata_in;
95          start <= 1;
96      end
97  end
98
99
100
101 assign gpio_out = gpio_out_s;
102 assign gpio_in_s_insp = gpio_in_s;
103
104 // Proces odpowiedzialny za automatyczne wyznaczanie liczb pierwszych
105 always @(posedge clk) begin
106     case(S)
```

```
107         INIT: begin
108             if (start == 1 && A > 0 && A <= 1000) begin
109                 counter <= 0;
110                 number <= 1;
111                 divisor <= 5;
112                 S <= CHECKING_DIV; // Przejdz do stanu wyszukiwania
113             end
114         end
115         CHECKING_DIV: begin
116             // Sprawdzanie podzielnosci przez 2 i 3 aktualnej liczby
117             if (number == 2 || number == 3) begin
118                 counter <= counter + 1;
119                 S <= PRIME;
120             end else if (number % 3 == 0 || number == 1) begin
121                 S <= NOT_PRIME;
122             end else begin
123                 if( (number >> 2) < 5) begin
124                     counter <= counter + 1;
125                     S <= PRIME;
126                 end else begin
127                     S <= CHECKING_PRIME;
128                 end
129             end
130         end
131         CHECKING_PRIME: begin
132             if (number % divisor == 0) begin
133                 S <= NOT_PRIME;
134             end else begin
135                 if (divisor * divisor <= number) begin
136                     divisor <= divisor + 2;
137                 end else begin
138                     counter <= counter + 1;
139                     S <= PRIME;
140                 end
141             end
142         end
143 
144         PRIME: begin
145             //counter <= counter + 1;
146             if (counter == A) begin
147                 prime_found <= prime_found + 1;
148                 S <= DONE;
149             end else begin
150                 if (number == 2) begin
151                     number <= number + 1;
152                 end else begin
153                     number <= number + 2;
154                 end
155                 divisor <= 5;
156                 S <= CHECKING_DIV;
157             end
158         end
159         NOT_PRIME: begin
160             if (number == 1) begin
161                 number <= number + 1;
162             end else begin
```

```
163                 number <= number + 2;
164             end
165             divisor <= 5;
166             S <= CHECKING_DIV;
167         end
168         DONE: begin
169             W <= number;
170             start <= 0;
171         end
172
173         default: S <= INIT;
174     endcase
175 end
176 endmodule
```

Program uruchamia obliczenia wraz z podaniem wartości na wejscie sdata_in pod odpowiedni adres (saddress) i po podaniu flagi swr. Stan maszyny, wynik obliczeń, jak i numer znalezionej liczby od początku włączenia systemu jest dostępny przez rejestr sdata_out, po podaniu odpowiedniego adresu oraz flagi srd.

2.3. Test modułu

Stworzyłem testbencha który przetestował moduł w różnych sytuacjach:
1)Zapis liczby pod zły adres + flaga swr nie daje żadnych rezultatów, ponieważ input nie zostaje wprowadzony.

```
PS D:\iverilog\bin> .\vvp.exe PrimeNumberGenerator.vvp
VCD info: dumpfile PrimeNumberGenerator.vcd opened for output.
At                   0, sdata_in =xxxxxxxx, sdata_out = xxxxxxxx, saddress = xxxx
At                 100, sdata_in =xxxxxxxx, sdata_out = 00000000, saddress = xxxx
At                 150, sdata_in =xxxxxxxx, sdata_out = 00000000, saddress = 00ff
At                 200, sdata_in =00000002, sdata_out = 00000000, saddress = 00ff
At                4300, sdata_in =00000002, sdata_out = 00000000, saddress = 00fc
```

Rys. 3. Test zapisu pod zły adres - brak wyniku (program nie zaczyna liczyć)

2)Zapis liczby pod odpowiedni adres + flaga swr, zaczyna obliczenia i podaje wynik w formacie: pierwsze 4 bity - numer znalezionej liczby pierwszej od włączenia systemu, następne 13 bitów - n'ta liczba pierwsza.
Input: 0x1 = decymalnie 1
Output: 0x21 - wynik decymalnie 2; pierwsza znaleziona liczba pierwsza

```
At                4450, sdata_in =00000002, sdata_out = 00000000, saddress = 00ec
At                4500, sdata_in =00000001, sdata_out = 00000000, saddress = 00ec
At                8600, sdata_in =00000001, sdata_out = 00000000, saddress = 00fc
At                8650, sdata_in =00000001, sdata_out = 00000021, saddress = 00fc
```

Rys. 4. Test zapisu pod odpowiedni adres - poprawny wynik na wyjściu

3)Poniższy test pokazuje, że odczyt statusu spod złego adresu nie daje rezultatów na wyjściu, oraz zapis pod odpowiedni adres daje poprawny wynik.
Input: 0x3e8 = decymalnie 1000
Output: 0x1eef2 = wynik decymalnie 7919; druga znaleziona liczba pierwsza

```
At                8800, sdata_in =000003e8, sdata_out = 00000021, saddress = 00ec
At                9400, sdata_in =000003e8, sdata_out = 00000021, saddress = 0105
At            16009500, sdata_in =000003e8, sdata_out = 00000021, saddress = 00fc
At            16009550, sdata_in =000003e8, sdata_out = 0001eef2, saddress = 00fc
```

Rys. 5. Odczyt statusu spod złego adresu - brak rezultatów na wyjściu, zapis wartości i odczyt wyniku spod dobrego adresu - poprawny rezultat

4)Ostatni test przedstawia poprawny odczyt statusu oraz wynik działania programu po wprowadzeniu na input liczby 250, ustawiając każdy adres poprawnie.
Input: 0xfa = decymalnie 250
Output: 0x62f3 = wynik decymalnie 1583; trzecia znaleziona liczba pierwsza

```
At            16010650, sdata_in =000003e8, sdata_out = 0001eef2, saddress = 00ec
At            16010700, sdata_in =000000fa, sdata_out = 0001eef2, saddress = 00ec
At            16011300, sdata_in =000000fa, sdata_out = 0001eef2, saddress = 0104
At            16011350, sdata_in =000000fa, sdata_out = 00000003, saddress = 0104
At            17511400, sdata_in =000000fa, sdata_out = 00000003, saddress = 00fc
At            17511450, sdata_in =000000fa, sdata_out = 000062f3, saddress = 00fc
```

Rys. 6. Poprawny odczyt statusu oraz poprawny wynik obliczeń

Wszystkie testy przebiegły zgodnie z oczekiwaniami.
Poniżej znajduje się kod testbencha.

```
1  `timescale 1ns/100ps
2
3  module gpioemu_tb;
4      // Deklaracja sygnalow
5      reg clk = 1;
6      reg start;
7      reg [9:0] A;
8      wire [31:0] W;
9      wire [2:0] S;
10     reg [12:0] divisor;
11     reg [12:0] number;
12     reg [9:0] counter;
13
14     reg         n_reset;
15     reg [15:0]  saddress;
16     reg         srd;
17     reg         swr;
18     reg [31:0]  sdata_in;
19     reg [31:0]  gpio_in;
```

```
20      reg         gpio_latch;
21      wire [31:0] gpio_out;
22      wire [31:0] sdata_out;
23      wire [31:0] gpio_in_s_insp;
24      reg [3:0] prime_found;
25
26
27      // Instancja testowanego modulu
28
29      initial begin
30                      $dumpfile("PrimeNumberGenerator.vcd");
31                      $dumpvars(0, PrimeNumberGenerator_tb);
32      end
33
34      initial begin
35                  forever begin
36                      # 5 clk = ~clk;
37                      end
38      end
39
40
41
42      // Inicjalizacja testu
43      initial begin
44          //Test 1
45          // Test swr
46          #5 n_reset = 1;
47          #5 n_reset = 0;
48
49          // Zapis pod zły adres
50          #5 saddress = 16'h0ff;
51          #5 sdata_in = 2;
52          #5 swr = 1;
53          #5 swr = 0;
54          #400 saddress = 16'h0fc;
55          #5 srd = 1;
56          #5 srd = 0;
57          // Zapis pod dobry adres tej samej liczby
58          #5 saddress = 16'h0ec;
59          #5 sdata_in = 1;
60          #5 swr = 1;
61          #5 swr = 0;
62          #400 saddress = 16'h0fc;
63          #5 srd = 1;
64          #5 srd = 0;
65          // Zapis pod dobry adres
66          #5 saddress = 16'h0ec;
67          #5 sdata_in = 1000;
68          #5 swr = 1;
69          #5 swr = 0;
70
71          // Odczyt statusu ze zlego adresu
72          #50 saddress = 16'h105;
73          #5 srd = 1;
74          #5 srd = 0;
75          // Odczyt wyniku
```

```
76          #1600000
77          saddress = 16'h0fc;
78          #5 srd = 1;
79          #5 srd = 0;
80          #100
81
82          //Test 2
83          // Test swr
84
85          #5 saddress = 16'h0ec;
86          #5 sdata_in = 250;
87          #5 swr = 1;
88          #5 swr = 0;
89          // Odczyt statusu
90          #50 saddress = 16'h104;
91          #5 srd = 1;
92          #5 srd = 0;
93          // Odczyt wyniku
94          #150000
95          saddress = 16'h0fc;
96          #5 srd = 1;
97          #5 srd = 0;
98
99          $finish;
100     end
101 PrimeNumberGenerator uut (
102         n_reset,
103     saddress, srd, swr,
104     sdata_in, sdata_out,
105     gpio_in, gpio_latch,
106     gpio_out,
107     clk,
108     gpio_in_s_insp
109     );
110
111 initial begin
112                 $monitor("At %t, sdata_in =%h, sdata_out = %h, saddress = %h",
                    ↪  $time, sdata_in, sdata_out, saddress);
113         end
114
115 endmodule
116
```

```
sykt@deb4sykom10:~/Gawlik_Mateusz/projekt$ makeQemuGpioEmu gpioemu.v
Checking enviroment...
Checking was done.
Compiling the Verilog file (/home/sykt/Gawlik_Mateusz/projekt/gpioemu.v) into CPP by Verilator...
Compiling the Verilog product in CPP format into BIN...
Compiling the wraper file for gpioemu extension...
Making libgpioemu library...
Libgpioemu is ready in: /home/sykt/Gawlik_Mateusz/projekt/libgpioemu.a
Linking the /var/local/qemu/riscv32-softmmu to final executable file with name qemu-system-riscv32-sykt...
The 'qemu-system-riscv32-sykt' is placed at '/home/sykt/Gawlik_Mateusz/projekt' location. Bye!
```

Rys. 7. Pomyślne zbudowanie emulatora QEMU

3. Moduł jądra

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/ioport.h>
#include <linux/kobject.h>
#include <asm/errno.h>
#include <asm/io.h>

MODULE_INFO(intree, "Y");
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Aleksander Pruszkowski");
MODULE_DESCRIPTION("Simple kernel module for SYKOM lecture");
MODULE_VERSION("0.01");

#define SYKT_GPIO_BASE_ADDR  (0x00100000)
#define SYKT_GPIO_SIZE       (0x8000)
#define SYKT_EXIT            (0x3333)
#define SYKT_EXIT_CODE       (0x7F)

#define SYKT_GPIO_ADDR_SPACE (SYKT_GPIO_BASE_ADDR)
#define SYKT_GPIO_A_ADDR     (SYKT_GPIO_ADDR_SPACE + 0xEC)
#define SYKT_GPIO_S_ADDR     (SYKT_GPIO_ADDR_SPACE + 0x104)
#define SYKT_GPIO_W_ADDR     (SYKT_GPIO_ADDR_SPACE + 0xFC)

// Deklaracja wskaźników na obszary pamięci
void __iomem *baseptr;
void __iomem *baseptrA;
void __iomem *baseptrW;
void __iomem *baseptrS;

static struct kobject *kobj_ref;
static int rejAgawmat;
static int rejSgawmat;
static int rejWgawmat;

// Deklaracje funkcji
static ssize_t rejAgawmat_store(struct kobject *kobj, struct kobj_attribute *attr,const char *buf,
size_t count);
static ssize_t rejAgawmat_show(struct kobject *kobj, struct kobj_attribute *attr, char *buf);
static ssize_t rejWgawmat_show(struct kobject *kobj, struct kobj_attribute *attr, char *buf);
static ssize_t rejSgawmat_show(struct kobject *kobj, struct kobj_attribute *attr, char *buf);

// ================= funkcje do komunikacji ==========================================
// odczyt argumentu A i zapis na odpowiednie miejsce w pamieci
static ssize_t rejAgawmat_store(struct kobject *kobj, struct kobj_attribute *attr,const char *buf,
size_t count)
{
        sscanf(buf,"%x",&rejAgawmat);
        writel(rejAgawmat, baseptrA);
        return count;
}
// odczyt inputa z modulu
static ssize_t rejAgawmat_show(struct kobject *kobj, struct kobj_attribute *attr, char *buf)
{
    return sprintf(buf, "%x", rejAgawmat);
```

```
}

// odczyt wyniku z modulu
static ssize_t rejWgawmat_show(struct kobject *kobj, struct kobj_attribute *attr, char *buf)
{
        rejWgawmat = readl(baseptrW);
    return sprintf(buf, "%x", rejWgawmat);
}
// odczyt statusu (czy modul skonczyl dzialanie)
static ssize_t rejSgawmat_show(struct kobject *kobj,
                struct kobj_attribute *attr, char *buf)
{
        rejSgawmat = readl(baseptrS);
    return sprintf(buf, "%x", rejSgawmat);
}

// Definicje atrybutów

static struct kobj_attribute rejAgawmat_attr = __ATTR_RW(rejAgawmat);
static struct kobj_attribute rejWgawmat_attr = __ATTR_RO(rejWgawmat);
static struct kobj_attribute rejSgawmat_attr = __ATTR_RO(rejSgawmat);

int my_init_module(void){
    printk(KERN_INFO "Init my module.\n");
    // Inicjalizacja obszarów pamięci
    baseptr = ioremap(SYKT_GPIO_BASE_ADDR, SYKT_GPIO_SIZE);
    baseptrA = ioremap(SYKT_GPIO_A_ADDR, 2);
    baseptrW = ioremap(SYKT_GPIO_W_ADDR, 2);
    baseptrS = ioremap(SYKT_GPIO_S_ADDR, 1);
    kobj_ref = kobject_create_and_add("sykom",kernel_kobj);
     if (!kobj_ref) {
        printk(KERN_INFO "Failed to create sysfs directory \"sykom\".\n");
        }

        if(sysfs_create_file(kobj_ref,&rejAgawmat_attr.attr)){
            printk(KERN_INFO"Cannot create sysfs file......\n");
        kobject_put(kobj_ref);
        }

        if(sysfs_create_file(kobj_ref,&rejWgawmat_attr.attr)){
            printk(KERN_INFO"Cannot create sysfs file......\n");
                sysfs_remove_file(kobj_ref, &rejAgawmat_attr.attr);
        kobject_put(kobj_ref);
        }

        if(sysfs_create_file(kobj_ref,&rejSgawmat_attr.attr)){
            printk(KERN_INFO"Cannot create sysfs file......\n");
                sysfs_remove_file(kobj_ref, &rejAgawmat_attr.attr);
                sysfs_remove_file(kobj_ref, &rejWgawmat_attr.attr);
        kobject_put(kobj_ref);
        }
    return 0;
}

void my_cleanup_module(void){
    printk(KERN_INFO "Cleanup my module.\n");
```

```
    writel(SYKT_EXIT | ((SYKT_EXIT_CODE)<<16), baseptr);
    kobject_put(kobj_ref);
    sysfs_remove_file(kobj_ref, &rejAgawmat_attr.attr);
    sysfs_remove_file(kobj_ref, &rejWgawmat_attr.attr);
    sysfs_remove_file(kobj_ref, &rejSgawmat_attr.attr);
    iounmap(baseptr);
    iounmap(baseptrA);
    iounmap(baseptrW);
    iounmap(baseptrS);
}

module_init(my_init_module)
module_exit(my_cleanup_module)
```

Zadeklarowane zostały wskaźniki na odpowiednie obszary w pamięci, zgodnie z przydzieloną w instrukcji adresacją. Stworzony został wirtualny system plików SysFs, w którym umieszczone są pliki służące do komunikacji z modułem Verilog. Funkcje komunikacyjne obejmują:

- odczyt i zapis z rejestru rejAgawmat,
- odczyt z rejestru rejSgawmat,
- odczyt z rejestru rejWgawmat.

Kompilacja przebiegła pomyślnie i oczekiwany system plików został utworzony.

```
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Saving random seed: [    2.556641] random: dd: uninitialized urandom read (512 bytes read)
OK
Starting network: OK

Welcome to Buildroot
buildroot login: root
# modprobe kernel_module
[   11.704136] Init my module.
# cd /sys/kernel[   21.727827] random: fast init done
^C
# cd /sys/kernel/sykom
# pwd
/sys/kernel/sykom
# ls
rejAgawmat  rejSgawmat  rejWgawmat
```

Rys. 8. Pomyślne załadowanie modułu jądra, utworzony system plików SysFs

```
# ./main
Compiled at Feb 23 2023 13:39:01
```

Rys. 9. Uruchomienie domyślnej aplikacji testowej

4. Testowanie z wykorzystaniem własnej aplikacji

4.1. Kod aplikacji

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

#define MAX_BUFFER 1024
#define SYSFS_FILE_A_IN "/sys/kernel/sykom/rejAgawmat"
#define SYSFS_FILE_S_OUT "/sys/kernel/sykom/rejSgawmat"
#define SYSFS_FILE_W_OUT "/sys/kernel/sykom/rejWgawmat"


unsigned int read_from_file(char *filePath){
    char buffer[MAX_BUFFER];
    int fd_out = open(filePath, O_RDONLY);
    if(fd_out < 0){
        printf("Open %s - error number %d\n", filePath, errno);
        exit(5);
    }
    lseek(fd_out, 0L, SEEK_SET);
    int n = read(fd_out, buffer, MAX_BUFFER);
    if(n>0) {
        buffer[n]='\0';
        close(fd_out);
        return strtoul(buffer, NULL, 16);
    }
}

void write_to_file(char *filePath, unsigned int value) {
    char buffer[MAX_BUFFER];

    int fd_in = open(filePath, O_RDWR | O_TRUNC, 0644);
    if(fd_in < 0) {
        printf("Open %s - error number %d\n", filePath, errno);
        exit(5);
    }
    snprintf(buffer, MAX_BUFFER, "%x", value);
    lseek(fd_in, 0L, SEEK_SET);
    ssize_t bytes_written = write(fd_in, buffer, strlen(buffer));
    if(bytes_written < 0) {
        printf("Write to %s - error number %d\n", filePath, errno);
        exit(6);
    }
    close(fd_in);
}

unsigned int get_prime(unsigned int a){
    write_to_file(SYSFS_FILE_A_IN, a);
    unsigned int read_status;
```

```
    unsigned int read;
    do{
        read_status = read_from_file(SYSFS_FILE_S_OUT);
    }
    while(read_status != 5);
    sleep(1);
    read = read_from_file(SYSFS_FILE_W_OUT);
    return read;
}

int test_module(){
    unsigned int res;
    unsigned int W;
    unsigned int number_found;
    unsigned int a[6] =       {1, 2, 3, 10, 20, 700};
    unsigned int results[6] = {2, 3, 5, 29, 71, 5279};
    int i = 0;
    while (i < 6) {
        res = get_prime(a[i]);
        W = res >> 4;
        number_found = (res & 0x0000000F);
        printf("A = 0x%x, W = 0x%x, Expected = 0x%x, found numbers so far: 0x%x\n", a[i], W,
        results[i], number_found);
        fflush(stdout);
        i++;
    }

    return 0;
}


int main(void){
    int test = test_module();
    if(test > 0){
        printf("TEST FAILED at %d values\n", test);
    } else {
        printf("====== TEST PASSED =====\n");
    }
    return 0;
}
```

Utworzony został wektor danych wejściowych, które po kolei wprowadzane są do modułu poprzez plik */sys/kernel/sykom/rejAgawmat* (poprzez funkcję write_to_file). Wynik pobierany jest z pliku */sys/kernel/sykom/rejWgawmat* (poprzez funkcję read_from_file) w momencie, gdy status modułu pobierany z pliku */sys/kernel/sykom/rejSgawmat* wynosi 5 (DONE). Rezultaty wraz z wartością oczekiwaną wyświetlane są na konsoli w postaci heksadecymalnej.

4.2. Wyniki

Po kompilacji modułu jądra i aplikacji *main.c* przy pomocy narzędzia *make_busybox_compile* (rys. 10), wystarczy uruchomić aplikacje juz w systemie Linux, aby wyniki zostały wyświetlone na konsoli (rys. 11).

```
sykt@deb4sykom10:~/Gawlik_Mateusz/projekt$ make_busybox_compile main.c
Changing place...
Checking contents of source files...
Removing previous compilation...
Preparing new compilation...
Module compilation in progress...
mke2fs 1.45.6 (20-Mar-2020)
Module compilation done.
Test application compilation...
Compiling the /home/sykt/Gawlik_Mateusz/projekt/main.c file...
Install /home/sykt/Gawlik_Mateusz/projekt/main file on destionation FS...
RootFS re-compilation in progress...
mke2fs 1.45.6 (20-Mar-2020)
RootFS re-compilation done (the '/home/sykt/Gawlik_Mateusz/projekt/main' - product of compilation process is placed in /root of the target FS).
Done, files: fw_jump.elf, Image, rootfs.ext2 are ready to use.
sykt@deb4sykom10:~/Gawlik_Mateusz/projekt$
```

Rys. 10. Kompilacja modułu jądra i aplikacji

```
# ./main
A = 0x1, W = 0x2, Expected = 0x2, found numbers so far: 0x1
A = 0x2, W = 0x3, Expected = 0x3, found numbers so far: 0x2
A = 0x3, W = 0x5, Expected = 0x5, found numbers so far: 0x3
A = 0xa, W = 0x1d, Expected = 0x1d, found numbers so far: 0x4
A = 0x14, W = 0x47, Expected = 0x47, found numbers so far: 0x5
A = 0x2bc, W = 0x149f, Expected = 0x149f, found numbers so far: 0x6
====== TEST PASSED =====
# ./main
A = 0x1, W = 0x2, Expected = 0x2, found numbers so far: 0x7
A = 0x2, W = 0x3, Expected = 0x3, found numbers so far: 0x8
A = 0x3, W = 0x5, Expected = 0x5, found numbers so far: 0x9
A = 0xa, W = 0x1d, Expected = 0x1d, found numbers so far: 0xa
A = 0x14, W = 0x47, Expected = 0x47, found numbers so far: 0xb
```

Rys. 11. Uruchomienie własnej aplikacji testowej z poziomu systemu Linux; uruchomienie za drugim razem w celu ukazania inkrementacji ilości znalezionych liczb pierwszych

Wyniki jakie otrzymujemy, są zgodne z oczekiwaniami, co zostało potwierdzone przez porównanie wartości oczekiwanych (Expected) z faktycznie znalezionymi n-tymi liczbami pierwszymi (W), jak przedstawiono na rysunku 11. Ponadto obserwujemy, że numer kolejnej znalezionej liczby od początku działania systemu iteruje się poprawnie. Wyniki te świadczą o poprawności obliczeń modułu verilogowego oraz o prawidłowej komunikacji między nim a jądrem systemu.