AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**DZIEDZINA NAUK INŻYNIERYJNO-TECHNICZNYCH**

DYSCYPLINA AUTOMATYKA, ELEKTRONIKA, ELEKTROTECHNIKA
I TECHNOLOGIE KOSMICZNE

# ROZPRAWA DOKTORSKA

## Rozwój scalonych systemów mikroprocesorowych dla hybrydowych detektorów promieniowania jonizującego

Autor: mgr inż. Paweł Skrzypiec

Promotor rozprawy: dr hab. inż. Robert Szczygieł, prof. AGH

Praca wykonana: Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Kraków, 2023

*Składam serdeczne podziękowania mojemu promotorowi, Panu Profesorowi Robertowi Szczygłowi, za okazywaną życzliwość, opiekę i pomoc w realizacji niniejszej rozprawy.*

*Serdecznie dziękuję Współpracownikom z Grupy Mikroelektronicznej KMiE, AGH, a w szczególności Panu Profesorowi Pawłowi Grybosiowi, Panu Profesorowi Mirosławowi Żołędziowi i Pani Doktor Weronice Zubrzyckiej, za liczne sugestie, rady i cenne wskazówki.*

*Dziękuję również moim rodzicom, Krystynie i Adamowi, oraz Bratu, Miłoszowi, za Ich obecność w moim życiu, okazywane uczucie oraz pomoc w realizacji marzeń.*

# Abstract

The main aim of the work presented in this thesis was the implementation of the advantages of edge computing in hybrid pixel detectors. The research conducted by the Author has contributed to the design and testing of the application-specific integrated circuit, which integrated RISC-V microprocessor and readout system for the hybrid pixel detector, in the common silicon substrate. The developed solution enabled the on-chip execution of algorithms that previously were executed only using assistive devices, such as PCs and FPGAs. An exemplary benefit resulting from on-chip data processing is the reduction of the time required for transmission of radiation detection results to a storing device, which has been achieved by the on-chip execution of data filtering algorithms.

The application-specific integrated circuit designed by the Author of the dissertation was manufactured in a 40 nm CMOS process and tested using a dedicated test environment. The test environment was constructed with PC and FPGA connected via PCIe Gen. 3 x8 interface, and was composed of a digital circuit implemented in FPGA, linux device driver and user space application, enabling tested device programming, commands transmission, and the visualization of data received from the tested integrated circuit.

The constructed test environment enabled the implementation and verification of the calibration procedure, allowing compensation of the offset voltages of the discriminators included in the analog read-out channels. Experiments carried out confirmed that the integration of the RISC-V microprocessor and hybrid pixel detector in a common silicon substrate enables the construction of autonomous readout systems, capable of on-chip analysis of the detector data and autocalibration. According to the Author's best knowledge, the presented integrated circuit is the first chip in the world that integrates the RISC-V microprocessor and a hybrid pixel detector in a common silicon substrate.

# Streszczenie

Głównym celem prac przedstawionych w niniejszej rozprawie było wykorzystanie zalet przetwarzania brzegowego w hybrydowych detektorach promieniowania jonizującego. Badania zrealizowane przez Autora przyczyniły się do zaprojektowania i przetestowania specjalizowanego układu scalonego, integrującego we wspólnym podłożu krzemowym mikroprocesor RISC-V i układ odczytowy dla hybrydowego pikselowego detektora promieniowania jonizującego. Opracowane rozwiązanie pozwoliło na wykonywanie wewnątrz układu scalonego algorytmów, które dotychczas realizowane były przy użyciu zewnętrznych urządzeń pomocniczych, takich jak komputery osobiste i układy FPGA. Przykładowym benefitem wynikającym z przetwarzania danych wewnątrz układu scalonego jest redukcja czasu potrzebnego na transmisję wyników rejestracji promieniowania do urządzenia odpowiedzialnego za ich magazynowanie, uzyskiwana poprzez wykonywanie wewnątrz układu odpowiednich algorytmów filtracyjnych.

Zaprojektowany przez Autora niniejszej rozprawy scalony system mikroprocesorowy dla hybrydowych detektorów promieniowania jonizującego został wyprodukowany w 40-nanometrowym procesie CMOS i przetestowany przez środowisko testowe skonstruowane przez Autora. Opracowane środowisko testowe zostało zrealizowane przy użyciu komputera PC i układu FPGA, połączonych za pośrednictwem interfejsu PCIe Gen. 3 x8, a w jego skład wchodziły układ cyfrowy zaimplementowany w FPGA, sterownik dla Linuksa (moduł jądra) oraz aplikacja przestrzeni użytkownika, umożliwiająca, między innymi programowanie testowanego układu, przesyłanie do niego komend i wizualizację danych transmitowanych przez ten układ.

Skonstruowane środowisko testowe pozwoliło Autorowi niniejszej rozprawy na implementację i przetestowanie algorytmu kalibracyjnego, umożliwiającego korekcję napięć niezrównoważenia dyskryminatorów wchodzących w skład analogowych torów odczytowych. Przeprowadzone eksperymenty wykazały, że integracja we wspólnym podłożu krzemowym mikroprocesora RISC-V i układu odczytowego dla hybrydowego pikselowego detektora promieniowania jonizującego, umożliwia budowę autonomicznych układów odczytowych, zdolnych do samodzielnej analizy danych odczytywanych z sensorów i autokalibracji. Przedstawiony w niniejszej rozprawie scalony system mikroprocesorowy jest, według najlepszej wiedzy Autora, pierwszym na świecie układem integrującym we wspólnym podłożu krzemowym mikroprocesor RISC-V i układ odczytowy dla hybrydowego pikselowego detektora promieniowania jonizującego.

# Spis treści

# Spis symboli i skrótów

| | |
|---|---|
| **ALU** | Arithmetic Logic Unit |
| **API** | Application Programming Interface |
| **ASIC** | Application-Specific Integrated Circuit |
| **Avalon-MM** | Avalon Memory Mapped Interface |
| **Avalon-ST** | Avalon Streaming Interface |
| **BAR** | Base Address Register |
| **CISC** | Complex Instruction Set Computing |
| **CMOS** | Complementary Metal-Oxide-Semiconductor |
| **CPU** | Central Processing Unit |
| **CSR** | Control and Status Register |
| **CXL** | Compute Express Link |
| **CU** | Control Unit |
| **DDR** | Double Data Rate |
| **DSC** | Data Storage Controller |
| **FMC** | FPGA Mezzanine Card |
| **FPGA** | Field Programmable Gate Array |
| **GPIO** | General Purpose Input Output |
| **GUI** | Graphical User Interface |
| **HDMI** | High Definition Multimedia Interface |
| **HDL** | Hardware Description Language |
| **HPD** | Hybrid Pixel Detector |
| **IDU** | Instruction Decode Unit |
| **IFU** | Instruction Fetch Unit |
| **ISA** | Instruction Set Architecture |
| **IOMUX** | Input Output Multiplexer |

| | |
|---|---|
| **I/O** | Input/Output |
| **IP** | Intellectual Property |
| **LED** | Light-Emitting Diode |
| **LSU** | Load-Store Unit |
| **LVDS** | Low-Voltage Differential Signaling |
| **MISO** | Master Input, Slave Output |
| **MOS** | Metal-Oxide-Semiconductor |
| **MOSI** | Master Output, Slave Input |
| **MOSFET** | Metal-Oxide-Semiconductor Field-Effect Transistor |
| **NMOS** | N-type Metal-Oxide-Semiconductor |
| **OCM** | On-Chip Memory |
| **PC** | Personal Computer |
| **PCIe** | Peripheral Component Interconnect Express |
| **PM** | Pixel Matrix |
| **PMC** | Pixel Matrix Controller |
| **PMCC** | Pixel Matrix Controller Coprocessor |
| **PMDC** | Pixel Matrix Data Converter |
| **RAM** | Random-Access Memory |
| **RISC** | Reduced Instruction Set Computing |
| **RF** | Register File |
| **ROM** | Read-Only Memory |
| **SCK** | Serial Clock |
| **SD** | Secure Digital |
| **SLVS** | Scalable Low-Voltage Signaling |
| **SoC** | System on Chip |
| **SPI** | Serial Peripheral Interface |
| **SS** | Slave Select |
| **SSD** | Solid State Drive |
| **TX** | Transceiver |
| **UART** | Universal Asynchronous Receiver / Transmitter |

# Rozdział 1

# Wstęp

## 1.1 Wprowadzenie

*Mikroprocesory* są scalonymi układami elektronicznymi, których działanie może być modyfikowane bez konieczności wprowadzania zmian w ich architekturze sprzętowej. Zadania realizowane przez tego typu urządzenia definiowane są za pośrednictwem instrukcji, czyli ciągów liczb przenoszących informacje o typie i argumentach żądanej operacji. Wynikająca z tej właściwości elastyczność mikroprocesorów sprawiła, że układy te znalazły szerokie zastosowanie nie tylko w centrach danych i superkomputerach, ale również w niewielkich, energooszczędnych urządzeniach, takich jak telefony komórkowe i inteligentne zegarki.

Mianem *systemów mikroprocesorowych* określane są układy elektroniczne, w skład których wchodzą mikroprocesory, pamięci oraz inne urządzenia peryferyjne, umożliwiające mikroprocesorom interakcję z otoczeniem. Przykładami tego typu systemów są mikrokontrolery, czyli urządzenia integrujące wspomniane komponenty w jednym układzie scalonym. Na rynku znajdują się zarówno mikrokontrolery ogólnego zastosowania, jak również układy specjalizowane do specyficznych zadań, na przykład w eksperymentach naukowych i medycynie. Mikrokontrolery dedykowane do konkretnych aplikacji mogą zawierać peryferia obsługujące specjalizowane czujniki.

Integracja mikroprocesorów ze specjalizowanymi czujnikami pozwala na przetwarzanie danych generowanych przez te sensory blisko miejsca ich wytworzenia. Skutkuje to eliminacją konieczności przesyłania danych do zewnętrznego urządzenia pomocniczego, odpowiedzialnego wyłącznie za ich przetworzenie. Umieszczenie czujnika i mikroprocesora w jednym układzie scalonym pozwala na skrócenie czasu potrzebnego na analizę danych i ewentualną korektę parametrów sensora, oraz na zmniejszenie ilości danych transmitowanych do systemu nadrzędnego. Realizowane w ten sposób przetwarzanie danych implementuje współczesny paradygmat przetwarzania brzegowego (ang. edge-computing), zwiększając ogólną wydajność systemu i zmniejszając jego zapotrzebowanie energetyczne.

Mimo licznych zalet wynikających z możliwości programowania układów elektronicznych, niektóre systemy realizowane są jako układy scalone o stałej funkcjonalności. Przykładami tego typu układów są hybrydowe pikselowe detektory promieniowania jonizującego. Składają się one z dwóch podstawowych komponentów: półprzewodnikowego sensora, konwertującego promieniowanie na impulsy prądowe, oraz układu elektronicznego, przetwarzającego te impulsy. Wstępne przetwarzanie impulsów jest zazwyczaj re-

alizowane przez specjalizowany układ scalony o stałej funkcjonalności. Następnie surowe dane przesyłane są do systemu nadrzędnego, dokonującego ich finalnej obróbki. Do zadań systemu nadrzędnego, poza odczytem wyników rejestracji promieniowania, należy również kalibracja detektora.

Celem opisanych w rozprawie prac jest wykorzystanie zalet przetwarzania brzegowego w hybrydowych detektorach promieniowania jonizującego. Przedstawiony cel zostanie osiągnięty poprzez realizację następujących zadań:

1. opracowanie scalonego układu odczytowego do hybrydowego pikselowego detektora promieniowania jonizującego, z wbudowanym mikroprocesorem, który rozszerzy możliwości istniejących rozwiązań,
2. opracowanie oprogramowania wymaganego do poprawnego działania skonstruowanego systemu,
3. opracowanie systemu (sprzętu i oprogramowania) do testowania, który:
    - umożliwi sprawne przetestowanie skonstruowanego układu po jego wyprodukowaniu,
    - pozwoli na weryfikację części cyfrowej projektowanego układu już na wstępnym etapie prac projektowych.

Badania zrealizowane przez Autora niniejszej rozprawy przyczyniły się do zaprojektowania, wyprodukowania i przetestowania specjalizowanego układu scalonego w technologii CMOS 40 nm, integrującego we wspólnym podłożu krzemowym mikroprocesor RISC-V i układ odczytowy dla hybrydowego pikselowego detektora promieniowania jonizującego. Opracowana metodologia projektowa oparta została na regresyjnych testach symulacyjnych i implementacji w FPGA prototypu projektowanego układu scalonego, dzięki czemu możliwe było przetestowanie najważniejszych funkcjonalności proponowanego rozwiązania jeszcze na etapie projektowania urządzenia. Skonstruowane środowisko testowe potwierdziło, iż zaprojektowany przez Autora niniejszej rozprawy scalony system mikroprocesorowy umożliwił wykonywanie wewnątrz układu scalonego analizy danych odczytywanych z matrycy pikseli i sterowanie detekcją promieniowania za pośrednictwem zrozumiałych dla człowieka komend tekstowych. Zaprojektowany przez Autora niniejszej rozprawy specjalizowany układ scalony, według najlepszej wiedzy Autora, jest pierwszym na świecie układem scalonym, integrującym we wspólnym podłożu krzemowym hybrydowy detektor promieniowania i mikroprocesor RISC-V.

## 1.2 Struktura pracy

Niniejsza rozprawa podzielona została na pięć rozdziałów przedstawiających wyniki badań przeprowadzonych przez Autora:

**Rozdział 1** zawiera wprowadzenie do tematyki omawianej w niniejszej rozprawie, przedstawia motywację Autora i prezentuje sformułowane przez niego tezy.

**Rozdział 2** obejmuje wprowadzenie teoretyczne do zagadnień prezentowanych w niniejszej pracy. W rozdziale tym zamieszczone zostały podstawowe informacje dotyczące budowy systemów mikroprocesorowych i architektury RISC-V oraz przybliżone zostały budowa i działanie hybrydowych detektorów promieniowania jonizującego.

**Rozdział 3** poświęcony został systemowi mikroprocesorowemu zaprojektowanemu przez Autora niniejszej rozprawy. W rozdziale tym przedstawione zostały architektura opracowanego układu, wyniki przeprowadzonych symulacji oraz opis mikroprocesora RISC-V zaprojektowanego przez Autora.

**Rozdział 4** przedstawia opracowane przez Autora niniejszej rozprawy środowisko do testowania specjalizowanych układów scalonych. W rozdziale tym zaprezentowane zostały architektura skonstruowanego środowiska oraz wyniki testów układu scalonego zaprojektowanego przez Autora.

**Rozdział 5** zawiera podsumowanie i plany dotyczące dalszych badań.

**Dodatek A** zawiera wysokiej rozdzielczości obrazy, prezentujące plan masek zaprojektowanego układu scalonego, fotografię mikroskopową tego układu oraz schemat montażu i pakowania zaprojektowanego układu scalonego.

# Rozdział 2

# Wprowadzenie

## 2.1 Mikroprocesorowe układy scalone

*Mikroprocesorowym układem scalonym* nazywany jest specjalizowany układ scalony (Application-Specific Integrated Circuit (ASIC)) integrujący na wspólnym podłożu krzemowym mikroprocesor, pamięci i układy peryferyjne. Wchodzący w jego skład *mikroprocesor* jest urządzeniem elektronicznym, którego działanie może być określane za pośrednictwem kodu, czyli ciągu rozpoznawanych przez takie urządzenie instrukcji.

Pierwsza informacja na temat tego typu urządzeń pojawiła się w roku *1971*, w którym to firma *Intel* zaprezentowała swój pierwszy 4-bitowy mikroprocesor *Intel 4004* [1, 2]. Ten składający się z 2300 tranzystorów układ został zrealizowany w procesie technologicznym Metal-Oxide-Semiconductor (MOS), w którym wymiar charakterystyczny tranzystorów N-type Metal-Oxide-Semiconductor (NMOS) wynosił 10 µm. Układ *Intel 4004* posiadał oddzielne magistrale instrukcji i danych, 16 rejestrów ogólnego przeznaczenia, 4-bitowy sumator i akumulator do przechowywania cząstkowych wyników dodawania. Mikroprocesor ten wspierał 46 instrukcji, z których większość wykonywana była przez 8 taktów zegara, co przy maksymalnej częstotliwości pracy wynoszącej 740 kHz, skutkowało realizacją pojedynczej instrukcji przez 10.8 µs.

Koncepcja programowalnych mikroprocesorów, powstała z myślą o ich wykorzystaniu w urządzeniach liczących, takich jak kalkulatory. Projekt układu *Intel 4004* został w roku 1969 zlecony firmie *Intel* przez japońską firmę *Busicom* zajmującą się wówczas produkcją tego typu urządzeń. Dziś, w roku 2023, mikroprocesory nie są już wykorzystywane jedynie do produkcji kalkulatorów i zawierają nie tylko jednostki arytmetyczno-logiczne, lecz również, miedzy innymi akceleratory kryptograficzne, koprocesory realizujące obliczenia algorytmów sztucznej inteligencji oraz sprzętowe kontrolery wirtualizacji [3].

### 2.1.1 Warstwowa budowa systemu mikroprocesorowego

Systemy mikroprocesorowe są układami elektronicznymi składającymi się z milionów, a niekiedy nawet i miliardów, komponentów zwanych *tranzystorami*. Wynik programu wykonywanego przez mikroprocesor jest rezultatem przepływu ładunków pomiędzy budującymi go tranzystorami. Ze względu na fakt, iż niemożliwa byłaby analiza programu oparta na przepływie ładunków przez mikroprocesor, w opisach działania systemów mikroprocesorowych stosuje się tzw. „abstrakcję", czyli opis działania systemu elektronicznego na pewnym określonym poziomie szczegółowości. W Tab. 2.1 przedstawione zostały przykładowe poziomy

abstrakcji systemu mikroprocesorowego [4]. Pogrubieniem wyróżnione zostały obszary, w obrębie których zrealizowana została niniejsza rozprawa.

Tablica 2.1: Poziomy abstrakcji systemu mikroprocesorowego.

| Domena | Poziom abstrakcji | Przykłady |
|---|---|---|
| **Oprogramowanie** | **Oprogramowanie aplikacyjne** | Aplikacje |
| | **Systemy operacyjne** | Sterowniki urządzeń, planista |
| **Sprzęt** | Architektura | Instrukcje, rejestry |
| | **Mikroarchitektura** | Ścieżki danych, kontrolery |
| | **Układy logiczne** | Sumatory, pamięci |
| | Układy cyfrowe | Bramki logiczne |
| | Układy analogowe | Wzmacniacze, filtry |
| | Elementy elektroniczne | Tranzystory |
| Fizyka | Cząstki elementarne | Elektrony |

Dalsza część niniejszej podsekcji zawierać będzie omówienie kolejnych poziomów abstrakcji przedstawionych w Tab. 2.1, począwszy od najniższego a skończywszy na najwyższym.

### Cząstki elementarne

Dane przetwarzane przez systemy mikroprocesorowe reprezentują cząstki elementarne propagowane poprzez struktury półprzewodnikowe. Realizowana przez mikroprocesor operacja odczytu, modyfikacji i ponownego zapisu danej do pamięci jest realizowana poprzez przepływ ładunków przez wspomniane komponenty i opisana zostać może za pośrednictwem praw fizycznych, takich jak równania Maxwella. Ze względu na złożoność systemów mikroprocesorowych, ich działanie przedstawiane jest przy użyciu opisów znacznie mniej szczegółowych niż opis przepływu cząstek elementarnych.

### Komponenty elektroniczne

Przepływ cząstek elementarnych przez systemy mikroprocesorowe kontrolowany jest za pośrednictwem komponentów elektronicznych zwanych *tranzystorami*. Urządzenia te, ze względu na swoje właściwości wykorzystywane są w systemach mikroprocesorowych w charakterze kluczy. Ich działanie determinuje, czy przepływ ładunków pomiędzy określonymi węzłami układu elektronicznego powinien zostać zrealizowany.

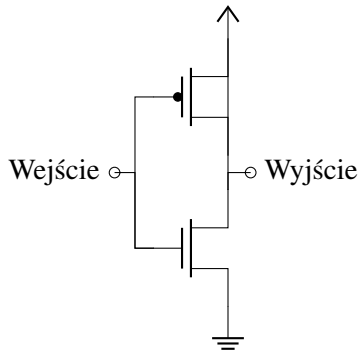
Współczesna mikroelektronika została zdominowana przez tranzystory typu Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) zwane również tranzystorami MOS [5]. Tego typu tranzystor zbudowany jest z bramki wykonanej z przewodzącego metalu, oddzielonej warstwą dielektryka (izolatora) od zubożonego (pozbawionego swobodnych nośników) podłoża krzemowego. W celu umożliwienia przepływu ładunku przez tranzystor, do podłoża podłączane są dwa wysoko-domieszkowane (zawierające w swojej strukturze krystalicznej poza atomami krzemu dużą liczbę atomów innych pierwiastków) terminale zwane źródłem oraz drenem.

### Układy analogowe

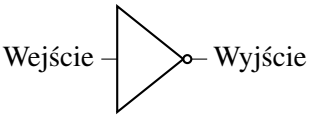
Mimo faktu, iż systemy mikroprocesorowe są urządzeniami przetwarzającymi dane binarne (dwuwartościowe), znaczna część doprowadzanych do nich sygnałów posiada charakterystykę analogową. Aby umożliwić systemowi mikroprocesorowemu analizę i interpretację sygnałów ciągłych w domenie amplitudy, niezbędna jest ich konwersja do domeny cyfrowej. Translacja pomiędzy wymienionymi domenami jest realizowana za pośrednictwem dedykowanych układów analogowych, takich jak filtry, wzmacniacze i przetworniki analogowo-cyfrowe.

### Układy cyfrowe

Klucze realizowane przy użyciu tranzystorów MOS wykorzystywane są do budowy układów elektronicznych zwanych *układami cyfrowymi*. Te działające w domenie cyfrowej urządzenia cechują się rozpoznawaniem jedynie dwóch poziomów napięciowych odpowiadających dwóm wartościom logicznym: prawdzie oraz fałszowi. Przykładowymi układami cyfrowymi są *bramki logiczne*, które swoim działaniem realizują podstawowe funkcje algebry Boole'a, takie jak iloczyn, suma i negacja. Rys. 2.1 przedstawia *inwerter*, czyli bramkę logiczną realizującą negację. Na Rys. 2.1a przedstawiony został schemat tej bramki a na Rys. 2.1b jej symbol.




(a) Schemat inwertera.




(b) Symbol inwertera.

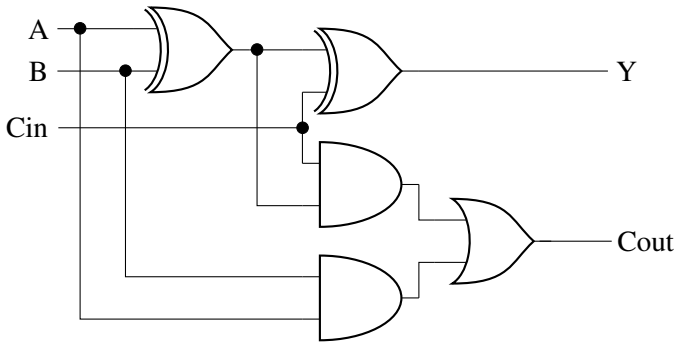
Rysunek 2.1: Inwerter.

### Układy logiczne

Bramki logiczne są układami elektronicznymi realizującymi podstawowe funkcje logiczne. W systemach mikroprocesorowych wykorzystywane są do budowy *układów logicznych*, realizujących, między innymi złożone operacje arytmetyczno-logiczne. Przykładem układu logicznego zbudowanego z bramek jest jednobitowy sumator, którego schemat przedstawiony został na Rys. 2.2.

### Mikroarchitektura

*Mikroarchitekturą* nazywamy organizację procesora wraz z głównymi jednostkami funkcjonalnymi, siecią połączeń oraz układami kontrolnymi [6]. Definiuje ona charakterystyczny dla danego mikroprocesora układ rejestrów, jednostek arytmetyczno-logicznych, maszyn stanów skończonych i wszystkich układów




Rysunek 2.2: Schemat jednobitowego sumatora.

cyfrowych implementujących architekturę. Jedna architektura może zostać zaimplementowana przy użyciu wielu różnych mikroarchitektur różniących się między sobą wydajnością, złożonością i zapotrzebowaniem energetycznym [7, 8].

### Architektura

*Architektura*, zwana również modelem programowym procesora, Instruction Set Architecture (ISA), jest interfejsem pomiędzy sprzętową implementacją mikroprocesora a niskopoziomowym oprogramowaniem. Określa ona, między innymi zbiór instrukcji wspieranych przez mikroprocesor, liczbę i rozmiar rejestrów oraz sposób dostępowania pamięci i układów peryferyjnych. Istnieją dwa główne typy architektur różniące się liczbą i złożonością definiowanych instrukcji: Reduced Instruction Set Computing (RISC) i Complex Instruction Set Computing (CISC), których przykładami są odpowiednio *ARM* i *RISC-V* oraz *x86* [9, 10].

### Systemy operacyjne

*Systemem operacyjnym* nazywamy oprogramowanie nadzorujące działanie systemu mikroprocesorowego i umożliwiające zarządzanie zadaniami wykonywanymi przez ten system. Głównymi aktywnościami realizowanymi przez tego typu oprogramowanie są dostarczanie aplikacjom warstwy abstrakcji umożliwiającej przenoszenie programów pomiędzy różnymi platformami sprzętowymi oraz zarządzanie zasobami systemu mikroprocesorowego [11]. Systemy operacyjne udostępniają aplikacjom użytkownika liczne biblioteki, stosy sieciowe i mechanizmy komunikacji międzyprocesowej, skracając tym samym czas potrzebny na tworzenie i rozwijanie nowego oprogramowania.

### Oprogramowanie aplikacyjne

Mianem *oprogramowania aplikacyjnego* określane są programy wykonywane przez system mikroprocesorowy, których działanie nadzorowane jest przez system operacyjny. Przykładami tego typu aplikacji są programy użytkowe, takie jak edytory tekstowe i przeglądarki internetowe. Oprogramowanie aplikacyjne może komunikować się z platformą sprzętową jedynie za pośrednictwem systemu operacyjnego, przy użyciu mechanizmu zwanego *wywołaniami systemowymi*.

**Wnioski**

Badania zrealizowane przez Autora niniejszej rozprawy skoncentrowane były wokół trzech głównych zagadnień: projektu odczytowego układu scalonego z wbudowanym mikroprocesorem RISC-V, budowy środowiska do testowania odczytowych układów scalonych oraz projektu mikroprocesora RISC-V. Analiza przedstawionych warstw systemów mikroprocesorowych pozwoliła na wskazanie obszarów, w obrębie których realizowane były prace:

1. odczytowy układ scalony z wbudowanym mikroprocesorem RISC-V
    - układy logiczne, prace nad projektowanym układem scalonym wymagały zaprojektowania układów logicznych, takich jak układy peryferyjne, umożliwiające wbudowanemu mikroprocesorowi komunikację ze światem zewnętrznym i sterowanie matrycą pikseli,
    - oprogramowanie aplikacyjne, działanie mikroprocesora wbudowanego w odczytowy układ scalony było kontrolowane za pośrednictwem dedykowanego oprogramowania,
2. środowisko do testowania odczytowych układów scalonych
    - układy logiczne, konstrukcja środowiska do testowania układów scalonych wymagała zaprojektowania układów logicznych, takich jak sprzętowy kontroler zapisu danych odbieranych z testowanego układu scalonego,
    - systemy operacyjne, w celu umożliwienia aplikacjom komunikacji z zaprojektowanymi układami logicznymi, opracowany został moduł jądra, pełniący rolę sterownika skonstruowanej platformy sprzętowej,
    - oprogramowanie aplikacyjne, zbudowane środowisko do testowania odczytowych układów scalonych było kontrolowane za pośrednictwem dedykowanych aplikacji sterujących,
3. mikroprocesor RISC-V
    - układy logiczne, projekt mikroprocesora RISC-V wymagał zaprojektowania układów logicznych, takich jak sumator wykorzystywany podczas wykonywania instrukcji dodawania,
    - mikroarchitektura, konstrukcja mikroprocesora wymagała opracowania całej jego mikroarchitektury, między innymi jednostki arytmetyczno-logicznej i zbioru rejestrów,
    - oprogramowanie aplikacyjne, działanie mikroprocesora zostało zweryfikowane za pośrednictwem wykonywanego przez ten rdzeń oprogramowania.

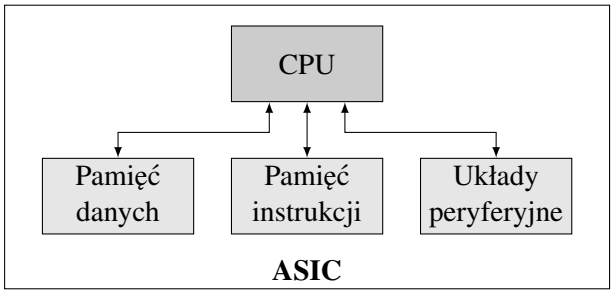
### 2.1.2 Architektury systemów mikroprocesorowych

Mikroprocesor jest układem elektronicznym wykorzystywanym do przetwarzania danych cyfrowych. Jego architektura definiuje, między innymi, jakie instrukcje może wykonywać oraz w jaki sposób może komunikować się z pamięciami i układami peryferyjnymi. Nie determinuje ona jednak metody jego integracji w kontrolowanym przez niego systemie mikroprocesorowym.

Relacja między mikroprocesorem a współpracującymi z nim układami elektronicznymi określana jest mianem *architektury systemu mikroprocesorowego*. Definiuje ona w jaki sposób mikroprocesor połączony jest z pamięciami, układami peryferyjnymi a nawet innymi mikroprocesorami. W niniejszej sekcji przedstawiona zostanie następująca klasyfikacja systemów mikroprocesorowych: mikrokontrolery, układy System on Chip (SoC), systemy lokalne oraz systemy sieciowe.

### Mikrokontrolery

*Mikrokontrolery* są niewielkimi systemami mikroprocesorowymi, które w jednym układzie scalonym integrują centralną jednostkę wykonawczą, Central Processing Unit (CPU), pamięci danych i kodu oraz liczne układy peryferyjne. Wewnątrz tego typu systemów zazwyczaj implementowane są komponenty analogowo-cyfrowe, takie jak przetworniki i komparatory, dzięki czemu mikrokontrolery znajdują zastosowanie w licznych gałęziach przemysłu [12, 13] i medycyny [14]. Uproszczony model przykładowego mikrokontrolera przedstawiony został na Rys. 2.3.



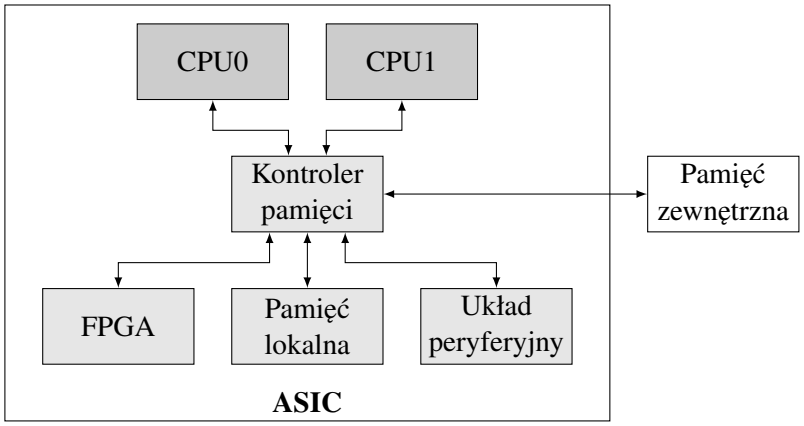

Rysunek 2.3: Uproszczony model przykładowego mikrokontrolera.

### Układy SoC

*Układy SoC* są podobnymi do mikrokontrolerów systemami mikroprocesorowymi, które integrują w pojedynczym układzie scalonym CPU, pamięci i układy peryferyjne. Jedną z głównych różnic występujących pomiędzy wymienionymi architekturami jest fakt, iż układy SoC posiadają znacznie większą moc obliczeniową niż mikrokontrolery, dzięki czemu ich praca może być nadzorowana przez systemy operacyjne ogólnego przeznaczenia, takie jak *Linux*. Wewnątrz układów SoC zintegrowane mogą zostać wielordzeniowe mikroprocesory, układy Field Programmable Gate Array (FPGA) i dedykowane układy akcelerujące przetwarzanie danych, co sprawia, że tego typu systemy znajdują zastosowanie w aplikacjach wymagających dużej wydajności, takich jak sterowanie pojazdami bezzałogowymi [15] i przetwarzanie danych sieciowych [16, 17]. Uproszczony model przykładowego układu SoC przedstawiony został na Rys. 2.4.

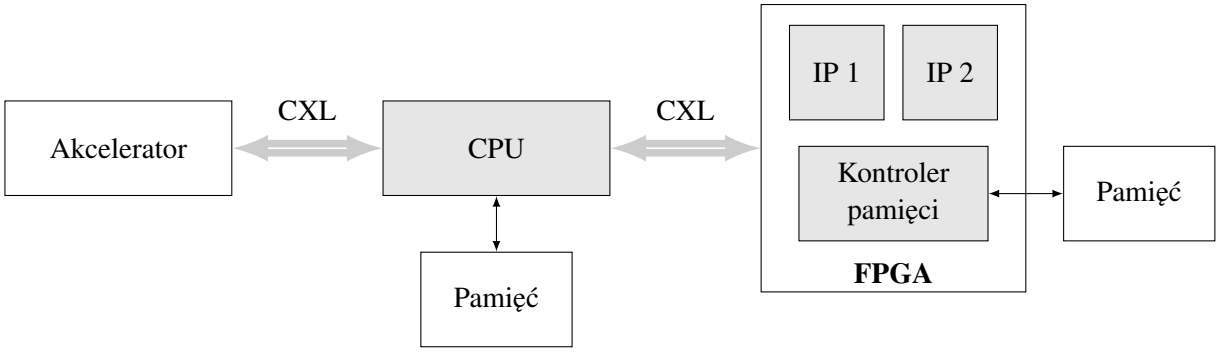
### Systemy lokalne

Mianem *systemów lokalnych* określić można systemy mikroprocesorowe, w których CPU komunikują się z układami podrzędnymi za pośrednictwem szybkich interfejsów lokalnych, takich jak Peripheral Component Interconnect Express (PCIe) i Compute Express Link (CXL) [18]. Wykorzystanie w tego typu systemach interfejsu CXL pozwala na budowę klastrów obliczeniowych współdzielących ze sobą w sposób




Rysunek 2.4: Uproszczony model przykładowego układu SoC.

wysokowydajny pamięć operacyjną [19]. Systemami lokalnymi są nie tylko urządzenia budujące centra danych, ale również, między innymi systemy kontrolno-pomiarowe wykorzystywane do testowania układów typu ASIC. Na Rys. 2.5 przedstawiony został uproszczony model przykładowego systemu lokalnego.




Rysunek 2.5: Uproszczony model przykładowego systemu lokalnego

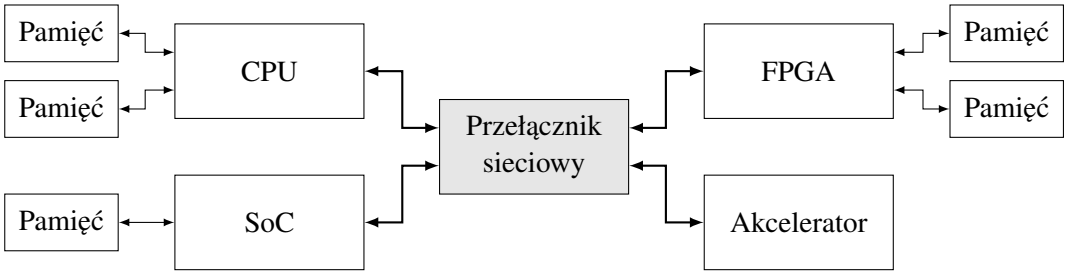
.

### Systemy sieciowe

*Systemami sieciowymi* nazywamy systemy mikroprocesorowe, których poszczególne węzły komunikują się ze sobą za pośrednictwem połączeń sieciowych. Tego typu rozwiązania są szeroko stosowane w centrach danych i superkomputerach, realizujących działania wymagające dużej mocy obliczeniowej. W skład systemów sieciowych, poza lokalnymi systemami mikroprocesorowymi, wchodzić mogą układy SoC, FPGA i dedykowane układy akcelerujące przetwarzanie danych. Na Rys. 2.6 przedstawiony został uproszczony model przykładowego systemu sieciowego.

### Wnioski

Przegląd przedstawionych architektur systemów mikroprocesorowych pozwolił na określenie docelowej architektury systemu mikroprocesorowego dla hybrydowych detektorów promieniowania jonizującego. Kryterium determinującym wybór, była konieczność integracji matrycy pikseli z mikroprocesorem we wspólnym podłożu krzemowym, co wykluczyło możliwość realizacji systemu jako lokalny albo sieciowy.




Rysunek 2.6: Uproszczony model przykładowego systemu sieciowego.

Ze względu na brak planów związanych z implementacją złożonych algorytmów, wymagających dużej mocy obliczeniowej i dużych zasobów pamięciowych, podjęta została decyzja o budowie systemu o architekturze mikrokontrolerowej.

### 2.1.3 Architektura RISC-V

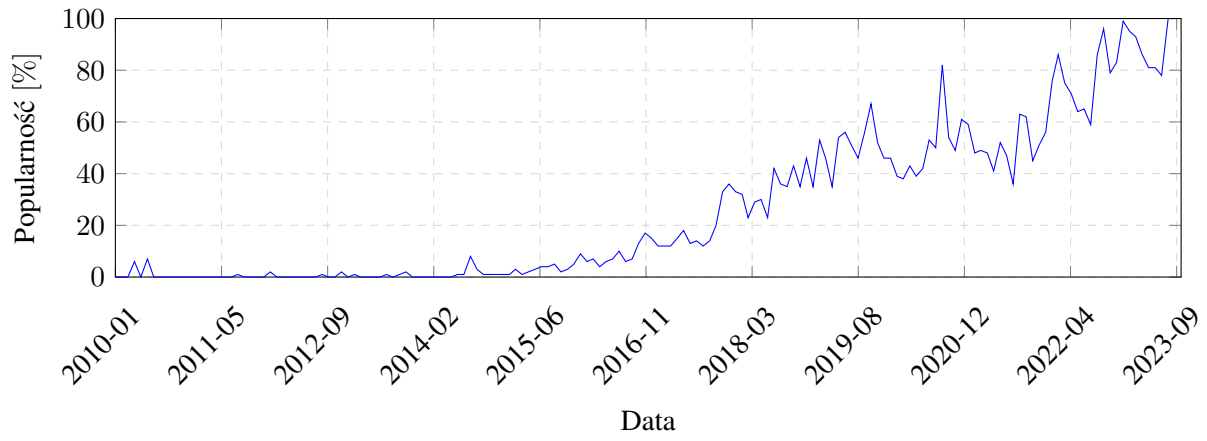
Architektura *RISC-V* została zaprezentowana w 2013 roku, jako odpowiedź na brak prostego i realistycznego zbioru instrukcji procesora (ISA), który pozwalałby na tworzenie w pełni funkcjonalnych systemów mikroprocesorowych, spełniających wymagania stawiane komercyjnie wykorzystywanym ISA ogólnego przeznaczenia [20]. Jednym z założeń przyjętych przez autorów tej architektury było opracowanie w pełni otwartego modelu, wokół którego skupiałaby się społeczność rozwijających ją inżynierów i naukowców. Drugim istotnym założeniem było unikanie optymalizacji pod kątem konkretnych technologii implementacyjnych, tak by możliwa była wydajna realizacja mikroprocesora RISC-V zarówno w układzie ASIC, jak i w FPGA.

#### Geneza

Prace nad architekturą RISC-V rozpoczęte zostały w roku 2010 na Uniwersytecie Kalifornijskim w Berkeley [21]. W badania nad nową architekturą zaangażowani byli, między innymi prof. Krste Asanović oraz dwóch absolwentów tego uniwersytetu: Yunsup Lee i Andrew Waterman, będący założycielami organizacji *SiFive*, zajmującej się projektowaniem mikroprocesorów, układów SoC i płyt deweloperskich, wykorzystujących opracowaną przez nich architekturę. Obecnie, w rozwój architektury RISC-V zaangażowanych jest ponad 350 firm i instytucji [22], wśród których znajdują się, między innymi *Google*, *Intel* oraz *NVIDIA*. Obserwowany w ostatnich latach wzrost popularności architektury RISC-V potwierdza przedstawiony na Rys. 2.7 wykres prezentujący relatywną popularność hasła „risc-v” w wyszukiwarce Google w okresie 01.01.2010 - 12.08.2023 (wartość 100 % oznacza maksymalną liczbę zarejestrowanych wyszukań) [23].

#### Zbiory instrukcji

Wśród instrukcji definiowanych przez architekturę RISC-V wyróżnić możemy bazowy zbiór instrukcji całkowitoliczbowych oraz opcjonalne zestawy rozszerzające [24]. Zbiór bazowy został sformułowany jako minimalny zbiór operacji mikroprocesora, niezbędnych do wykonywania programów ogólnego przeznaczenia,




Rysunek 2.7: Relatywna popularność hasła „risc-v” w wyszukiwarce Google w okresie 01.01.2010 - 12.08.2023.

pisanych w językach takich jak C i C++. Mikroprocesory implementujące bazowy zbiór instrukcji RISC-V mogą być wykorzystywane do budowy w pełni funkcjonalnych systemów mikroprocesorowych.

Określenie „architektura RISC-V” nie odnosi się do jednego konkretnego ISA, lecz do rodziny 4 zestawów, różniących się między sobą liczbą i szerokością rejestrów procesora. Architektura RISC-V definiuje dwa dostępne rozmiary rejestrów (32-bity i 64-bity), determinujące długość przetwarzanych słów i rozmiar wspieranej przestrzeni adresowej, oraz dwie ich liczby (32 rejestry w przypadku systemów ogólnego przeznaczenia i 16 w przypadku mikrokontrolerów). Wybrana konfiguracja bazowa wraz z wybranymi opcjonalnymi rozszerzeniami definiują pełną nazwę architektury, zgodnie z następującym schematem:

$$RVXYZ,$$

gdzie

$X$ – szerokość słowa procesora (*32* - 32-bity, *64* - 64-bity),

$Y$ – liczba rejestrów procesora (*I* - 32, *E* - 16),

$Z$ – opcjonalne rozszerzenia.

Przykładem pełnej nazwy architektury RISC-V jest $RV64I$, definiująca 64-bitowy mikroproceosor, posiadający 32 rejestry i implementujący jedynie bazowy zbiór instrukcji całkowitoliczbowych.

Architektura RISC-V została zaprojektowana w taki sposób, by umożliwić jej rozszerzanie i dostosowywanie zbiorów wspieranych instrukcji do potrzeb konkretnych aplikacji. W związku z tym, każda z bazowych architektur może zostać rozszerzona o dodatkowe zestawy instrukcji, umożliwiające realizację specjalistycznych zadań, takich jak, np. wykonywanie operacji na liczbach zmiennoprzecinkowych. Wybrane opcjonalne rozszerzenia architektury RISC-V przedstawione zostały w Tab. 2.2.

Przykładem architektury RISC-V implementującej opcjonalne zbiory instrukcji jest architektura $RV64IAV$, definiująca 64-bitowy mikroprocesor, posiadający 32 rejestry i implementujący bazowy zbiór instrukcji, rozszerzony o operacje atomowe i wektorowe.

Tablica 2.2: Wybrane opcjonalne rozszerzenia architektury RISC-V.

| Nazwa | Opis |
|---|---|
| I | Instrukcje całkowitoliczbowe |
| M | Całkowitoliczbowe mnożenie i dzielenie |
| A | Operacje atomowe |
| F | Operacje zmiennoprzecinkowe pojedynczej precyzji |
| D | Operacje zmiennoprzecinkowe podwójnej precyzji |
| V | Operacje wektorowe |

**Wnioski**

Autor niniejszej rozprawy zdecydował się na wykorzystanie w projektowanym systemie mikroprocesorowym, mikroprocesora o architekturze RISC-V. Wpływ na tę decyzję miały zarówno model dystrybucji i rosnąca popularność tej architektury, jak i wynikająca z nich niemała liczba mikroprocesorów, udostępnianych w ramach licencji, pozwalających na ich nieodpłatne wykorzystywanie. Nie bez znaczenia pozostał również fakt, iż architektura RISC-V od samego początku przewidywała możliwość definiowania własnych rozszerzeń, dając tym samym szansę na opracowanie niestandardowego zbioru instrukcji, dedykowanego hybrydowym detektorom promieniowania jonizującego, będącego jednym z potencjalnych kierunków kontynuacji badań realizowanych przez Autora niniejszej rozprawy.

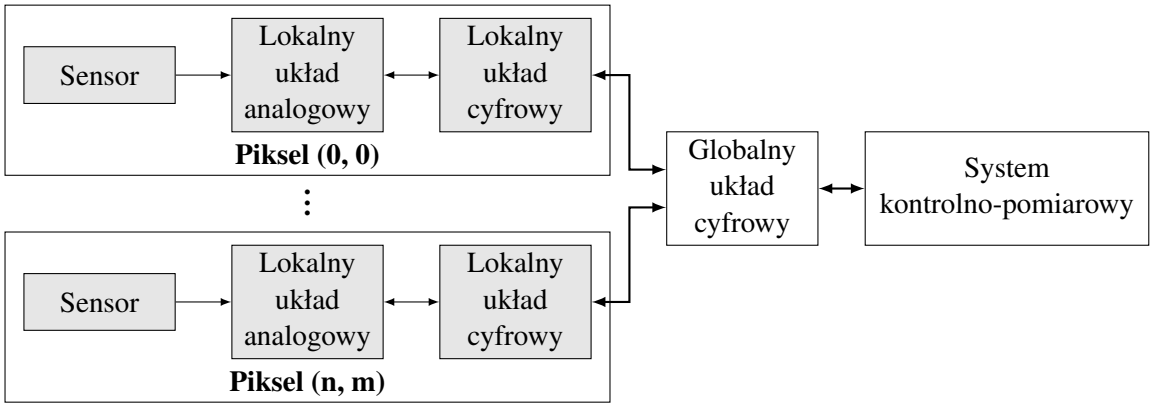
## 2.2 Hybrydowe detektory promieniowania jonizujacego

### 2.2.1 Budowa i działanie hybrydowych detektorów pikselowych

*Hybrydowe detektory promieniowania jonizujacego* są układami elektronicznymi wykorzystywanymi do detekcji i pomiarów parametrów promieniowania, przez które przechodzi. Urządzenia te zbudowane są z sensorów generujących impulsy elektryczne, których właściwości są zależne od materiału, z którego są zbudowane, parametrów rejestrowanego promieniowania, układów analogowych, odpowiedzialnych za wzmacnianie, kształtowanie i przetwarzanie sygnałów generowanych przez sensory oraz układów cyfrowych, sterujących działaniem detektorów. Hybrydowe detektory promieniowania jonizujacego znajdują zastosowanie, między innymi w eksperymentach fizyki wysokich energii [25], badaniach materiałowych [26] i medycynie [27].

Jednym z typów hybrydowych detektorów promieniowania jonizującego są *detektory pikselowe* (ang. Hybrid Pixel Detector (HPD)). Każdy z pikseli HPD składa się z sensora, lokalnego układu analogowego i lokalnego układu cyfrowego, które pozwalają na odczyt parametrów promieniowania przenikającego przez dany piksel. Działanie poszczególnych pikseli kontrolowane jest przez globalne układy cyfrowe, pośredniczące w komunikacji pomiędzy systemami kontrolno-pomiarowymi i matrycami. Na Rys. 2.8 przedstawiony został uproszczony model HPD. Sensor stanowi zazwyczaj matryca polaryzowanych zaporowo diod półprzewodnikowych, np. Si, CdTe), natomiast układy analogowe i cyfrowe są umieszczane w układzie ASIC

o geometrii dopasowanej do sensora. Sensor i układ odczytowy są produkowane niezależnie i w różnych technologiach, stąd nazwa: „hybrydowe”.




Rysunek 2.8: Uproszczony model HPD.

Niedokładność półprzewodnikowych procesów produkcyjnych skutkuje w układach typu ASIC rozrzutem właściwości komponentów o takich samych parametrach geometrycznych. Zjawisko to może dawać w wyniku odczyt różnych wartości z pikseli wystawionych na działanie jednorodnej wiązki promieniowania. W celu ograniczenia wpływu parametrów środowiskowych i rozrzutu komponentów na wyniki detekcji, właściwa rejestracja promieniowania poprzedzana jest wykonaniem procedury kalibracyjnej. Ze względu na różnorodność hybrydowych detektorów promieniowania [28], opracowanych zostało wiele algorytmów kalibracyjnych, opartych zarówno na wykorzystaniu zewnętrznych urządzeń pomocniczych [29], jak i realizacji tej procedury przez zintegrowany układ sterujący [30].

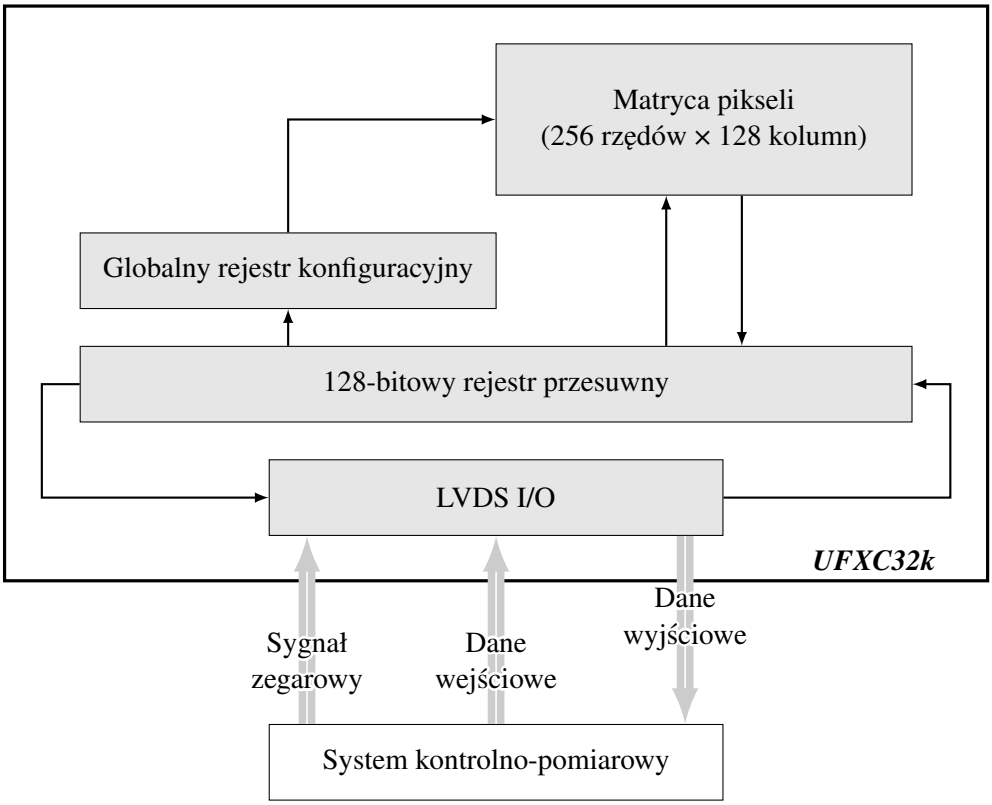
### 2.2.2 Omówienie wybranych układów odczytowych

W niniejszej sekcji przedstawionych zostało pięć scalonych układów odczytowych, których analiza pomogła Autorowi niniejszej rozprawy w określeniu architektury prezentowanego rozwiązania. Układy *UFXC32k* oraz *PIX45XF* zostały zaprojektowane przez pracowników Grupy Mikroelektronicznej, działającej w Katedrze Metrologii i Elektroniki AGH w Krakowie, której członkiem jest Autor niniejszej rozprawy.

#### *UFXC32k*

Układ scalony *UFXC32k* został wykonany w 2014 roku w procesie technologicznym Complementary Metal-Oxide-Semiconductor (CMOS) 130 nm [31]. Zawiera on matrycę 128 × 256 pikseli o rozmiarze 75 µm × 75 µm. Komunikacja z tym układem odbywała się się za pośrednictwem standardu Low-Voltage Differential Signaling (LVDS), wykorzystywanego zarówno do transmisji sygnału zegarowego, jak i danych zapisywanych i odczytywanych z urządzenia. Układ nadawczo-odbiorczy w *UFXC32k* został zaimplementowany w formie 128-bitowego rejestru przesuwnego, którego wejście i wyjścia podłączone były do odpowiednich linii LVDS, łączących urządzenie z nadzorującym go systemem kontrolno-pomiarowym. Na Rys. 2.9 przedstawiony został uproszczony model układu *UFXC32k*.

Zaimplementowany w układzie *UFXC32k* protokół komunikacyjny opierał się na transmisji niestandardowego ciągu bitowego. Sterowanie urządzeniem odbywało się za pośrednictwem dedykowanej linii sygna-

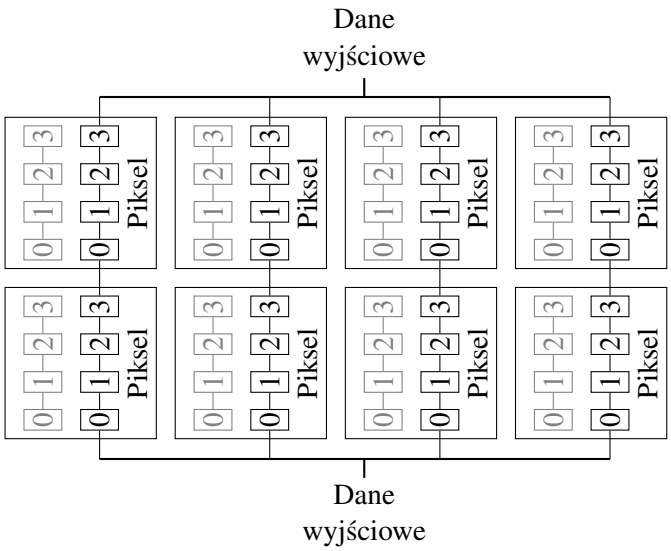



Rysunek 2.9: Uproszczony model układu *UFXC32k*.

łowej, ustawienie na której wysokiego stanu logicznego, powodowało przepisanie zawartości 128-bitowego rejestru przesuwnego do 128-bitowego globalnego rejestru konfiguracyjnego. Wspomniany rejestr przesuwny wykorzystywany był nie tylko do ustawiania globalnej konfiguracji układu, ale również do odczytu zawartości pikseli i zapisywania ich konfiguracji lokalnych.

Każdy z pikseli zawierał w sobie dwa typy 16-bitowych rejestrów: lokalne rejestry konfiguracyjne oraz lokalne rejestry przesuwne. W każdej ze 128 kolumn matrycy, zaimplementowany został 4096-bitowy rejestr przesuwny, składający się z lokalnych rejestrów każdego z 256 pikseli. Zastosowany przez autorów układu *UFXC32k* mechanizm komunikacyjny pozwolił na znaczną redukcję zagęszczenia ścieżek w układzie scalonym (względem bezpośrednio odczytywanych i zapisywanych pikseli), jednakże skutkował koniecznością przesuwania danych przez 4096-bitowe rejestry przesuwne, zarówno w przypadku odczytu, jak i zapisu danych do matrycy. Na Rys. 2.10 przedstawiony został uproszczony model przykładowej matrycy pikseli, implementującej podobne rozwiązanie.

Przesuwanie danych przez rejestry przesuwne matrycy realizowane było przy wykorzystaniu 128-bitowego rejestru przesuwnego i globalnego rejestru konfiguracyjnego. Propagacja danych przez piksele poprzedzana była ustawianiem w rejestrze konfiguracyjnym bitu, odpowiedzialnego za włączanie przesuwania. Dane były przekazywane pomiędzy kolejnymi przerzutnikami, po wystąpieniach zbocza narastającego dedykowanego sygnału zegarowego, sterowanego za pośrednictwem dedykowanej linii LVDS. Po wystąpieniu aktywnego zbocza tego sygnału, ostatnie bity poszczególnych rejestrów przesuwnych matrycy, wpisywane były do odpowiadających im przerzutników 128-bitowego rejestru przesuwnego (do najmłodszego bitu tego rejestru wpisywana była wartość odczytywana z kolumny zerowej).




Rysunek 2.10: Uproszczony model przykładowej matrycy pikseli.

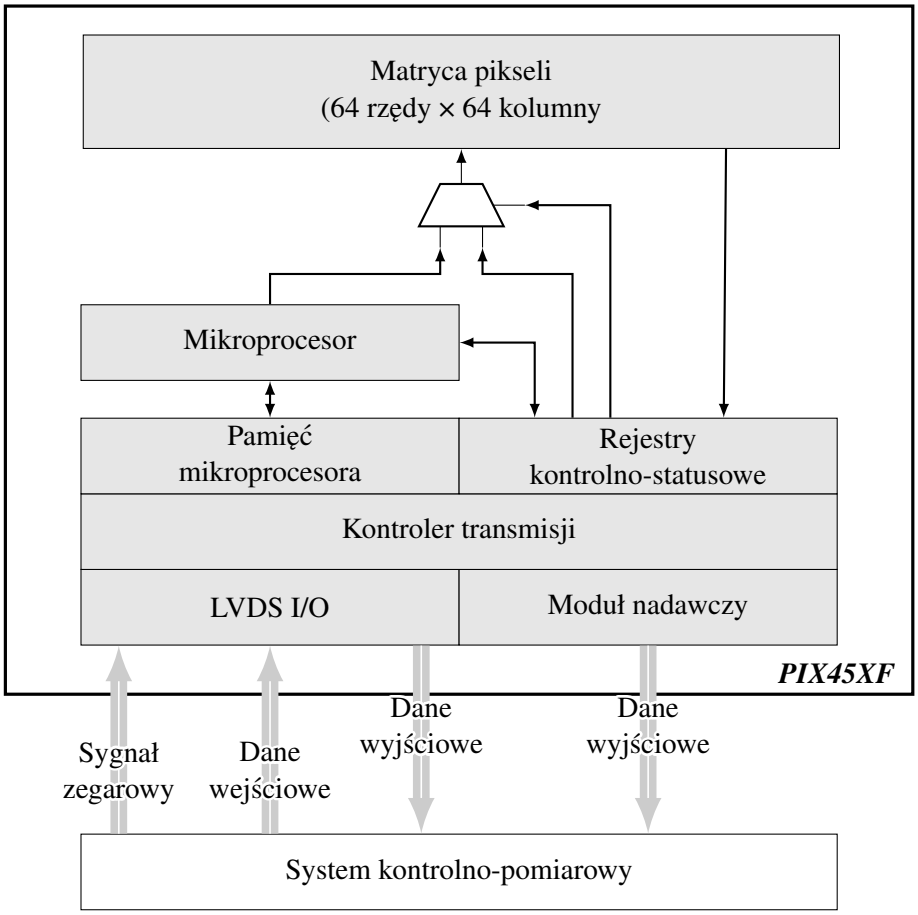
Domyślną wartościową wpisywaną do rejestrów przesuwnych matrycy były zera. Ustawienie odpowiedniego bitu globalnego rejestru konfiguracyjnego, pozwalało na zastąpienie zer wartościami odczytywanymi ze 128-bitowego rejestru przesuwnego. Mechanizm ten pozwalał na wpisywanie parametrów konfiguracyjnych do lokalnych rejestrów przesuwnych poszczególnych pikseli, które nastepnie mogły być zatrzaskiwane w lokalnych rejestrach konfiguracyjnych. Przepisania wartości z lokalnych rejestrów przesuwnych do lokalnych rejestrów konfiguracyjnych realizowane były przy wykorzystaniu dedykowanej linii LVDS.

### *PIX45XF*

Zaprojektowany w 2018 roku układ *PIX45XF* [32] był HPD wykonanym w procesie technologicznym CMOS 40 nm. Wewnątrz układu zintegrowane zostały matryca 4096 pikseli (64 wiersze × 64 kolumny), której wewnętrzna architektura wzorowana była na architekturze matrycy zaimplementowanej w układzie *UFXC32k*, oraz dedykowany układ odczytowy. Interesującym, nowatorskim rozwiązaniem zastosowanym w tym układzie był mikroprocesor sterujący matrycą pikseli.

Komunikacja z układem *PIX45XF* została zrealizowana przy wykorzystaniu linii LVDS, które użyte zostało zarówno do transmisji sygnału zegarowego, jak danych wejściowych i wyjściowych. W przeciwieństwie do układu *UFXC32k*, dane nie były bezpośrednio przesyłane do układu *PIX45XF*, lecz ich transmisja poprzedzana była kodowaniem realizowanym przy użyciu protokołu 8b10b. W celu skrócenia czasu przesyłu danych odczytywanych z matrycy, układ *PIX45XF* wyposażony został w dedykowany moduł nadawczy, umożliwiający transmisję danych do systemu kontrolno-pomiarowego za pośrednictwem 4 dedykowanych linii LVDS, z których każda pozwalała na przesył danych z przepustowością sięgająca 4 Gbps. Na Rys. 2.11 przedstawiony został uproszczony model układu *PIX45XF*.

Wymiana danych pomiędzy systemem kontrolno-pomiarowym a układem *PIX45XF* realizowana była w sposób w pełni asynchroniczny. W związku z tym, urządzenia nie mogły przewidzieć, kiedy za pośrednictwem łączących je linii LVDS transmitowane będą użyteczne dane. W celu wyeliminowania ryzyka




Rysunek 2.11: Uproszczony model układu *PIX45XF*.

przetwarzania niepoprawnych danych i przeoczenia danych wartościowych, transmisja użytecznych danych poprzedzana była transmisją predefiniowanych symboli synchronizacyjnych.

Nawiązywanie połączenia pomiędzy urzadzeniami, rozpoczynało się od wysłania przez system kontrolno-pomiarowy sekwencji symboli inicjalizacyjnych, na które układ scalony odpowiadał ramkami potwierdzającymi nawiązanie synchronizacji [33]. Dane odbierane przez układ *PIX45XF* analizowane były przez wbudowany kontroler transmisji, który na podstawie kodu przetwarzanej komendy określał, jaka operacja powinna zostać wykonana przez urządzenie. Przykładowymi komendami wspieranymi przez kontroler były zapis i odczyt rejestrów kontrolno-statusowych oraz zapis pamięci mikroprocesora.

Zaimplementowane w układzie *PIX45XF* rejestry kontrolno-statusowe pozwoliły na sterowanie urządzeniem i monitorowanie jego stanu. Za ich pośrednictwem możliwe było nie tylko ustawienie parametrów globalnych układu, takich jak konfiguracje analogowa i cyfrowa, ale również odczyt jego statusu, między innymi licznika programu mikroprocesora. Wśród rejestrów kontrolnych i statusowych zaimplementowany został także 64-bitowy, dwukierunkowy rejestr danych, umożliwiający zapis i odczyt danych z matrycy pikseli.

Jedną z metod sterowania matrycą zaimplementowanych w układzie *PIX45XF* pikseli była aktualizacja dedykowanego rejestru kontrolnego. Mechanizm ten nie spełniał jednak wszystkich założonych wymagań projektowych, ponieważ każda aktualizacja sygnałów sterujących matrycą wymagała wysłania do układu odpowiedniej komendy. W celu umożliwienia sterowania matrycą pikseli z rozdzielczością czasową wyno-

szącą jeden takt zegara, w układzie *PIX45XF* zaimplementowany został mikroprocesor sterujący, którego rolą było odczytywanie komend zapisanych w dedykowanej pamięci i ustawianie na ich podstawie odpowiednich sygnałów kontrolnych.

### *Timepix3*

Układ *Timepix3* został wykonany w procesie technologicznym CMOS 130 nm [34], a w jego wnętrzu umieszczona została matryca 256 × 256 pikseli o rozmiarze 55 µm × 55 µm. Piksele pogrupowane zostały w 8-elementowe bloki nazwane „superpikselami", nadzorujące komunikację pomiędzy poszczególnymi pikselami a systemem kontrolno-pomiarowym. Odczyt danych z układu został zrealizowany za pośrednictwem 8 linii różnicowych Scalable Low-Voltage Signaling (SLVS), pracujących w trybie Double Data Rate (DDR), z których każda transmitować mogła dane z przepustowością sięgającą 640 Mbps.

W układzie *Timepix3* zaimplementowane zostały dwa tryby odczytowe. W pierwszym z nich, układ scalony samodzielnie przesyłał dane do zewnętrznego systemu kontrolno-pomiarowego, bezpośrednio po zakończeniu rejestracji promieniowania. W drugim, układ po zakończeniu pomiaru oczekiwał na nadejście komendy wyzwalającej procedurę odczytową.

Dane transmitowane pomiędzy pikselami a superpikselami budowanymi przez te piksele przesyłane były w postaci pakietów. Mianem pakietu określany był 48-elementowy ciąg bitowy, zawierający, między innymi informacje o współrzędnych piksela i wynikach wykonanej przez ten piksel rejestracji promieniowania. Ze względu na fakt, iż tylko jeden z ośmiu pikseli budujących dany superpiksel mógł komunikować się z tym elementem, dostęp do superpiksela był nadzorowany w oparciu o algorytm karuzelowy (ang. round robin). Mechanizm ten wykorzystany został również do określania, dane z którego superpiksela powinny zostać wysłane do systemu kontrolno-pomiarowego.

### *EIGER*

*EIGER*, jest wykonanym w procesie technologicznym CMOS 250 nm [35], układem scalonym, we wnętrzu którego umieszczona została matryca 256 × 256 pikseli. Każdy z pikseli o rozmiarze 75 µm × 75 µm został zaprojektowany do rejestracji promieniowania jonizującego. Odczyt wyników pomiaru promieniowania realizowany był za pośrednictwem 32 linii sygnałowych, działających z częstotliwością 100 MHz, w trybie DDR.

Odczyt zawartości poszczególnych pikseli układu *EIGER* rozpoczynał się od załadowania odpowiedniej konfiguracji do dedykowanego rejestru przesuwnego. Kolejne bity tego rejestru określały, które z wierszy matrycy powinny zostać odczytane. Podczas procedury odczytowej dane były przesyłane wzdłuż kolumn do dedykowanych bloków zwanych „superserializerami", z których każdy sterował jedną z 32 linii sygnałowych, łączących układ scalony z systemem kontrolno-pomiarowym. Każdy superserializer odpowiedzialny był za transmisję danych odczytywanych z jednego z 8-kolumnowych bloków zwanych „superkolumnami".

### *IBEX*

Układ scalony *IBEX* został wykonany w procesie technologicznym CMOS 110 nm [36]. W jego wnętrzu umieszczona została matryca 256 × 256 pikseli o rozmiarze 75 µm × 75 µm. Odczyt danych z układu realizowany był za pośrednictwem 16 linii sygnałowych.

Wewnątrz układu scalonego zaimplementowane zostały dwa moduły sterujące matrycą: kontroler wierszy oraz kontroler kolumn. Pierwszy z nich, został wykorzystany do konfiguracji parametrów pikseli wchodzących w skład poszczególnych wierszy. Drugi, kontrolował odczyt danych z kolejnych kolumn matrycy. Odczytywane dane były przekazywane do modułu nadzorującego komunikację z systemem kontrolno-pomiarowym.

### Wnioski

Przedstawione w niniejszej sekcji układy odczytowe różniły się zarówno pod kątem architektury, jak i sposobu działania. Mimo to, Autorowi niniejszej rozprawy udało się wskazać trzy główne zadania, które powinny zostać zrealizowane przez projektowany system mikroprocesorowy dla hybrydowych detektorów promieniowania jonizującego:

1. ustawianie globalnej konfiguracji matrycy pikseli,
2. zapis danych do poszczególnych pikseli,
3. odczyt zawartości poszczególnych pikseli.

Ze względu na fakt, iż w zaprojektowanym przez Autora niniejszej rozprawy układzie scalonym umieszczona została matryca pikseli skonstruowana na potrzeby układu *PIX45XF*, będąca zmodyfikowaną wersją matrycy zintegrowanej w układzie *UFXC32k*, rozważania zostały skoncentrowane na analizie rozwiązań opracowanych przez uprzednio wspomnianą Grupę Mikroelektroniczną.

Zastosowane w układzie *UFXC32k* rozwiązania pozwoliły na spełnienie wszystkich wymagań, stawianych scalonym układom odczytowym dla HPD. Funkcjonalność tego układu została jednak znacząco ograniczona poprzez implementację komunikacji opartej na 128-bitowym rejestrze przesuwnym oraz umieszczenie w układzie tylko jednego globalnego rejestru konfiguracyjnego. Zastosowane mechanizmy niosły bowiem ze sobą ryzyko nadpisania globalnych konfiguracji urządzenia, podczas aktualizacji sygnałów sterujących matrycą. Ponadto, w przeciwieństwie do układu *PIX45XF*, układ *UFXC32k* nie pozwalał na sterowanie matrycą pikseli z rozdzielczością czasową wynoszącą jeden takt zegara.

Układ *PIX45XF*, będący następcą układu *UFXC32k*, pozbawiony był ograniczeń występujących w poprzedniku. Implementacja globalnych rejestrów kontrolno-statusowych i mikroprocesora sterującego matrycą pikseli pozwoliły na separację parametrów konfiguracyjnych i sterowanie matrycą z rozdzielczością czasową wynoszącą jeden takt zegara. Przeprowadzone przez Autora niniejszej rozprawy testy układu *PIX45XF* pozwoliły na sformułowanie potencjalnych usprawnień, spośród których najistotniejszym było wyeliminowanie systemu kontrolno-pomiarowego i zaimplementowanie logiki sterującej HPD wewnątrz układu scalonego. Wyniki realizacji niniejszej koncepcji zostały przedstawione przez Autora w niniejszej rozprawie.

# Rozdział 3

# Odczytowy układ scalony z wbudowanym mikroprocesorem RISC-V

## 3.1 Architektura sprzętowa

### 3.1.1 Wykorzystany mikroprocesor RISC-V

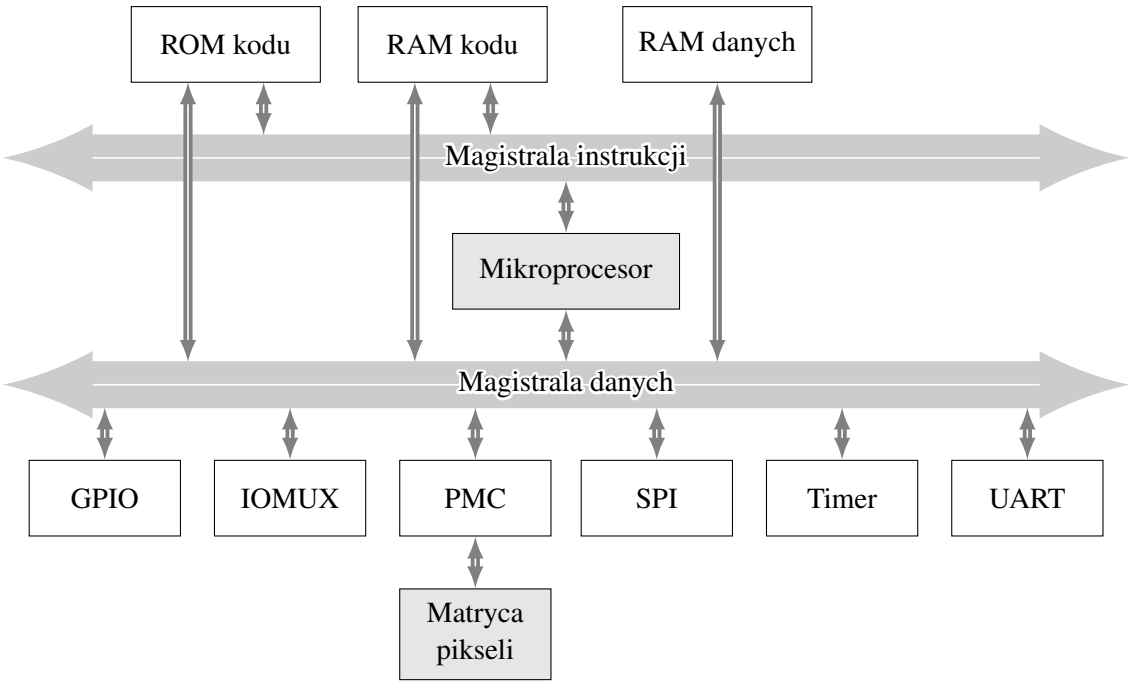
Jedną z pierwszych decyzji projektowych podjętych przez Autora niniejszej rozprawy było określenie kierunku i zakresu realizowanych badań. Ze względu na fakt, iż rozpoczynane wówczas działania były pierwszymi badaniami poświęconymi rozwojowi scalonych systemów mikroprocesorowych realizowanymi w Katedrze Metrologii i Elektroniki Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie, podjęta została decyzja o nieprojektowaniu mikroprocesora i wykorzystaniu jednego z układów udostępnionych w ramach licencji pozwalających na bezpłatne wykorzystanie w projekcie. Autor rozprawy postanowił rozpocząć badania od integracji mikroprocesora z układem odczytowym dla sensora promieniowania jonizującego, odkładając projekt własnej jednostki wykonawczej na dalsze etapy prac badawczych.

Mikroprocesorem wybranym do sterowania scalonym układem odczytowym dla sensora promieniowania jonizującego był *Ibex*. Autor niniejszej rozprawy zdecydował się na wybór tego CPU z następujących powodów:

- kod źródłowy tego mikroprocesora [37] został udostępniony w serwisie GitHub w ramach licencji Apache 2.0 [38], pozwalającej na bezpłatne wykorzystywanie opublikowanego projektu,
- rdzeń ten został zaprojektowany pod kątem wykorzystania w systemach wbudowanych i posiadał wiele parametrów konfiguracyjnych, umożliwiających dostosowanie go do konkretnych aplikacji,
- wraz z kodem źródłowym rdzenia udostępnione zostały dokumentacja [39] i przykładowy projekt [40],
- układ ten został zaimplementowany w języku SystemVerilog, który został wybrany przez Autora niniejszej rozprawy do realizacji projektowanego odczytowego układu scalonego,

- projekt rozwijany jest od roku 2015 (początkowo przez naukowców z Politechniki Federalnej w Zurychu i Uniwersytetu Bolońskiego, obecnie przez organizację *lowRISC*, zaangażowaną w rozwój otwartoźródłowych narzędzi do syntezy i implementacji układów scalonych oraz projekty systemów mikroprocesorowych opartych na architekturze RISC-V).

Ten 32-bitowy mikroprocesor został zaprojektowany zgodnie z założeniami architektury harwardzkiej, co oznacza, że komunikował się on ze światem zewnętrznym za pośrednictwem dwóch wyodrębnionych magistral. Cecha ta zdeterminowała architekturę całego układu scalonego, powodując, iż dane i instrukcje transmitowane były poprzez oddzielne interfejsy. Do pierwszej z tych magistral (magistrali danych) podłączone zostały układy peryferyjne i pamięci. Druga z nich (magistrala instrukcji) podłączona została do pamięci przechowujących kod wykonywany przez mikroprocesor. Ogólna architektura zaprojektowanego scalonego układu odczytowego z wbudowanym mikroprocesorem RISC-V przedstawiona została na Rys. 3.1.




Rysunek 3.1: Ogólna architektura scalonego układu odczytowego z wbudowanym mikroprocesorem RISC-V.

### 3.1.2 Pamięci

Pamięci są układami elektronicznymi służącymi do przechowywania danych cyfrowych. Urządzenia te wykorzystywane są przez systemy mikroprocesorowe do magazynowania kodów aplikacji wykonywanych przez mikroprocesory i danych przetwarzanych przez te programy. W prezentowanym scalonym układzie odczytowym umieszczone zostały trzy pamięci: Random-Access Memory (RAM) danych oraz Read-Only Memory (ROM) i RAM kodu, odpowiedzialne kolejno za przechowywanie danych programów, kodu aplikacji inicjalizującej i kodu programu sterującego układem odczytowym.

Ze względu na ograniczenia licencyjne technologii półprzewodnikowej, w której zrealizowany został prezentowany układ scalony, Autor nie miał możliwości umieszczenia w systemie nieulotnej pamięci Flash,

do której wgrywane mogłyby być programy wykonywane przez mikroprocesor. Niezbędna była więc implementacja mechanizmu, pozwalającego na wgrywanie do układu scalonego kodów aplikacji, które powinny być wykonywane przez mikroprocesor. Wyzwanie to zostało zrealizowane poprzez umieszczenie w systemie pamięci ROM, zbudowanej z komórek standardowych udostępnianych przez wybraną technologię półprzewodnikową, w której zawarty był kod programu wgrywającego do ulotnej pamięci RAM kod docelowej aplikacji (bootloader). W układzie scalonym umieszczone zostały dwie pamięci RAM, z których każda miała pojemność wynoszącą 16 kB.

### 3.1.3 Układy peryferyjne

Systemy mikroprocesorowe do poprawnego działania wymagają jedynie komponentów umożliwiających przechowywanie danych i kodu aplikacji. Zazwyczaj jednak realizacja wykonywanego programu wymaga interakcji mikroprocesora z otoczeniem, realizowanej za pośrednictwem układów elektronicznych zwanych peryferiami. Prezentowany w niniejszej rozprawie system mikroprocesorowy wyposażony został w układy peryferyjne, pozwalające na sterowanie układem odczytowym dla sensora promieniowania jonizującego. Wszystkie przedstawione w dalszej części tej sekcji peryferia zostały zaprojektowane przez Autora rozprawy.

Jednym z najmniej skomplikowanych, a zarazem najczęściej wykorzystywanych układów peryferyjnych jest kontroler General Purpose Input Output (GPIO), składający się z obwodów elektronicznych zarządzających cyfrowymi wyprowadzeniami ogólnego przeznaczenia. Peryferium to pełniło istotną rolę w opracowanej procedurze inicjalizacyjnej, ponieważ umożliwiło sterowanie przebiegiem tego procesu i pozwoliło na sygnalizowanie potencjalnych błędów. Wpływ cyfrowych wyprowadzeń ogólnego przeznaczenia na przebieg inicjalizacji przedstawiony został w Podsekcji 3.2.1.

Interfejsem komunikacyjnym wykorzystywanym do wymiany danych z zaprojektowanym systemem mikroprocesorowym był Universal Asynchronous Receiver / Transmitter (UART). Kontroler tego interfejsu szeregowego umożliwił dwukierunkową, asynchroniczną komunikację pomiędzy mikroprocesorem RISC-V, a zewnętrznym systemem nadzorującym działanie scalonego układu odczytowego. Funkcja tego peryferium nie została jednak ograniczona jedynie do odbioru komend i transmisji informacji diagnostycznych, ponieważ pełniło ono również bardzo istotną rolę w procesie inicjalizacji całego systemu. Kontroler UART umożliwił przesyłanie do układu scalonego danych, które po odebraniu zapisywane były przez aplikację inicjalizującą w pamięci RAM kodu, a przedstawiony proces stał się podstawową metodą programowania prezentowanego systemu mikroprocesorowego. Alternatywnym sposobem programowania opracowanego systemu było kopiowanie do pamięci RAM danych odczytywanych z karty Secure Digital (SD), realizowane za pośrednictwem interfejsu Serial Peripheral Interface (SPI).

Działanie każdego ze wspomnianych układów peryferyjnych wymagało dostępu do fizycznych wyprowadzeń układu scalonego. Ze względu na ograniczenia wynikające z wybranych technologii produkcji i pakowania układu scalonego, nie było jednak możliwe przypisanie poszczególnym peryferiom dedykowanych wyprowadzeń. Sposobem na załagodzenie wspomnianego ograniczenia stała się implementacja modułu Input Output Multiplexer (IOMUX), umożliwiającego współdzielenie jednego fizycznego wyprowadzenia

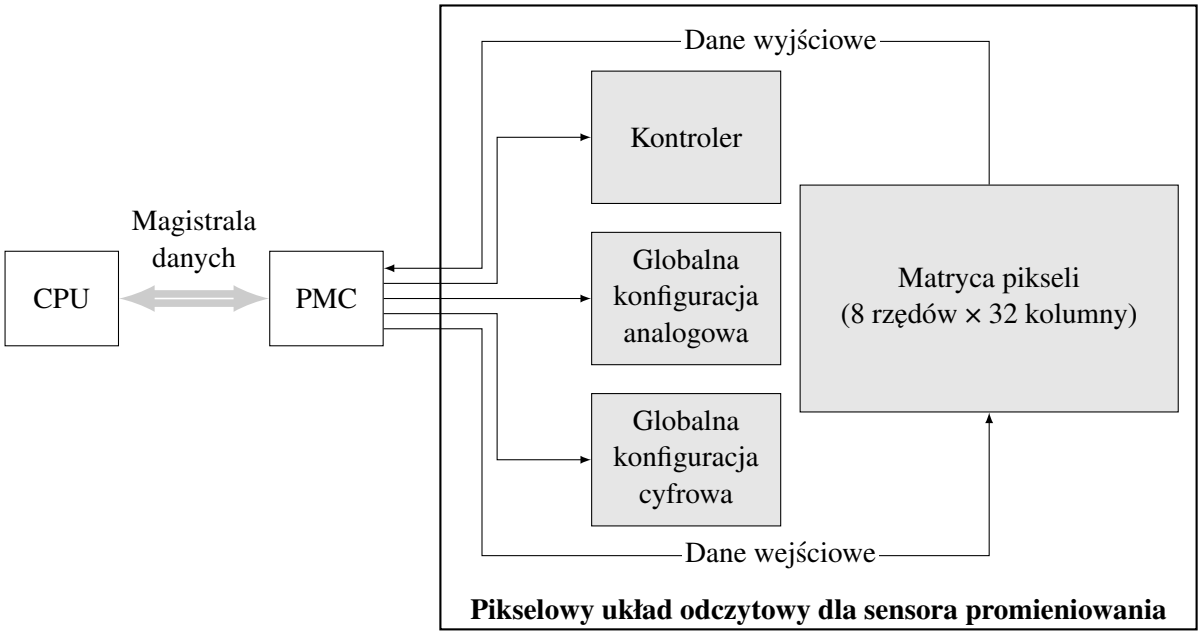
przez kilka układów peryferyjnych. Jedynym peryferium, które nie zostało podłączone do tego komponentu był układ czasowo-licznikowy (Timer), umożliwiający cykliczne wykonywanie zadań.

### 3.1.4 Kontroler układu odczytowego dla sensora promieniowania jonizującego

#### Interfejs układu odczytowego

Układ odczytowy dla sensora promieniowania jonizującego jest układem elektronicznym składającym się z pikseli, które za pomocą wiązań typu bump-bonding połączone zostać mogą z pikselowym sensorem promieniowania jonizującego. W tak skonstruowanym systemie sensor pełni rolę źródła impulsów elektrycznych, których parametry są zależne od liczby zarejestrowanych cząstek promieniowania i których przetwarzanie realizowane jest w analogowych torach odczytowych zlokalizowanych w pikselach układu odczytowego. Połączenie kontrolera matrycy pikseli z układem odczytowym dla sensora promieniowania jonizującego przedstawione zostało na Rys. 3.2. Działanie i parametry układu odczytowego mogą być konfigurowane za pośrednictwem następujących interfejsów komunikacyjnych:

- kontrolnego, podłączonego do kontrolera układu odczytowego,
- konfiguracji analogowej, sterującego globalnymi parametrami analogowymi,
- konfiguracji cyfrowej, determinującego tryb pracy układu odczytowego,
- danych wejściowych, umożliwiającego wpisywanie danych do pikseli,
- danych wyjściowych, pozwalającego na odczytywanie danych z pikseli.




Rysunek 3.2: Połączenie kontrolera matrycy pikseli z układem odczytowym dla sensora promieniowania jonizującego.

W dalszej części niniejszej rozprawy pikselowy układ odczytowy dla sensora promieniowania jonizującego określany będzie mianem matrycy pikseli, Pixel Matrix (PM).

### Ogólna architektura kontrolera układu odczytowego jonizującego

Kontroler układu odczytowego dla sensora promieniowania jonizującego, który w dalszej części niniejszej rozprawy nazywany będzie Kontrolerem Matrycy Pikseli, Pixel Matrix Controller (PMC), był układem peryferyjnym pełniącym rolę interfejsu pomiędzy mikroprocesorem RISC-V a PM. Komponent ten został dostosowany do sterowania matrycą składającą się z 256 pikseli (8 rzędów × 32 kolumny), zaprojektowanych przez naukowców z Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie [41]. Ze względu na fakt, iż szerokość interfejsów danych PM odpowiadała liczbie pikseli umieszczonych w wierszu, liczba kolumn PM została dopasowana do szerokości słowa mikroprocesora. Sprawiło to, że zarówno ustawianie interfejsu danych wejściowych, jak i odczytywanie interfejsu danych wyjściowych, realizowane były za pośrednictwem pojedynczych instrukcji zapisu (`sw`) i odczytu (`lw`) słowa.

Interfejs kontrolny PM umożliwiał PMC sterowanie działaniem układu odczytowego. Przesyłanie za jego pośrednictwem odpowiednich wartości pozwalało na wykonywanie operacji, takich jak zapis konfiguracji do pikseli i rejestracja padającego promieniowania. Interfejs ten składał się z sześciu sygnałów, z których każdy spełniał ściśle określoną rolę:

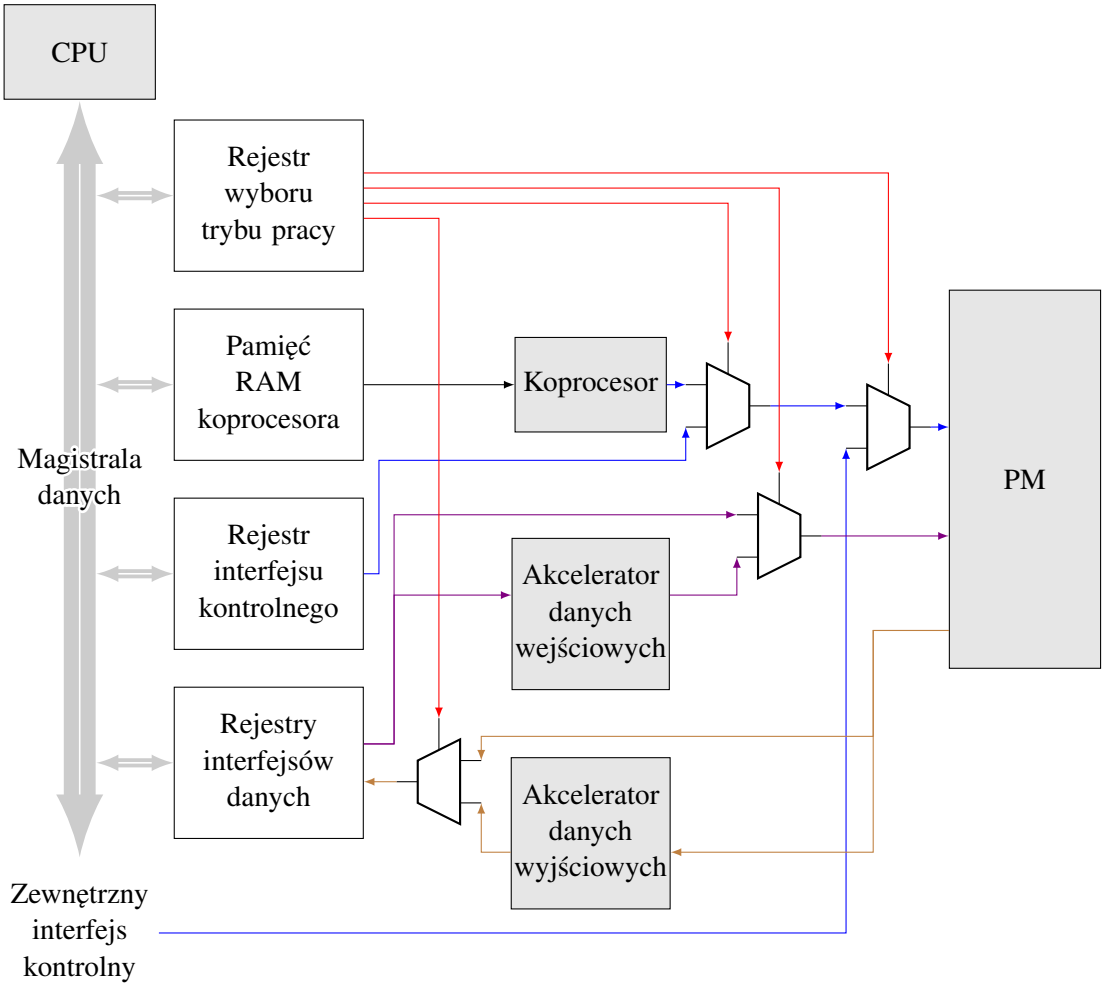
- `clk_sh` - każdy z pikseli PM zawierał w sobie dwa lokalne 16-bitowe rejestry przesuwne (A i B), które wraz z lokalnymi rejestrami przesuwnymi sąsiednich pikseli z tych samych kolumn, formowały globalne rejestry przesuwne. Sygnał `clk_sh` jest sygnałem zegarowym tych rejestrów,
- `sh_a` i `sh_b` - ustawienie wysokiego stanu logicznego na jednej z tych linii włączało przesuwanie danych przez wybrany globalny rejestr przesuwny PM (A albo B),
- `gate` - sygnał ten pozwalał na określenie, czy wyjście analogowego toru odczytowego powinno być podłączone do układu zliczającego impulsy. Ustawienie sygnału `gate` w stan wysoki powoduje włączenie rejestracji danych w pikselach,
- `strobe` - linia ta pozwalała na generację impulsów testowych,
- `store` - sygnał ten umożliwiał zapisywanie zawartości lokalnych rejestrów przesuwnych w lokalnych rejestrach konfiguracyjnych piksela.

Rejestry Kontrolno-Statusowe, Control and Status Register (CSR), PMC pozwalały mikroprocesorowi na sterowanie interfejsami PM za pośrednictwem standardowych instrukcji zapisu i odczytu. Ten generyczny mechanizm umożliwił pełną kontrolę nad PM, lecz nie zagwarantował parametrów niezbędnych do realizacji precyzyjnych pomiarów. W celu zwiększenia elastyczności i funkcjonalności PMC, zaimplementowane zostały cztery tryby pracy tego kontrolera:

- bezpośredni, w którym interfejs kontrolny i interfejsy danych PM były bezpośrednio podłączone do CSR,
- wykorzystujący koprocesor, w którym interfejs kontrolny PM był podłączony do dedykowanego koprocesora a interfejsy danych bezpośrednio do CSR,

- wykorzystujący koprocesor i akceleratory sprzętowe, w którym interfejs kontrolny był podłączony do koprocesora a interfejsy danych do dedykowanych akceleratorów sprzętowych,
- sterowania zewnętrznego, w którym interfejsy danych i wszystkie sygnały interfejsu kontrolnego poza *gate* oraz *strobe* podłączone były do CSR, a wymienione sygnały bezpośrednio do wyprowadzeń układu scalonego.

Wybór trybu pracy PMC realizowany był za pośrednictwem jednego z rejestrów CSR. Aktualizacja jego zawartości powodowała rekonfigurację muliplekserów zaimplementowanych wewnątrz kontrolera. Ogólna architektura PMC przedstawiona została na Rys. 3.3.




Rysunek 3.3: Ogólna architektura kontrolera układu odczytowego (PMC).

### Koprocesor kontrolera układu odczytowego

Jednym z najważniejszych wymagań stawianych układom odczytowym dla sensorów promieniowania jonizującego jest precyzja sterowania w czasie. Ustawianie interfejsu kontrolnego PM za pośrednictwem CSR wiąże się z opóźnieniami wynikającymi z konieczności przygotowania danej przez mikroprocesor przed jej wpisaniem do odpowiedniego rejestru. Biorąc pod uwagę również fakt, iż przebieg programu wykonywanego przez mikroprocesor może zostać zaburzony przez czynniki zewnętrzne, takie jak np. wystąpienie

przerwania, nie można być pewnym, że czas pomiędzy kolejnymi aktualizacjami CSR będzie stały. Zjawisko to negatywnie wpływa na rzetelność wykonywanych pomiarów, ponieważ może wprowadzić niezamierzone różnice w czasach rejestracji promieniowania. Problemy wynikające z braku determinizmu czasowego w algorytmach sterujących PM wykonywanych przez mikroprocesor mogą zostać załagodzone poprzez implementację dodatkowej programowalnej jednostki wykonawczej podłączonej do interfejsu kontrolnego PM.

Koprocesor Kontrolera Matrycy Pikseli, Pixel Matrix Controller Coprocessor (PMCC), był programowalnym układem elektronicznym, odpowiedzialnym za sterowanie interfejsem kontrolnym PM. Komponent ten umożliwiał precyzyjne, niezależne od mikroprocesora RISC-V, sterowanie matrycą, a jego działanie definiowane było za pośrednictwem kodu zapisywanego przez mikroprocesor w dedykowanej dwu-portowej pamięci RAM. Opracowany koprocesor wspierał następujący zbiór instrukcji:

1. `store` - ustawienie interfejsu kontrolnego PM,
2. `storeb` - ustawienie interfejsu kontrolnego PM i warunkowe wykonanie skoku,
3. `loop` - zdefiniowanie nowej pętli i zapisanie informacji o niej w dedykowanym kontrolerze pętli,
4. `jump` - bezwarunkowe wykonanie skoku,
5. `waitt` - wprowadzenie koprocesora w stan oczekiwania na wystąpienie sygnału wyzwalającego.

Znaczna część algorytmów sterujących PM wymagała wpisywania do interfejsu kontrolnego powtarzających się wartości. W celu redukcji rozmiaru kodu wpisywanego do pamięci RAM PMCC, w koprocesorze zaimplementowany został kontroler pętli, nadzorujący wykonywanie powtarzających się sekwencji sterujących. Przykładem wykorzystującego kontroler pętli programu PMCC jest przedstawiony na List.3.1 kod przesuwający dane przez jeden z globalnych rejestrów przesuwnych PM.

Listing 3.1: Program PMCC przesuwający dane przez globalny rejestr przesuwny A.

```
0xc0, 0x02, 0x00,   /* 0x000: store 0x0002: set shA, clear clkSh */
0x00,               /* 0x003: waitt */
0xc0, 0x03, 0x00,   /* 0x004: store 0x0003: set shA and clkSh */
0xc0, 0x02, 0x00,   /* 0x007: store 0x0002: set shA, clear clkSh */
0x20, 0x03          /* 0x00a: jump 0x003 */
```

Zaprojektowany koprocesor działał niezależnie od mikroprocesora RISC-V sterującego PMC. W celu zapewnienia synchronizacji pomiędzy programami wykonywanymi przez wspomniane jednostki wykonawcze, w PMCC zaimplementowany został mechanizm oczekiwania na sygnał wyzwalający generowany przez mikroprocesor poprzez wpis do CSR PMC. Rozwiązanie to zagwarantowało poprawną transmisję danych pomiędzy PM i mikroprocesorem RISC-V.

#### Sprzętowe akceleratory interfejsów danych

Wymiana danych pomiędzy PMC a PM realizowana była za pośrednictwem 32-bitowych interfejsów, których poszczególne bity podłączone były do odpowiednich kolumn PM. Oznacza to, że podczas wymiany

danych pomiędzy mikroprocesorem RISC-V a PM, każdy z transmitowanych bitów przenosił informację związaną z jedną kolumną. Właściwość ta sprawiała, że wszystkie przesyłane dane musiały być odpowiednio przetwarzane przez mikroprocesor przy użyciu operacji takich jak testowanie wartości, ustawianie i zerowanie poszczególnych bitów 32-bitowych słów. Eksperymenty wykonane przez Autora niniejszej rozprawy wykazały, że czas wykonywania przez mikroprocesor wspomnianych konwersji ma znaczny wpływ na czasy wykonywania odczytów i zapisów PM.

Konwerter Danych Matrycy Pikseli, Pixel Matrix Data Converter (PMDC), był układem elektronicznym przyśpieszającym transmisję danych pomiędzy mikroprocesorem RISC-V a PM. Komponent ten składał się z akceleratora danych wejściowych i akceleratora danych wyjściowych, podpiętych odpowiednio do interfejsu danych wejściowych i wyjściowych PM. Zadaniem tych układów była konwersja danych pomiędzy formatem wykorzystywanym przez mikroprocesor, a formatem wymaganym przez interfejsy danych PM.

### 3.1.5 Topografia układu scalonego

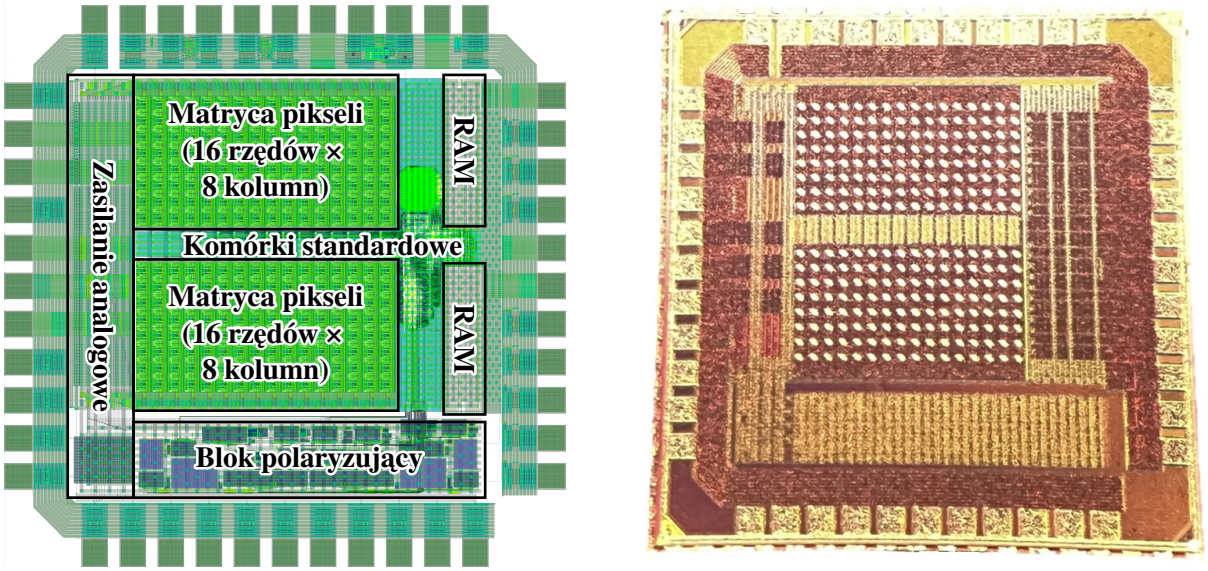
Prezentowany w niniejszej rozprawie odczytowy układ scalony z wbudowanym mikroprocesorem RISC-V został zaprojektowany i wykonany w 40-nanometrowym procesie CMOS. Synteza i implementacja tego układu wykonane zostały przy użyciu narzędzi Cadence Genus i Cadence Innovus, przy założeniu, że układ będzie taktowany sygnałem zegarowym o częstotliwości 50 MHz. Całkowity rozmiar układu wyniósł 1.73 mm × 1..73 mm, a w jego obrębie umieszczone zostały 44 wyprowadzenia, co przedstawione zostało na Rys. 3.4.

Ze względu na fakt, iż weryfikacja poprawności proponowanego rozwiązania nie wymagała łączenia odczytowego układu scalonego z sensorem, topografia układu została zoptymalizowana pod kątem zaplanowanych testów. Pierwszym z wykonanych w tym celu działań było dostosowanie kształtu układu i rozmieszczenia jego wyprowadzeń do wymagań obudowy JLCC44. Drugą istotną optymalizacją było podzielenie matrycy o rozmiarze 32 × 8 na dwa bloki o wymiarze 16 × 8, umożliwiające zmieszczenie w założonym obszarze matrycy o liczbie kolumn odpowiadającej szerokości słowa mikroprocesora.

## 3.2 Oprogramowanie

### 3.2.1 Ogólna architektura programowa

Oprogramowanie sterujące prezentowanym odczytowym układem scalonym zostało opracowane w formie trzech komponentów programowych: aplikacji inicjalizującej, aplikacji sterującej PM oraz bibliotek zawierających kod współdzielony pomiędzy aplikacjami. Wszystkie komponenty napisane zostały w języku C++, a relacje pomiędzy nimi zostały określone przy użyciu narzędzia CMake. Architektura programowa odczytowego układu scalonego z wbudowanym mikroprocesorem RISC-V została zaprojektowana przez Autora niniejszej rozprawy.




(a) Plan masek litograficznych.

(b) Fotografia mikroskopowa.

Rysunek 3.4: Zaprojektowany układ scalony.

### Współdzielone biblioteki

W skład oprogramowania sterującego prezentowanym układem scalonym wchodziły trzy współdzielone biblioteki. Pierwsza z nich przechowywała definicje klas reprezentujących poszczególne układy peryferyjne i umożliwiała interakcję z komponentami sprzętowymi opracowanego rozwiązania. Druga zawierała definicje funkcji manipulujących łańcuchami znakowymi i ułatwiających wysyłanie informacji diagnostycznych poprzez UART. Ostatnia z opracowanych bibliotek przechowywała definicje klas sterujących PM.

### Aplikacja inicjalizująca

Pierwszą instrukcją, wykonywaną przez mikroprocesor RISC-V bezpośrednio po wyjściu ze stanu zerującego, była instrukcja znajdująca się pod adresem `0x0000'0080`. Przy użyciu skryptu linkera i asemblera, pod adresem tym umieszczona została instrukcja skoku do napisanej w asemblerze procedury obsługi resetu. W czasie wykonywania tej procedury zerowane były rejestry robocze procesora, inicjalizowane były zmienne globalne i statyczne oraz wywoływane były konstruktory globalnych obiektów reprezentujących układy peryferyjne. Ostatnią wykonywaną przez tę procedurę operacją był wywołanie funkcji `main` napisanej w C++.

Funkcja `main` aplikacji inicjalizującej rozpoczynała swoje działanie od konfiguracji kontrolera UART, wykorzystywanego do transmisji informacji diagnostycznych. Następnie, przy użyciu kontrolera GPIO, określała układ peryferyjny, który wykorzystywany był do załadowania aplikacji użytkownika. Wykrycie wysokiego stanu logicznego na sprawdzanym wyprowadzeniu cyfrowym ogólnego przeznaczenia powodowało wejście programu do pętli, w której mikroprocesor oczekiwał na dane wysyłane poprzez UART przez układ nadrzędny, taki jak np. komputer osobisty. Odczytanie z tego wyprowadzenia niskiego stanu logicznego, uruchamiało z kolei procedurę odczytu kodu aplikacji z karty SD. Niezależnie od wybranego źródła,

po zapisaniu kodu aplikacji w odpowiedniej pamięci RAM, mikroprocesor wykonywał skok do pierwszej instrukcji załadowanego programu.

#### Aplikacja sterująca

Aplikacja sterująca układem odczytowym została zrealizowana w formie interpretera komend. Sterowanie nią odbywało się za pośrednictwem poleceń przesyłanych przez UART. Wśród operacji wspieranych przez tę aplikację znajdowały sie, między innymi zapis konfiguracji do pikseli, odczyt wyników rejestracji promieniowania i wybór trybu pracy PMC.

### 3.2.2 Algorytmy sterujące matrycą pikseli

Głównym celem integracji we wspólnym podłożu krzemowym mikroprocesora RISC-V i PM, było umożliwienie wykonywania algorytmów sterujących PM wewnątrz układu scalonego. Aby zdiagnozować potencjalne błędy sprzętowe, których naprawienie nie byłoby możliwe po wyprodukowaniu układu, algorytmy zostały zweryfikowane przy użyciu symulatora *Cadence Xcelium* oraz prototypu zaimplementowanego w FPGA. Wykonane podczas testów pomiary wydajności pozwoliły na wskazanie fragmentów algorytmów, które mogłoby zostać zoptymalizowane poprzez wykorzystanie dedykowanych akceleratorów sprzętowych.

Przedstawione w niniejszej podsekcji liczby cykli zegara potrzebnych na wykonanie poszczególnych algorytmów zostały zmierzone za pośrednictwem układu czasowo-licznikowego wbudowanego w mikroprocesor, podczas eksperymentów wykonanych przy użyciu symulatora *Cadence Xcelium*. Podane w tablicach czasy wykonywania zostały wyliczone dla układu taktowanego zegarem o częstotliwości 50 MHz. Ze względu na fakt, iż tryb sterowania zewnętrznego PMC wykorzystywany był jedynie na sterowanie sygnałami `gate` oraz `strobe`, tryb ten nie był uwzględniany w porównaniach algorytmów sterujących PM.

#### Odczyt matrycy pikseli

Jednym z najczęściej wykonywanych algorytmów sterujących PM jest odczyt zawartości jej pikseli. Procedura ta polega na wielokrotnie powtarzanej generacji narastającego zbocza sygnału kontrolnego `clk_sh`, przeplatanej odczytem interfejsu danych PM. Analiza przebiegu tego algorytmu wykazała, że najbardziej czasochłonnym jego fragmentem jest konwersja danych opisana w 3.1.4. Wyniki symulacji wykonanych dla wybranych trybów pracy PMC zostały przedstawione w Tab. 3.1.

Tablica 3.1: Wyniki analizy algorytmu odczytującego matrycę pikseli.

| Tryb pracy PMC | Cykle zegara | Czas [µs] | Różnica względem trybu bezpośredniego [%] |
|---|---|---|---|
| Bezpośredni | 103343 | 2066,9 | 0,0 |
| Wykorzystujący koprocesor | 103275 | 2065,5 | −0,1 |
| Wykorzystujący koprocesor i akceleratory | 3272 | 65,4 | −96,8 |

Przeprowadzony eksperyment wykazał, że wybór trybu pracy PMC ma znaczący wpływ na czas odczytu PM. Wykorzystanie koprocesora i sprzętowego akceleratora interfejsu danych wyjściowych po-

zwala na skrócenie czasu potrzebnego na odczyt zawartości PM o 96 % względem czasu potrzebnego na wykonanie tej procedury przy użyciu bezpośredniego trybu pracy PMC.

### Zapis matrycy pikseli

Kolejnym często wykonywanym algorytmem sterującym PM jest zapis danych do jej pikseli. Procedura ta, podobnie jak algorytm odczytowy, polega na wielokrotnie powtarzanej generacji narastającego zbocza sygnału kontrolnego `clk_sh`, z tą różnicą, że zamiast odczytem interfejsu wyjściowego PM, zmiany sygnału kontrolnego przeplatane są zapisem jej interfejsu wejściowego. Wyniki symulacji wykonanych dla wybranych trybów pracy PMC zostały przedstawione w Tab. 3.2.

Tablica 3.2: Wyniki analizy algorytmu zapisującego matrycę pikseli.

| Tryb pracy PMC | Cykle zegara | Czas [µs] | Różnica względem trybu bezpośredniego [%] |
|---|---|---|---|
| Bezpośredni | 108958 | 2179,2 | 0,0 |
| Wykorzystujący koprocesor | 112748 | 2255,0 | +3,5 |
| Wykorzystujący koprocesor i akceleratory | 3383 | 67,7 | −96,9 |

Przeprowadzony eksperyment wykazał, że podobnie jak w przypadku odczytu zawartości pikseli, wybór trybu pracy PMC ma znaczący wpływ na czas zapisu danych do PM. Wykorzystanie koprocesora i sprzętowego akceleratora interfejsu danych wejściowych pozwala na skrócenie czasu potrzebnego na zapis danych do PM o 96 %, względem czasu potrzebnego na wykonanie tej procedury przy użyciu bezpośredniego trybu pracy PMC.

### Kalibracja analogowego toru odczytowego

Wytwarzanie układów półprzewodnikowych jest zaawansowanym technologicznie procesem, pozwalającym na produkcję urządzeń, których rozmiary wyrażane są w nanometrach. Mimo że, wykorzystywane w tym procesie narzędzia są urządzeniami o wysokiej precyzji, w półprzewodnikowych układach scalonych zaobserwować można zjawisko niedopasowania, czyli występowania rozrzutu parametrów fizycznych pomiędzy komponentami o takich samych projektowanych wymiarach geometrycznych. W wyniku tego zjawiska, poszczególne instancje tego samego układu analogowego mogą posiadać różne charakterystyki działania.

Występowanie zjawiska niedopasowania w układach odczytowych dla detektorów promieniowania jonizującego zaburza działanie analogowych torów odczytowych zlokalizowanych w poszczególnych pikselach. Właściwość ta sprawia, iż każdy z pikseli naświetlanych wiązką jednorodnego promieniowania rejestrować może inną liczbę cząstek. W celu ograniczenia negatywnych skutków tego zjawiska wewnątrz pikseli, umieszczane są komponenty dostrajające, takie jak przetworniki cyfrowo-analogowe, pozwalające na kompensację wpływu rozrzutu parametrów fizycznych na napięcia elektryczne występujące w poszczególnych punktach toru odczytowego.

Algorytmem sterującym PM redukującym wpływ rozrzutu komponentów na działanie analogowych torów odczytowych jest kalibracja. W prezentowanym układzie odczytowym, algorytm ten jest procedurą iteracyjną, podczas której do każdego z pikseli PM wpisywane są kolejne wartości korygujące, po zapisaniu których wykonywane są rejestracje sygnałów szumowych. W wyniku tego algorytmu do lokalnych rejestrów konfiguracyjnych poszczególnych pikseli wpisywane są wartości, które umożliwiły zarejestrowanie największej liczby impulsów szumowych, co odpowiada optymalnej wartości wejściowej przetwornika DAC korygującego napięcie niezrównoważenia dyskryminatora amplitudy impulsu.

Wyniki symulacji wykonanych dla wybranych trybów pracy PMC zostały przedstawione w Tab. 3.3.

Tablica 3.3: Wyniki analizy algorytmu kalibrującego analogowy tor odczytowy.

| Tryb pracy PMC | Cykle zegara | Czas [s] | Różnica względem trybu bezpośredniego [%] |
|---|---|---|---|
| Bezpośredni | 2473266017 | 49,5 | 0,0 |
| Wykorzystujący koprocesor | 2857419105 | 57,1 | +15,4 |
| Wykorzystujący koprocesor i akceleratory | 689339429 | 13,8 | −72,1 |

Algorytm kalibrujący analogowe tory odczytowe jest procedurą iteracyjną składającą się z wielokrotnie powtarzanych zapisów i odczytów PM. Przeprowadzony eksperyment wykazał, że tak jak w przypadku uprzednio wspomnianych algorytmów, wykorzystanie koprocesora i sprzętowych akceleratorów interfejsów danych pozwala na znaczne, bo sięgające 72.1 %, skrócenie czasu wykonywania procedur, względem czasu potrzebnego na wykonanie tej procedury przy użyciu bezpośredniego trybu pracy PMC.

### Transmisja podobszaru matrycy

Jednym z wyzwań, z którymi mierzyć muszą się projektanci układów odczytowych dla detektorów promieniowania jonizującego, jest ograniczona powierzchnia układów scalonych. Restrykcja ta bezpośrednio wpływa na ilość miejsca, które może zostać przeznaczone na implementację pamięci buforujących dane odczytywane z PM. Istotnym z punktu widzenia ogólnej wydajności układu odczytowego jest więc czas transmisji danych na zewnątrz układu scalonego, determinujący szybkość opróżniania dostępnych buforów, a co za tym idzie maksymalną częstość zapisywania w nich danych uzyskiwanych w wyniku kolejnych rejestracji promieniowania.

Istnieje wiele metod redukcji czasu potrzebnego na transmisję danych ze scalonego układu odczytowego do zewnętrznego systemu nadzorującego. Jedną z nich jest ograniczenie liczby transmitowanych pikseli poprzez określenie podobszarów PM, których zawartości są niezbędne do analizy wyników rejestracji. W Tab. 3.4 przedstawione zostały wyniki analizy algorytmu przesyłającego wybrane podobszary PM za pośrednictwem kontrolera UART transmitującego dane z szybkością 115 200 B. W tablicy zamieszczone zostały również różnice liczby cykli zegara potrzebnych na transmisję wybranych podregionów, względem liczby cykli potrzebnych na przesłanie całej matrycy. Referencyjna liczba cykli zegara potrzebnych na transmisję całej matrycy wynosiła 991 194.

Tablica 3.4: Wyniki analizy algorytmu transmitującego podobszar PM.

| Obszar | Piksele | Cykle zegara | Czas [µs] | Różnica [%] |
|---|---|---|---|---|
| (0, 0) - (0, 0) | 1 | 2335 | 46,7 | −99,8 |
| (0, 0) - (1, 1) | 4 | 11212 | 224,2 | −98,9 |
| (0, 0) - (2, 2) | 9 | 26675 | 533,5 | −97,3 |
| (0, 0) - (3, 3) | 16 | 48724 | 974,5 | −95,1 |
| (0, 0) - (4, 4) | 25 | 83467 | 1669,3 | −91,6 |
| (0, 0) - (5, 5) | 36 | 126832 | 2536,6 | −87,2 |
| (0, 0) - (6, 6) | 49 | 178819 | 3576,4 | −82,0 |
| (0, 0) - (7, 7) | 64 | 239428 | 4788,6 | −75,8 |

Przeprowadzony eksperyment wykazał, że gdy do analizy wyników rejestracji potrzebne są jedynie dane pochodzące z podobszaru matrycy, częstość rejestracji promieniowania może zostać znacznie zwiększona poprzez transmisję do układu nadrzędnego jedynie wartości odczytanych z pikseli wchodzących w skład określonego podobszaru. Czas transmisji podobszaru PM jest wprost proporcjonalny do liczby transmitowanych pikseli i niezależnie od ich liczby, jest mniejszy od czasu potrzebnego na transmisję całej matrycy.

#### Transmisja danych z pikseli spełniających zadane kryterium

Kolejną metodą redukcji czasu potrzebnego na przesłanie danych ze scalonego układu odczytowego do zewnętrznego systemu nadzorującego jest transmisja jedynie danych spełniających zadane kryterium. W Tab. 3.5 przedstawione zostały wyniki analizy algorytmu transmitującego do układu nadrzędnego dane pochodzące z pikseli, z których odczytane zostały wartości większe od uprzednio zdefiniowanego progu. Podobnie jak w przypadku transmisji podobszaru PM, dane były transmitowane za pośrednictwem interfejsu UART o szybkości 115 200 B, z tą różnicą, że wraz z wartościami odczytywanymi z pikseli transmitowane były ich współrzędne. W tablicy zamieszczone zostały również różnice liczby cykli zegara potrzebnych na transmisję wybranych pikseli, względem liczby cykli potrzebnych na przesłanie całej matrycy. Referencyjna liczba cykli zegara potrzebnych na transmisję całej matrycy wynosiła 991 194.

Przeprowadzony eksperyment wykazał, że wykorzystanie mikroprocesora RISC-V do transmisji danych z pikseli spełniających zadane kryterium może znacznie skrócić czas potrzebny na transmisję danych odczytywanych z PM, a w efekcie zwiększyć częstość rejestracji promieniowania. Jednakże, korzyść ze stosowania tego rozwiązania występuje jedynie wtedy, gdy zadane kryterium spełnione jest przez mniej niż połowę pikseli, co wynika z faktu, iż w celu zapewnienia poprawnej interpretacji danych, wraz z wartością odczytaną z piksela przetransmitowane zostać muszą jego współrzędne.

## 3.3 Zaprojektowany mikroprocesor RISC-V

Analiza kodu algorytmów sterujących PM wykazała, iż do ich realizacji wymagane są jedynie instrukcje całkowitoliczbowe, zdefiniowane w zbiorze instrukcji określonym nazwą kodową RV32I. Autor rozprawy

Tablica 3.5: Wyniki analizy algorytmu transmitującego dane z pikseli spełniających zadane kryterium.

| Piksele | Cykle zegara | Czas [µs] | Różnica [%] |
|---|---|---|---|
| 1 | 16584 | 331,7 | −98,3 |
| 2 | 27392 | 547,8 | −97,2 |
| 4 | 48966 | 979,3 | −95,1 |
| 8 | 92114 | 1842,3 | −90,7 |
| 16 | 178704 | 3574,1 | −82,0 |
| 32 | 340914 | 6818,3 | −65,6 |
| 64 | 676472 | 13529,4 | −31,8 |
| 128 | 1347084 | 26941,7 | +35,9 |
| 256 | 2577262 | 51545,2 | +160,0 |

postanowił zaprojektować mikroprocesor, którego funkcjonalność zostałaby ograniczona do wykonywania instrukcji zawartych we wspomnianym zbiorze, zastąpić dotychczas wykorzystywany mikroprocesor autorskim rozwiązaniem, powtórzyć eksperymenty opisane w Podsekcji 3.2.2 i porównać wyniki przeprowadzonych eksperymentów. Założony cel został zrealizowany, a ogólna architektura sprzętowa zaprojektowanego mikroprocesora oraz wyniki wykonanych eksperymentów zostały przedstawione odpowiednio w Podsekcjach 3.3.1 i 3.3.2.

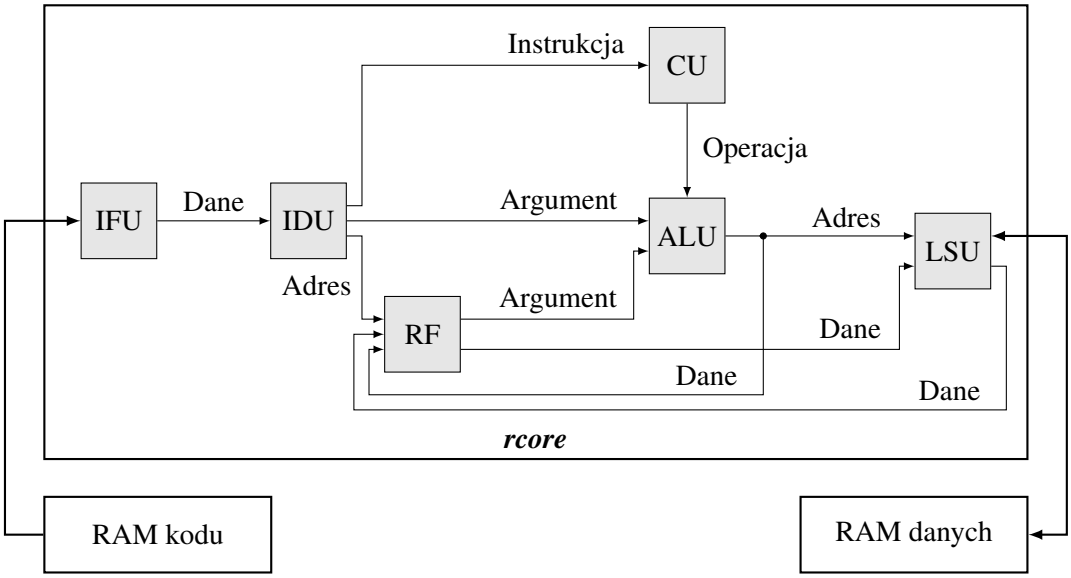
### 3.3.1 Ogólna architektura sprzętowa

*rcore* jest 32-bitowym mikroprocesorem o architekturze harwardzkiej. Układ ten został zaimplementowany w języku SystemVerilog, a poprawność jego działania została zweryfikowana przy użyciu symulatora i prototypu zaimplementowanego w FPGA. W skład ogólnej architektury sprzętowej tego mikroprocesora wchodzą następujące komponenty:

- Instruction Fetch Unit (IFU), komponent sterujący magistralą instrukcji i przekazujący do kolejnych modułów informację o kolejnej instrukcji oczekującej na wykonanie,
- Instruction Decode Unit (IDU), moduł dekodujący dane dostarczane przez IFU i przekazujący pozostałym komponentom informację o tym, jaka instrukcja powinna zostać wykonana i jakie dane powinny być jej argumentami,
- Arithmetic Logic Unit (ALU), komponent wykonujący operacje arytmetyczne i logiczne wymagane do realizacji wybranych instrukcji,
- Register File (RF), moduł grupujący rejestry robocze procesora wykorzystywane do przechowywania danych tymczasowych, takich jak wyniki operacji wykonywanych przez ALU,
- Load-Store Unit (LSU), komponent sterujący magistralą danych i nadzorujący wymianę danych z pamięciami i układami peryferyjnymi,

- Control Unit (CU), moduł nadrzędny sterujący działaniem pozostałych komponentów.

Uproszczony schemat mikroarchitektury zaprojektowanego mikroprocesora RISC-V został przedstawiony na Rys. 3.5.




Rysunek 3.5: Uproszczony schemat mikroarchitektury zaprojektowanego microprocesora RISC-V.

### 3.3.2 Porównanie algorytmów sterujących matrycą pikseli

Architektura opracowanego układu odczytowego dla sensorów promieniowania jonizującego, umożliwiła zastąpienie wykorzystywanego dotąd rdzenia Ibex [39] zaprojektowanym mikroprocesorem i powtórzenie eksperymentów weryfikujących algorytmy sterujące PM. Jednakże, różnice w budowie wykorzystywanych mikroprocesorów wymusiły wykonanie następujących zmian w skonstruowanym środowisku testowym:

1. ze względu na fakt, iż zaprojektowany mikroprocesor nie wspierał instrukcji skompresowanych i nie posiadał sprzętowej mnożarki, zmodyfikowana została flaga kompilatora wykorzystywana do generacji plików binarnych (flaga `-march=rv32imc` została zastąpiona flagą `-march=rv32i`),

2. z powodu braku układu czasowo-licznikowego wbudowanego w zaprojektowany mikroprocesor, do pomiaru liczb cykli zegara potrzebnych na wykonanie poszczególnych algorytmów wykorzystany został układ czasowo-licznikowy podłączony do magistrali danych.

W celu rzetelnego porównania wyników uzyskanych dla poszczególnych mikroprocesorów i wyeliminowania różnic wynikających ze zmiany wspomnianej flagi kompilatora, eksperymenty symulacyjne zostały wykonane zarówno dla zaprojektowanego mikroprocesora, jak i dla układu Ibex. Rezultaty wykonanych eksperymentów zostały przedstawione w Tab. 3.6, Tab. 3.7, Tab. 3.8 i Tab. 3.9.

Wykonane eksperymenty wykazały, że liczby cykli zegara potrzebnych na wykonanie poszczególnych algorytmów zmniejszyły się o ok. 30 % po zastąpieniu układu Ibex zaprojektowanym mikroprocesorem. Porównanie przebiegów cyfrowych wygenerowanych podczas testu odczytu PM wykazało, że w przypadku

Tablica 3.6: Zestawienie liczb cykli zegara potrzebnych do odczytu PM dla różnych mikroprocesorów.

| Tryb pracy PMC | Ibex | rcore | Różnica [%] |
|---|---|---|---|
| Bezpośredni | 108764 | 74385 | −31,6 |
| Wykorzystujący koprocesor | 116674 | 79196 | −32,1 |
| Wykorzystujący koprocesor i akceleratory | 3625 | 2508 | −30,8 |

Tablica 3.7: Zestawienie liczb cykli zegara potrzebnych do zapisu PM dla różnych mikroprocesorów.

| Tryb pracy PMC | Ibex | rcore | Różnica [%] |
|---|---|---|---|
| Bezpośredni | 120244 | 78580 | −34,6 |
| Wykorzystujący koprocesor | 126130 | 81874 | −35,1 |
| Wykorzystujący koprocesor i akceleratory | 3703 | 2611 | −29,5 |

zaprojektowanego mikroprocesora wykonanie najkrótszej instrukcji trwało jeden takt zegara systemowego, podczas gdy w przypadku rdzenia Ibex czas ten był dwukrotnie dłuższy. Dokładniejsza analiza problemu wykazała, że przyczyną różnic w czasach wykonywania instrukcji były różnice w budowie modułów pełniących rolę arbitrów magistral instrukcji.

W opracowanym przez Autora niniejszej rozprawy układzie zarządzającym magistralą instrukcji mikroprocesora Ibex, zaimplementowana została maszyna stanów kontrolująca odczyt instrukcji. Moduł ten uwzględniał, że zarówno wystawiane przez kontroler pamięci potwierdzenie przyjęcia żądania (`core_instr_bus.gnt`), jak i potwierdzenie umieszczenia przez ten moduł na magistrali instrukcji zażądanej danej (`core_instr_bus.rvalid`), mogą nie być ustawiane natychmiast po wystawieniu przez rdzeń żądania odczytu instrukcji (`core_instr_bus.req`). Jednakże, w celu skrócenia czasu implementacji tego modułu, odroczona została obsługa sytuacji, w której przekazywanie do rdzenia odczytanej instrukcji i przesyłanie do pamięci kodu żądania odczytu instrukcji kolejnej, realizowane byłyby symultanicznie. Niestety, ze względu na napięty harmonogram prac projektowych, późniejsza implementacja tej funkcjonalności stała się niemożliwa. Na List. 3.2 przedstawiony został fragment definicji arbitra magistrali instrukcji rdzenia Ibex, prezentujący maszynę stanów obsługującą odczyt instrukcji z pamięci kodu.

Tablica 3.8: Zestawienie liczb cykli zegara potrzebnych do transmisji podobszaru PM dla różnych mikroprocesorów.

| Obszar | Piksele | Ibex | rcore | Różnica [%] |
|---|---|---|---|---|
| (0, 0) - (0, 0) | 1 | 7589 | 5040 | −33,6 |
| (0, 0) - (1, 1) | 4 | 32422 | 22206 | −31,5 |
| (0, 0) - (2, 2) | 9 | 74717 | 51636 | −30,9 |
| (0, 0) - (3, 3) | 16 | 134598 | 93404 | −30,6 |
| (0, 0) - (4, 4) | 25 | 217741 | 153370 | −29,6 |
| (0, 0) - (5, 5) | 36 | 320492 | 227696 | −29,0 |
| (0, 0) - (6, 6) | 49 | 442987 | 316476 | −28,6 |
| (0, 0) - (7, 7) | 64 | 584932 | 419530 | −28,3 |

Tablica 3.9: Zestawienie liczb cykli zegara potrzebnych do transmisji przefiltrowanych danych dla różnych mikroprocesorów.

| Piksele | Ibex | rcore | Różnica [%] |
|---|---|---|---|
| 1 | 34364 | 24459 | −28,8 |
| 2 | 61282 | 43691 | −28,7 |
| 4 | 115246 | 82231 | −28,6 |
| 8 | 223134 | 159291 | −28,6 |
| 16 | 438638 | 313243 | −28,6 |
| 32 | 858726 | 610487 | −28,9 |
| 64 | 1709052 | 1215179 | −28,9 |
| 128 | 3410042 | 2424721 | −28,9 |
| 256 | 6688608 | 4725195 | −29,4 |

Listing 3.2: Fragment definicji arbitra magistrali instrukcji rdzenia Ibex, prezentujący maszynę stanów obsługującą odczyt instrukcji z pamięci kodu.

```
91  always_ff @(posedge clk or negedge rst_n) begin
92      if (!rst_n)
93          state <= IDLE;
94      else
95          state <= state_nxt;
96  end
97
98  always_comb begin
99      state_nxt = state;
100
101     case (state)
102     IDLE: begin
```

```
103         if (core_instr_bus.req) begin
104             state_nxt = core_instr_bus.gnt ?
105                 WAITING_FOR_RESPONSE : WAITING_FOR_GRANT;
106         end
107     end
108     WAITING_FOR_GRANT: begin
109         if (core_instr_bus.gnt)
110             state_nxt = WAITING_FOR_RESPONSE;
111     end
112     WAITING_FOR_RESPONSE: begin
113         if (core_instr_bus.rvalid)
114             state_nxt = IDLE;
115     end
116     endcase
117 end
```

Ze względu na fakt, iż przedstawione ograniczenie arbitra magistrali instrukcji dostrzeżone zostało po wysłaniu projektu układu scalonego do fabryki, występuje ono w wyprodukowanym układzie. Mimo tego, że poprawienie wydajności układu scalonego nie było możliwe, podjęta została decyzja o zastąpieniu arbitra magistrali mikroprocesora Ibex jego odpowiednikiem zaprojektowanym dla układu rcore, symulacyjnym pomiarze czasu wykonywania algorytmów sterujących PM i zestawieniu uzyskanych wyników z wynikami wcześniejszymi. W Tab. 3.10, Tab. 3.11, Tab. 3.12 i Tab. 3.13 przedstawione zostały zestawienia uzyskanych wyników.

Tablica 3.10: Zestawienie liczb cykli zegara potrzebnych do odczytu PM dla różnych układów zarządzających magistralą instrukcji rdzenia Ibex.

| Tryb pracy PMC | Układ oryginalny | Układ poprawiony | Różnica [%] |
|---|---|---|---|
| Bezpośredni | 108764 | 63367 | −41,7 |
| Wykorzystujący koprocesor | 116674 | 67755 | −41,9 |
| Wykorzystujący koprocesor i akceleratory | 3625 | 2256 | −37,8 |

Tablica 3.11: Zestawienie liczb cykli zegara potrzebnych do zapisu PM dla różnych układów zarządzających magistralą instrukcji rdzenia Ibex.

| Tryb pracy PMC | Układ oryginalny | Układ poprawiony | Różnica [%] |
|---|---|---|---|
| Bezpośredni | 120244 | 63215 | −47,4 |
| Wykorzystujący koprocesor | 126130 | 66337 | −47,4 |
| Wykorzystujący koprocesor i akceleratory | 3703 | 2227 | −39,9 |

Przeprowadzone eksperymenty wykazały, że modyfikacja modułu zarządzającego magistralą instrukcji mikroprocesora Ibex może znacząco zmniejszyć czas wykonywania algorytmów sterujących PM. Najwięk-

Tablica 3.12: Zestawienie liczb cykli zegara potrzebnych do transmisji podobszaru PM dla różnych układów zarządzających magistralą instrukcji rdzenia Ibex.

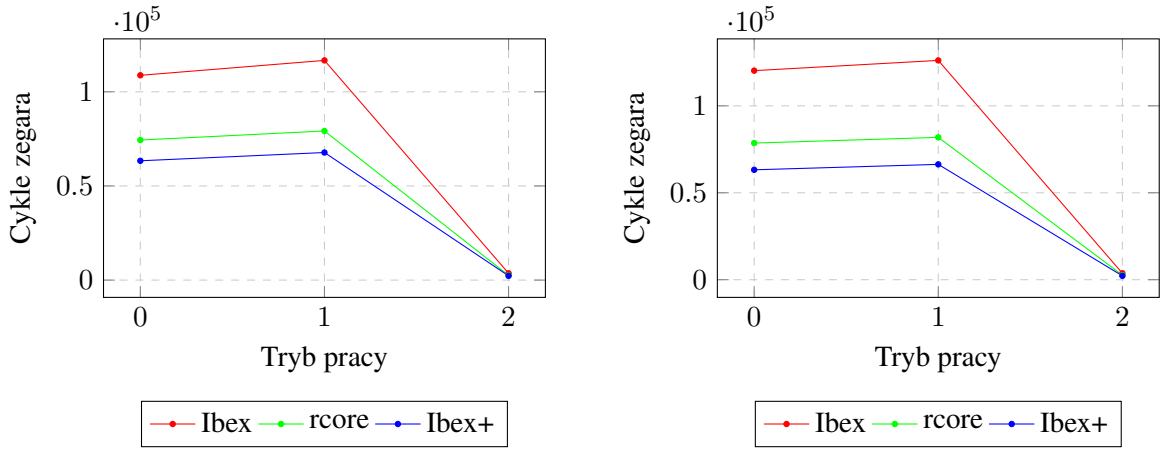
| Obszar | Piksele | Układ oryginalny | Układ poprawiony | Różnica [%] |
|---|---|---|---|---|
| (0, 0) - (0, 0) | 1 | 7589 | 4347 | −42,7 |
| (0, 0) - (1, 1) | 4 | 32422 | 19405 | −40,1 |
| (0, 0) - (2, 2) | 9 | 74717 | 45319 | −39,3 |
| (0, 0) - (3, 3) | 16 | 134598 | 82085 | −39,0 |
| (0, 0) - (4, 4) | 25 | 217741 | 135520 | −37,8 |
| (0, 0) - (5, 5) | 36 | 320492 | 201811 | −37,0 |
| (0, 0) - (6, 6) | 49 | 442987 | 281158 | −36,5 |
| (0, 0) - (7, 7) | 64 | 584932 | 373051 | −36,2 |

Tablica 3.13: Zestawienie liczb cykli zegara potrzebnych do transmisji przefiltrowanych danych dla różnych układów zarządzających magistralą instrukcji rdzenia Ibex.

| Piksele | Układ oryginalny | Układ poprawiony | Różnica [%] |
|---|---|---|---|
| 1 | 34364 | 21511 | −37,4 |
| 2 | 61282 | 38598 | −37,0 |
| 4 | 115246 | 72737 | −36,9 |
| 8 | 223134 | 141292 | −36,7 |
| 16 | 438638 | 278035 | −36,6 |
| 32 | 858726 | 541361 | −37,0 |
| 64 | 1709052 | 1077953 | −36,9 |
| 128 | 3410042 | 2150976 | −36,9 |
| 256 | 6688608 | 4182713 | −37,5 |

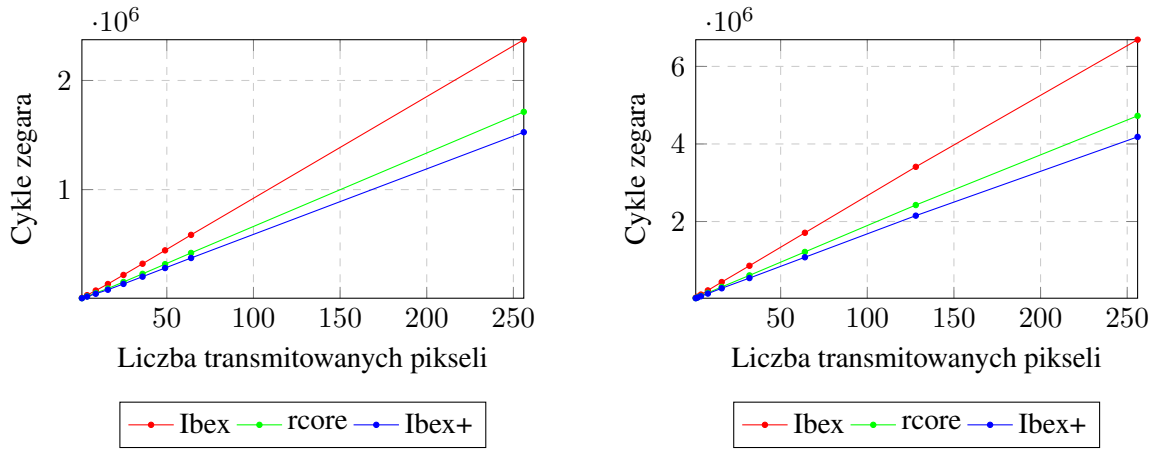
sza różnica zaobserwowana została podczas testu zapisu danych do matrycy i w przypadku zapisów wykonywanych w trybie bezpośrednim PMC i w trybie wykorzystujacym koprocesor sięgała 47.4 %. Wszystkie czasy wykonywania algorytmów zmierzone dla rdzenia Ibex z poprawionym układem zarządzającym magistralą instrukcji były o co najmniej 10 % mniejsze od czasów zmierzonych dla mikroprocesora rcore. Na Rys. 3.6a, Rys. 3.6a Rys. 3.7a i Rys. 3.7b przedstawione zostały wyniki zmierzone dla testowanych systemów mikroprocesorowych. Etykietą „Ibex+" oznaczony został system mikroprocesorowy, w którym rdzeń Ibex podłączony został do pamięci kodu za pośrednictwem zmodyfikowanego modułu zarządzającego magistralą instrukcji.

Wszystkie przedstawione w niniejszym rozdziale eksperymenty zostały zrealizowane przy wykorzystaniu symulatora oraz prototypu zaimplementowanego w FPGA. Opracowany prototyp wchodził w skład środowiska do testowania odczytowych układów scalonych, skonstruowanego przez Autora niniejszej rozprawy. Budowa środowiska testowego pozwoliła nie tylko na przeprowadzenie eksperymentów wykorzystujących wspomniany prototyp, ale również na ich wykonanie w wyprodukowanym układzie scalonym.




(a) Odczyt PM. (b) Zapis PM.

Rysunek 3.6: Zestawienie liczb cykli zegara potrzebnych do wykonania wybranych algorytmów dla różnych trybów pracy PMC, dla różnych systemów mikroprocesorowych (Oznaczenia trybów pracy PMC na wykresie: 1 - bezpośredni, 2 - wykorzystujący koprocesor, 3 - wykorzystujący koprocesor i akceleratory).




(a) Transmisja podobszaru. (b) Filtracja danych.

Rysunek 3.7: Zestawienie liczb cykli zegara potrzebnych do wykonania wybranych algorytmów dla różnych systemów mikroprocesorowych.

# Rozdział 4

# Środowisko do testowania odczytowych układów scalonych

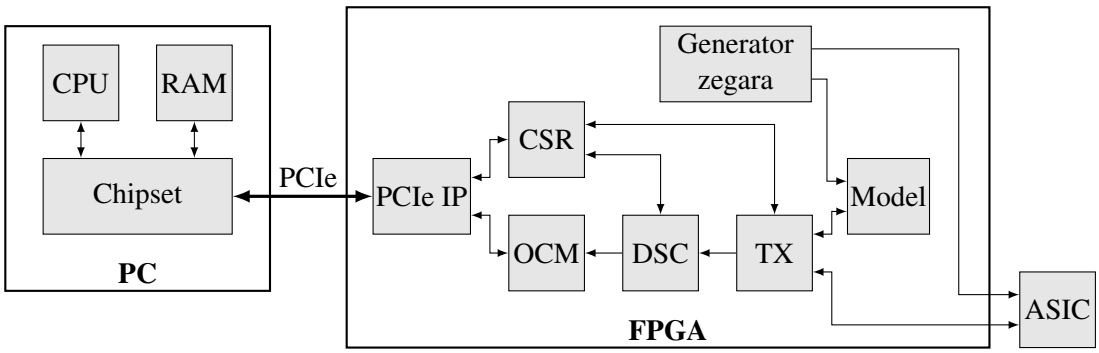
Odczytowe układy scalone są urządzeniami elektronicznymi zawierającymi analogowe tory odczytowe, których parametry mogą różnić się zarówno między egzemplarzami układu, jak i pomiędzy poszczególnymi ich instancjami wewnątrz jednego układu scalonego. Weryfikacja poprawności analogowych torów odczytowych wiąże się z koniecznością wykorzystania środowiska testowego kompatybilnego z konkretnym układem odczytowym. Prezentowany w niniejszej rozprawie scalony system mikroprocesorowy dla hybrydowych detektorów promieniowania jonizującego został przetestowany przy użyciu dedykowanego środowiska testowego, opracowanego przez Autora niniejszej rozprawy.

## 4.1 Architektura sprzętowa

Zaprojektowane przez Autora niniejszej rozprawy środowisko do testowania odczytowych układów scalonych zostało zbudowane z dwóch komponentów sprzętowych: układu FPGA i komputera osobistego, Personal Computer (PC). Pierwszym z nich była płyta deweloperska *Intel Arria 10 GX* [42], wyposażona w układ programowalny *10AX115S2F45I1SG*, z rodziny *Intel Arria 10 GX 1150 FPGA* [43]. Drugim był komputer klasy PC, w skład którego wchodziły, między innymi mikroprocesor *Intel(R) Core(TM) i3-6300 [44]*, 8 GB RAM i dysk Solid State Drive (SSD), o pojemności 256 GB. Przedstawione komponenty zostały połączone za pośrednictwem 8-liniowego interfejsu PCIe trzeciej generacji, a ich działanie było nadzorowane przez *Linuksa 5.17.9* [45] wchodzącego w skład dystrybucji *Fedora Workstation 36* [46]. Ogólna architektura zaprojektowanego środowiska do testowania odczytowych układów scalonych została przedstawiona na Rys. 4.1. W dalszej części rozdziału zostaną szczegółowo opisane główne interfejsy pomiędzy PC, FPGA i układem scalonym, oraz komponenty zaimplementowane w FPGA:

- PCIe IP – blok użyty do konfiguracji interfejsu PCIe,
- CSR – zbiór rejestrów kontrolno-statusowych, umożliwiających sterowanie działaniem układu i monitorowanie jego stanu,
- OCM – pamięć zlokalizowana wewnątrz układu scalonego,

- DSC – komponent kontrolujący zapis danych odczytywanych z testowanego układu scalonego,
- TX – moduł nadawczo-odbiorczy sterujący transmisją danych pomiędzy FPGA i testowanym układem,
- generator zegara – komponent generujący zegar wykorzystywany do taktowania testowanego układu,
- model – wykorzystywany w celach diagnostycznych model testowanego układu.




Rysunek 4.1: Ogólna architektura środowiska do testowania odczytowych układów scalonych.

### 4.1.1 Układ cyfrowy zaimplementowany w FPGA

#### Konfiguracja interfejsu PCIe

Komunikacja pomiędzy PC nadzorującym działanie skonstruowanego środowiska testowego, a FPGA pełniącym rolę interfejsu testowanego układu scalonego, została zrealizowana za pośrednictwem interfejsu PCIe x8 Gen 3. Konfiguracja złącza krawędziowego i powiązanych z nim modułów nadawczo-odbiorczych została wykonana przy użyciu dedykowanego komponentu Intellectual Property (IP), udostępnianego pod nazwą *Intel Arria 10/Cyclone 10 Hard IP for PCI Express* [47] przez środowisko projektowe *Intel® Quartus® Prime* [48].

Wspomniane IP pozwoliło na zdefiniowanie dwóch regionów pamięci, Base Address Register (BAR), udostępnianych systemowi operacyjnemu. Pierwszy z nich umożliwiał dynamiczną modyfikację wybranych parametrów kontrolera PCIe, a drugi reprezentował przestrzeń adresową Avalon-MM [49] zdefiniowaną wewnątrz FPGA. Generowane przez PC żądania zapisu i odczytu udostępnianych regionów były konwertowane przez wybrane IP na operacje zapisu i odczytu odpowiednich fragmentów wewnętrznych przestrzeni adresowych FPGA, dzięki czemu możliwa była, między innymi interakcja z rejestrami kontrolnymi i statusowymi komponentów sterujących testowanym układem scalonym.

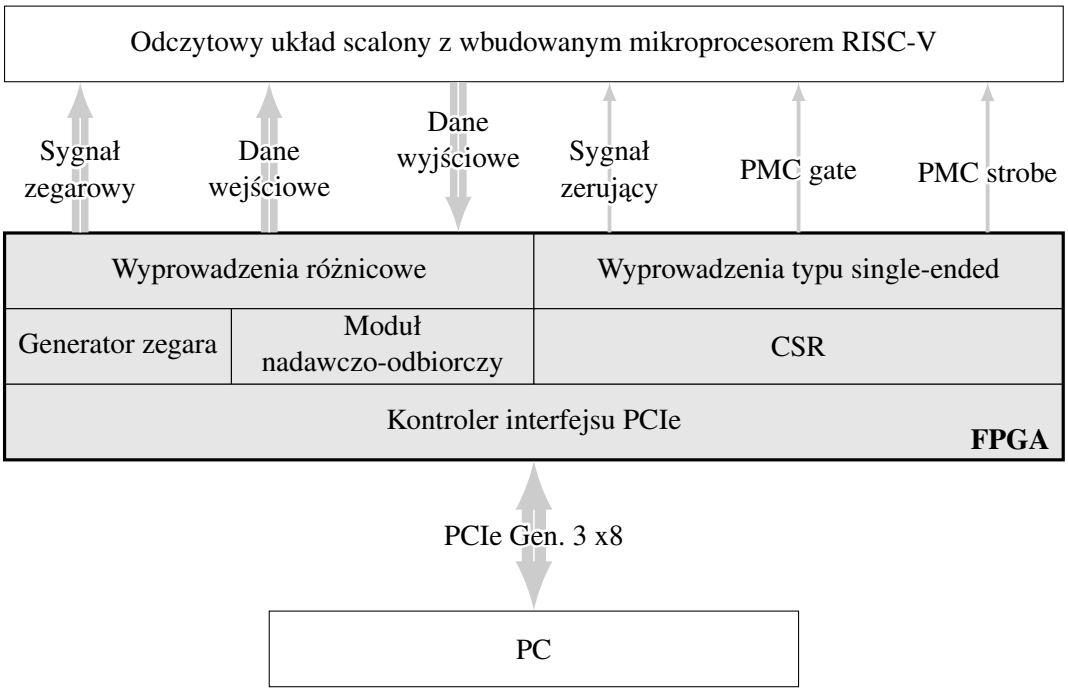
#### Interfejs testowanego układu scalonego

Skonstruowane środowisko testowe zostało połączone z zaprojektowanym scalonym układem odczytowym za pośrednictwem linii różnicowych, LVDS i asymetrycznych (single-ended). Ze względu na restrykcje dotyczące liczby wyprowadzeń układu scalonego, połączenia różnicowe wykorzystane zostały jedynie

do transmisji sygnałów szybkozmiennych, mających krytyczne znaczenie dla działania testowanego układu, takich jak zegar, dane wejściowe i dane wyjściowe. Wolnozmienne sygnały kontrolne, takie jak sygnał zerujący i sygnały sterujące PMC, zostały podłączone za pośrednictwem linii asymetrycznych. Na Rys. 4.2 przedstawiony został schemat blokowy prezentujący połączenie skonstruowanego środowiska testowego z zaprojektowanym układem scalonym.

Testowanie specjalizowanych układów scalonych jest czasochłonnym procesem, podczas którego wystąpić mogą problemy, wynikające, między innymi z braku rzetelnego połączenia pomiędzy środowiskiem testowym a testowanym układem. W celu ułatwienia diagnostyki tego typu błędów, w FPGA zaimplementowany został model układu, pozwalający na weryfikację poprawności środowiska testowego. Ponadto, realizacja tego komponentu umożliwiła przetestowanie środowiska testowego przed podłączeniem do niego odczytowego układu scalonego z wbudowanym mikroprocesorem RISC-V.

Do taktowania testowanego układu scalonego i modelu zaimplementowanego w FPGA wykorzystany został dedykowany sygnał zegarowy o częstotliwości 50 MHz. Zegar ten został wygenerowany przy użyciu dedykowanego IP udostępnianego w środowisku projektowym *Intel® Quartus® Prime*, pod nazwą *IOPLL Intel® FPGA IP* [50].

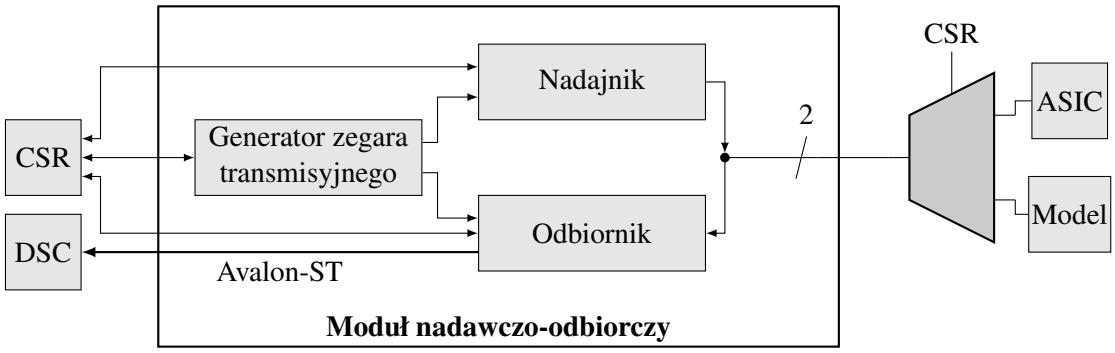



Rysunek 4.2: Schemat blokowy połączenia środowiska testowego i testowanego układu.

### Moduł nadawczo-odbiorczy

Komunikacja pomiędzy FPGA a testowanym układem scalonym została zrealizowana za pośrednictwem interfejsu UART. Mimo faktu iż, środowisko projektowe *Intel® Quartus® Prime* udostępniało IP implementujące ten interfejs, Autor niniejszej rozprawy postanowił samodzielnie zaprojektować moduł nadawczo-odbiorczy, Transceiver (TX), wzorowany na kontrolerze UART, zaimplementowanym w testowanym układzie scalonym. Powodem, dla którego podjęta została decyzja o implementacji niestandardowego kontrolera

tego interfejsu był fakt, iż działanie dostępnego IP było kontrolowane i monitorowane za pośrednictwem magistrali Avalon Memory Mapped Interface (Avalon-MM). W związku z tym, wykorzystanie tego komponentu wiązałoby się z koniecznością sterowania nim przy użyciu CPU wchodzącego w skład używanego PC. To z kolei niosłoby ze sobą ryzyko utraty danych, wynikające z braku determinizmu czasowego systemu operacyjnego nadzorującego środowisko testowe. W celu eliminacji ryzyka utraty odbieranych danych, Autor zaprojektował niestandardowy kontroler UART, który po odebraniu danych przesyłał je za pośrednictwem interfejsu Avalon Streaming Interface (Avalon-ST) do komponentu odpowiedzialnego za ich archiwizację (DSC). Opracowany TX składał się z trzech komponentów: nadajnika, odbiornika i generatora zegara transmisyjnego, a jego ogólna architektura została przedstawiona na Rys. 4.3.




Rysunek 4.3: Ogólna architektura modułu nadawczo-odbiorczego.

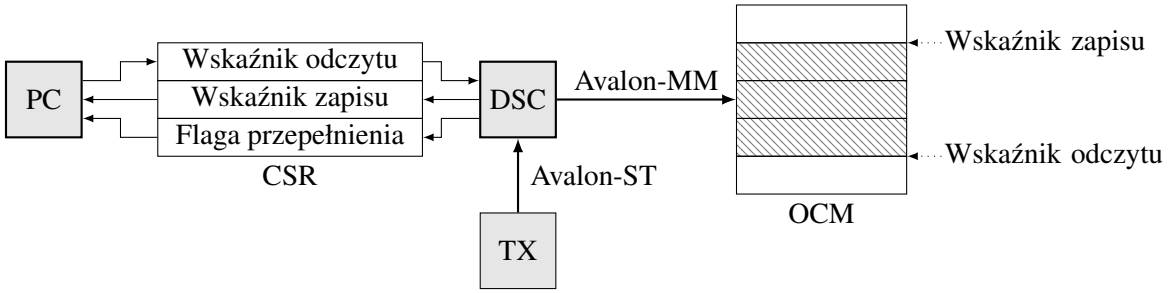
#### Moduł nadzorujący przechowywanie danych

W skonstruowanym środowisku testowym dane odbierane przez TX były przekazywane za pośrednictwem interfejsu Avalon-ST do modułu nadzorującego przechowywanie danych, Data Storage Controller (DSC). Obsługa odbiornika UART została zrealizowana w FPGA ze względu na brak determinizmu czasowego oprogramowania wykonywanego przez PC. W celu eliminacji ryzyka utraty danych przychodzących, wszystkie odebrane zapisywane były przez DSC w dedykowanej wbudowanej pamięci, On-Chip Memory (OCM), pełniącej rolę bufora cyklicznego.

Każdorazowe odebranie przez DSC danej powodowało generację żądania zapisu na magistrali Avalon-MM, do której podłączony był wspomniany OCM i inkrementację wskaźnika zapisu udostępnianego oprogramowaniu przez dedykowany rejestr CSR. W module tym zaimplementowany został mechanizm detekcji przepełnienia bufora, który ustawiał odpowiednią flagę w jednym z rejestrów statusowych CSR, gdy wskaźnik zapisu (ustawiany przez DSC) i odczytu (aktualizowany przez oprogramowanie) były równe, a moduł nadzorujący przechowywanie danych odebrał kolejną daną wymagającą zapisania. Ogólna architektura modułu nadzorującego przechowywanie danych przedstawiona została na Rys. 4.4.

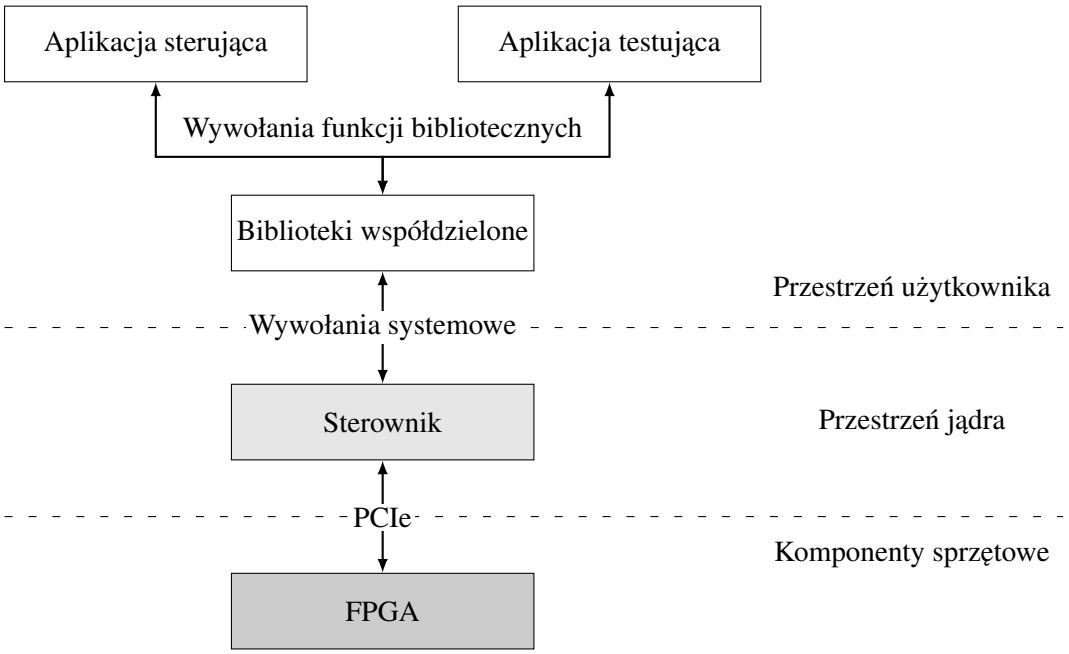
## 4.2 Architektura programowa

Oprogramowanie sterujące skonstruowanym systemem do testowania scalonych układów odczytowych było wykonywane na PC pracującym pod kontrolą Linuksa. Ze względu na szeroki zakres działań realizowanych




Rysunek 4.4: Kontrola przepływu danych w opracowanym systemie testowym.

przez opracowane oprogramowanie, zostało ono rozdzielone pomiędzy przestrzenią jądra i przestrzenią użytkownika. Oprogramowanie działające w przestrzeni jądra było bezpośrednio odpowiedzialne za sterowanie komponentami sprzętowymi skonstruowanego środowiska, a kod wykonywany w przestrzeni użytkownika wchodził w skład aplikacji umożliwiających testowanie odczytowego układu scalonego. Ogólna architektura opracowanego oprogramowania została przedstawiona na Rys. 4.5.




Rysunek 4.5: Ogólna architektura programowa.

### 4.2.1 Oprogramowanie działające w przestrzeni jądra

#### Inicjalizacja

Opracowany sterownik został zrealizowany jako moduł jądra. Podczas ładowania go do systemu operacyjnego, lista wspieranych przez niego urządzeń była porównywana z listą urządzeń podłączonych za pośrednictwem interfejsu PCIe skonstruowaną podczas inicjalizacji systemu. Gdy identyfikatory urządzeń zostały do siebie dopasowane, wykonywana była następująca sekwencja inicjalizacyjna:

- alokacja pamięci zarządzanej przez urządzenie; po załadowaniu modułu do jądra, sterownik alokował pamięć niezbędną do przechowywania prywatnych danych. Wykorzystanie mechanizmu zarządzania zasobami przez urządzenie [51], pozwoliło na przeniesienie odpowiedzialności za zwalnianie zaalokowanego obszaru z kodu zaimplementowanego w module na dedykowany mechanizm jądra.

- inicjalizacja urządzenia PCIe; zasoby urządzenia były pobierane i inicjalizowane przy użyciu dedykowanego Application Programming Interface (API) interfejsu PCIe [52]. Podczas tego etapu urządzenie było włączane, a BAR reprezentujący przestrzeń adresową Avalon-MM zdefiniowaną wewnątrz FPGA był mapowany, w celu udostępnienia jego bazowego adresu aplikacjom sterującym skonstruowanym środowiskiem.

- rejestracja urządzenia w systemie operacyjnym; w celu umożliwienia komunikacji pomiędzy procesami przestrzeni użytkownika a opracowanym modułem, niezbędne było zarejestrowanie w systemie operacyjnym urządzania wspieranego przez sterownik. Rejestracja została zrealizowana przy użyciu mechanizmu udostępnianego przez jądro pod nazwą „miscellaneous device" [53].

- konfiguracja logiki zaimplementowanej w FPGA; ostatnią operacją wykonywaną przez jądro podczas ładowania modułu była inicjalizacja logiki zaimplementowanej w FPGA, polegająca na ustawieniu cyfrowych wyprowadzeń ogólnego przeznaczenia sterujących testowanym układem, wyzerowaniu wewnętrznych komponentów systemu i generacji sygnału zerującego testowany układ scalony.

#### Interfejs przestrzeni użytkownika

Komunikacja pomiędzy opracowanym modułem jądra a procesami przestrzeni użytkownika została zrealizowana przy użyciu wywołań systemowych [54]. W sterowniku zaimplementowane zostało wsparcie dla następujących operacji:

- `read`, wykorzystywana do kopiowania danych z bufora cyklicznego nadzorowanego przez DSC, do bufora zaalokowanego przez proces przestrzeni użytkownika (podczas obsługi tego wywołania, sterownik dokonywał aktualizacji wskaźnika odczytu przechowywanego w CSR),

- `write`, używana do przesyłania danych poprzez UART do testowanego układu scalonego,

- `mmap`, wykorzystywana do translacji przestrzeni adresowej zdefiniowanej wewnątrz FPGA na wirtualną przestrzeń adresową procesu przestrzeni użytkownika.

### 4.2.2 Oprogramowanie działające w przestrzeni użytkownika

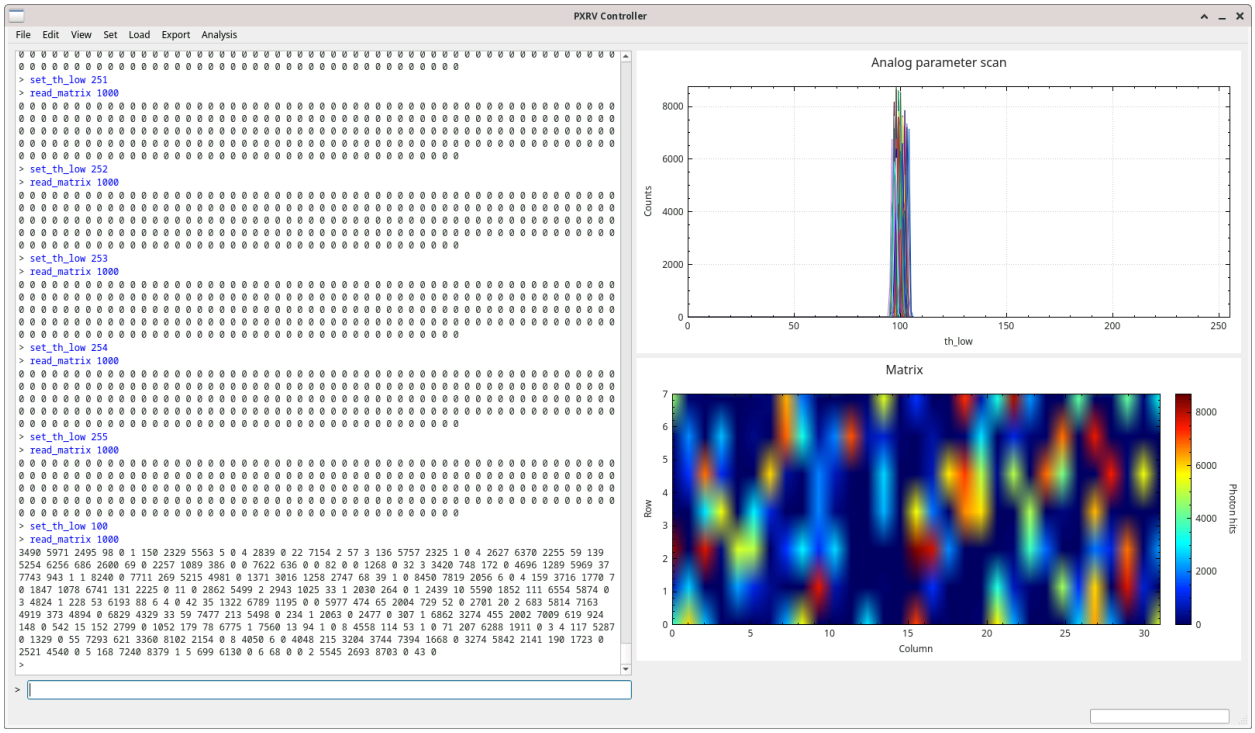
#### Współdzielone biblioteki

Niskopoziomowe funkcje współdzielone pomiędzy aplikacją sterującą i aplikacją testującą, zostały zdefiniowane w bibliotekach, dynamicznie łączonych z aplikacjami podczas ich uruchamiania. Zastosowanie takiego podejścia pozwoliło na wykorzystywanie tego samego kodu zarówno podczas interaktywnego sterowania środowiskiem testowym, jak i podczas uruchamiania zautomatyzowanych procedur testowych.

### Interaktywny kontroler środowiska testowego

Interaktywny kontroler środowiska testowego był opracowaną w języku C++ aplikacją graficzną opartą na bibliotece Qt [55]. Jej graficzny interfejs użytkownika, Graphical User Interface (GUI), był wzorowany na aplikacjach sterujących kontrolerami portów szeregowych, a dominująca część jej głównego okna przeznaczona została na wyświetlanie danych tekstowych wysyłanych przez testowany układ odczytowy. Główne okno interaktywnego kontrolera środowiska testowego przedstawione zostało na Rys. 4.6.

Wraz z tekstową prezentacją wszystkich danych odbieranych od testowanego układu, zaimplementowana została graficzna wizualizacja danych będących wynikami wybranych komend, takich jak rejestracja promieniowania. Mechanizm ten został wdrożony w celu ułatwienia interpretacji wyników odczytu PM i polegającego na pomiarze wyników rejestracji promieniowania dla kolejnych wartości globalnej konfiguracji analogowej PM, skanu napięć progowych. Wizualizacja danych została zrealizowana przy wykorzystaniu biblioteki QCustomPlot [56].



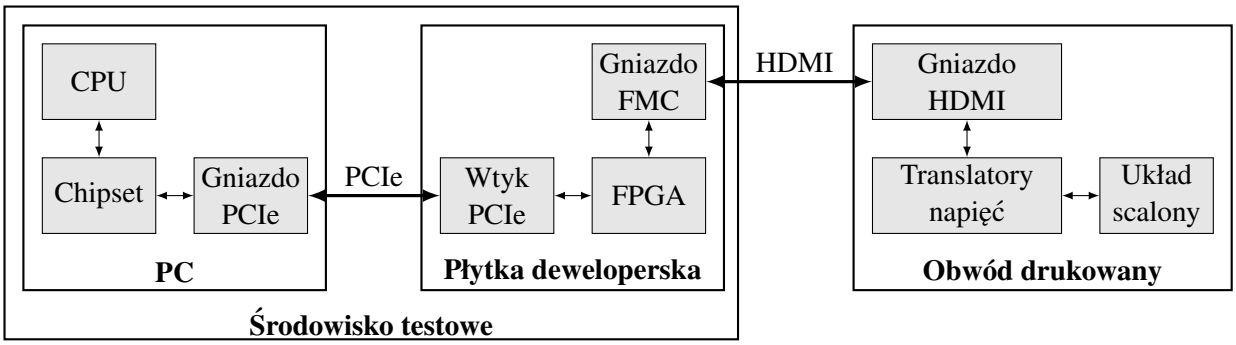

Rysunek 4.6: Główne okno interaktywnego kontrolera środowiska testowego.

### Aplikacja testująca

Zautomatyzowana aplikacja testująca była opartym na bibliotece GoogleTest [57] programem, umożliwiającym automatyczne testowanie scalonego układu odczytowego, w oparciu o uprzednio zdefiniowane procedury testowe. Aplikacja ta ułatwiła weryfikację kolejnych egzemplarzy wyprodukowanego układu scalonego i skróciła czas niezbędny do przeprowadzenia wymaganych testów.

## 4.3 Testy odczytowego układu scalonego z wbudowanym mikroprocesorem RISC-V

Zaprojektowane przez Autora niniejszej rozprawy środowisko testowe zostało wykorzystane do przetestowania układu scalonego przedstawionego w Rozdziale 3. Ogólna architektura skonstruowanego stanowiska testowego przedstawiona została na Rys. 4.7



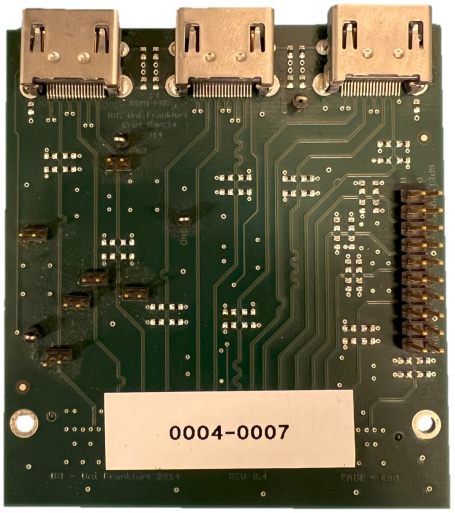

Rysunek 4.7: Ogólny schemat stanowiska testowego zbudowanego przy użyciu opracowanego środowiska.

Wykorzystana do budowy środowiska testowego płytka deweloperska Intel Arria 10 GX została wyposażona, między innymi w krawędziowy konektor PCIe oraz dwa gniazda FPGA Mezzanine Card (FMC) charakteryzujace się dużą liczbą wyprowadzeń. Pierwsze z wymienionych złącz wykorzystane zostało do połączenia układu FPGA z PC, którego parametry przedstawione zostały w Sekcji 4.1, natomiast jeden ze wspomnianych konektorów FMC użyty został do skomunikowania środowiska testowanym układem scalonym.

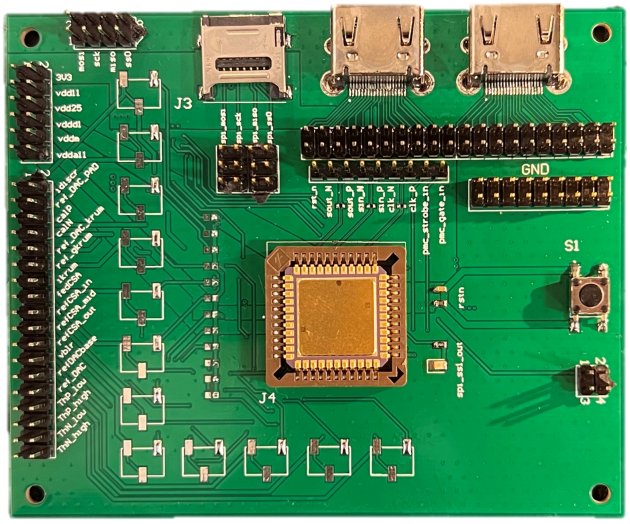
Połączenie pomiędzy płytą deweloperską Intel Arria 10 GX a układem scalonym zaprojektowanym przez Autora niniejszej rozprawy zrealizowane zostało za pośrednictwem kabla High Definition Multimedia Interface (HDMI). Mimo iż wykorzystywana płytka została wyposażona w gniazdo HDMI, nie mogło ono zostać użyte do komunikacji z testowanym chipem ze względu na fakt, że zostało ono podłączone do FPGA poprzez pomocniczy układ scalony pełniący rolę kodeka. Wykorzystywany kabel HDMI został podłączony do FPGA za pośrednictwem przedstawionej na Rys 4.8a przejściówki FMC-HDMI.

Podczas składania zamówienia na produkcję układów ASIC zawierających odczytowy układ scalony z wbudowanym mikroprocesorem RISC-V, zamówiona została również usługa pakowania produkowanych układów w obudowach JLCC44 zgodnie z diagramem montażowym przedstawionym na Rys. A.3, który ze względu na duży rozmiar umieszczony został w Dodatku A. Tak zapakowane układy scalone montowane były na dedykowanym obwodzie drukowanym [58], zawierającym, między innymi złączę HDMI, potencjometry do regulacji prądów polaryzujących PM i złącza typu goldpin umożliwiające pomiar napięć występujących wewnątrz układu scalonego. Obwód drukowany wykorzystywany podczas testów zaprojektowanego układu scalonego przedstawiony został na Rys. 4.8b

Stanowisko laboratoryjne wykorzystywane do testów układu scalonego zaprojektowanego przez Autora niniejszej rozprawy zostało skonstruowane w Laboratorium Mikrostruktur oraz Montażu Mikrosystemów zlokalizowanym na terenie Akademii Górniczo-Hutniczej. W skład stanowiska testowego poza przedstawionym środowiskiem testowym i testowanym układem scalonym wchodziły dwa zasilacze laboratoryjne



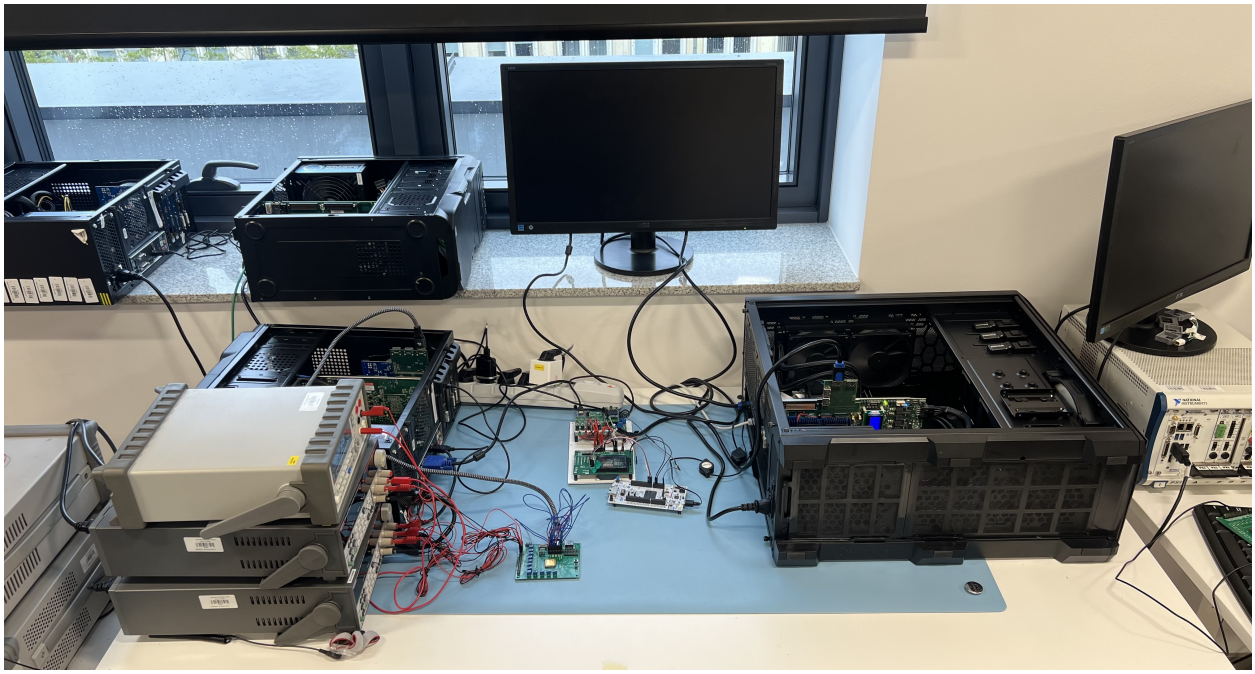

(a) Przejściówka FMC-HDMI.




(b) Obwód testowy dla układu scalonego.

Rysunek 4.8: Obwody drukowane wykorzystywane podczas testów zaprojektowanego układu scalonego.

Keithley 2231A-30-3 [59], które wykorzystane zostały do zasilenia testowanego układu scalonego. Skonstruowane stanowisko laboratoryjne przedstawione zostało na Rys. 4.9.



Rysunek 4.9: Stanowisko laboratoryjne do testów zaprojektowanego układu scalonego.

### 4.3.1 Pierwsze uruchomienie układu scalonego

Pierwsze uruchomienie układu scalonego zaprojektowanego przez Autora niniejszej rozprawy, wymagało podłączenia do urządzenia trzech źródeł napięcia, zasilających odpowiednio rdzeń, bufory wejścia wyjścia, Input/Output (I/O), oraz pamięci RAM. Doprowadzenie do obwodu drukowanego, a w związku z tym rów-

nież do układu scalonego, odpowiednich napięć, poskutkowało wykonaniem przez mikroprocesor pierwszej instrukcji programu inicjalizującego, znajdującego się w dedykowanej pamięci ROM.

### Weryfikacja wykonywania kodu przez mikroprocesor

Podczas projektowania aplikacji inicjalizującej, pod uwagę wzięte zostały różne scenariusze, które mogłyby wystąpić podczas próby uruchomienia układu scalonego. Uwzględnione zostały zarówno problemy związane z błędami zawartymi w strukturze krzemowej, jak i komplikacje wynikające z niepoprawnie wykonanych połączeń na obwodzie drukowanym i niewłaściwie przylutowanych komponentów. W aplikacji tej zaimplementowane zostały mechanizmy pozwalające na określenie potencjalnych problemów i wskazanie ich przyczyn.

Pierwszym zaimplementowanym mechanizmem diagnostycznym było wyświetlanie uprzednio zdefiniowanych wzorców świetlnych przy użyciu diody Light-Emitting Diode (LED) podłączonej do jednego z wyprowadzeń testowanego układu scalonego. Sterowanie stanem tego komponentu wymagało od procesora wykonywania zapisów do odpowiedniego rejestru kontrolnego w określonych odstępach czasowych, w związku z czym, zmiana stanu tej diody potwierdzała wykonywanie przez CPU kolejnych poprawnych instrukcji. Mechanizm ten pozwolił nie tylko na potwierdzenie działania mikroprocesora, ale również na śledzenie aktualnie wykonywanych gałęzi kodu aplikacji inicjalizującej i sygnalizowanie potencjalnych błędów.

### Sprawdzenie transmisji informacji diagnostycznych

Dioda LED dobrze sprawdzała się w sytuacjach, w których opracowany system mikroprocesorowy musiał sygnalizować wystąpienie jakiegoś zdarzenia, np. przekroczenie czasu oczekiwania na przesyłany poprzez UART kod aplikacji użytkownika. Każdy z wyświetlanych za pomocą tego komponentu wzorców świetlnych mógł przenosić inną informację, jednakże ich rozróżnianie stawało się tym trudniejsze, im więcej wzorców było zaimplementowanych. Niezbędnym stało się więc opracowanie nowego mechanizmu, który pozwoliłby na przekazywanie informacji diagnostycznych w sposób łatwiejszy do analizy.

Zaimplementowanym rozwiązaniem była transmisja tekstowych informacji diagnostycznych za pośrednictwem interfejsu UART. Mechanizm ten umożliwił systemowi mikroprocesorowemu nie tylko informowanie użytkownika o wystąpieniu konkretnych zdarzeń, takich jak uprzednio wspomniane przekroczenie czasu oczekiwania na kod aplikacji użytkownika, ale również na przesyłanie dodatkowych informacji diagnostycznych, takich jak, np. liczba odebranych bajtów. Na List. 4.1 przedstawione zostały przykładowe informacje diagnostyczne przesłane przez opracowany system mikroprocesorowy po upłynięciu czasu oczekiwania na kod aplikacji użytkownika (10 sekund).

Listing 4.1: Przykładowe informacje diagnostyczne przesłane przez aplikację inicjalizującą.

```
bootloader started
INFO: codeload source: uart
ERROR: received bytes: 0
ERROR: codeload from uart failed
```

#### Test ładowania aplikacji

Ostatnim eksperymentem zrealizowanym podczas pierwszego uruchomienia opracowanego układu scalonego była weryfikacja ładowania aplikacji użytkownika. Test ten zrealizowany został przy użyciu skonstruowanego środowiska testowego i dedykowanego programu, którego zadaniem było odsyłanie poprzez UART ciągu znakowego „pong", po uprzednim otrzymaniu przez ten interfejs ciągu znakowego „ping". Na List. 4.2 przedstawione zostały informacje diagnostyczne przesłane przez opracowany system mikroprocesorowy w czasie trwania eksperymentu. Kolorem niebieskim wyróżnione zostały dane przesłane do układu scalonego.

Listing 4.2: Informacje diagnostyczne zebrane podczas testu ładowania aplikacji.

```
bootloader started
INFO: codeload source: uart
INFO: codeload finished
INFO: bootloader finished
ping_pong started
> ping
pong
```

#### Wnioski

Mimo złożoności całego systemu skonstruowanego przez Autora niniejszej rozprawy (oprogramowanie działające na PC, układ cyfrowy zaimplementowany w FPGA oraz zaprojektowany system mikroprocesorowy dla hybrydowych detektorów promieniowania jonizującego), pierwsze uruchomienie układu scalonego przebiegło sprawnie i w jego trakcie nie wystąpiły żadne komplikacje. Sytuacja ta była rezultatem wielu testów przeprowadzonych przy wykorzystaniu prototypu zaimplementowanego w FPGA. Prototyp ten pozwolił na opracowanie całej architektury sprzętowej i oprogramowania sterującego systemem, jeszcze przed wyprodukowaniem układu scalonego. Wszystkie przeprowadzone eksperymenty przebiegły pomyślnie.

### 4.3.2 Pomiar natężeń prądów

Po pierwszym uruchomieniu układu scalonego wykonany został kolejny eksperyment, podczas którego zmierzony został pobór prądu z poszczególnych źródeł napięciowych. W celu wykonania odpowiednich pomiarów, do układu scalonego podłączony został sygnał zegarowy o częstotliwości 50 MHz oraz wgrany został do pamięci kodu program, którego zadaniem było zmienianie stanu diody statusowej z częstotliwością wynoszącą 1 Hz. W Tab. 4.1 przedstawione zostały wyniki pomiarów zrealizowanych przy użyciu multimetru Keithley 2110 [60].

### 4.3.3 Testy układów peryferyjnych

Testowany układ scalony był systemem mikroprocesorowym zawierającym zarówno komponenty dedykowane HPD, jak i również standardowe układy peryferyjne, znane, między innymi z ogólnodostępnych

Tablica 4.1: Wyniki pomiaru prądów pobieranych przez testowany układ scalony.

| Domena | Zasilane komponenty | Napięcie [V] | Miganie diodą [mA] | Zapis i odczyt PM [mA] |
|---|---|---|---|---|
| *vdd11* | Mikroprocesor i układy peryferyjne | 1,10 | 2,92 | 3,29 |
| *vdd25* | Bufory I/O | 2,50 | 16,59 | 16,11 |
| *vddd11* | Komponenty cyfrowe PM | 1,10 | 1,58 | 23,61 |
| *vdda11* | Komponenty analogowe PM i pamięci RAM | 1,10 | 0,0001 | 0,0002 |
| *vddm* | PM | 0,81 | 0,08 | 0,69 |

mikrokontrolerów. Ze względu na fakt, iż wszystkie te komponenty zostały zaprojektowane przez Autora niniejszej rozprawy, niezbędna była weryfikacja ich poprawności. W związku z tym, w aplikacji testującej, wchodzącej w skład opracowanego środowiska testowego, zaimplementowane zostały testy, weryfikujące wybrane funkcjonalności zaprojektowanych układów peryferyjnych. W Tab. 4.2 przedstawione zostały wyniki przeprowadzonych testów.

Tablica 4.2: Wyniki testów wybranych funkcjonalności opracowanych układów peryferyjnych.

| Układ peryferyjny | Funkcjonalność | Model | ASIC |
|---|---|---|---|
| GPIO | Sterowanie wyprowadzeniami wyjściowymi | + | + |
| | Odczyt wyprowadzeń wejściowych | + | + |
| | Przerwanie wyzwolone wystąpieniem zbocza narastającego | + | + |
| | Przerwanie wyzwolone wystąpieniem zbocza opadającego | + | + |
| SPI | Transmisja i odbiór danych (odbiornik podłączony do nadajnika) | + | + |
| | Przerwanie wyzwolone zakończeniem transmisji | + | + |
| Timer | Przerwanie wyzwolone upłynięciem określonej liczby cykli zegara | + | + |
| UART | Odbiór danych | + | + |
| | Transmisja danych | + | + |
| | Przerwanie wyzwolone zakończeniem transmisji | + | + |
| | Przerwanie wyzwolone zakończeniem odbioru | + | + |

### GPIO

Opracowana aplikacja testująca weryfikowała układy peryferyjne w kolejności alfabetycznej, biorąc pod uwagę zarówno model zaimplementowany w FPGA, jak i wyprodukowany układ scalony. Jako pierwszy był więc uruchamiany zbiór testów weryfikujących kontroler GPIO. Sprawdzane były wówczas nie tylko podstawowe funkcjonalności tego komponentu, takie jak sterowanie cyfrowymi wyprowadzeniami ogólnego przeznaczenia i odczyt stanów tych wyprowadzeń, ale również testowane było generowanie przerwań wyzwalanych wykryciem zmian sygnału doprowadzonego do monitorowanego wyprowadzenia wejściowego.

Wszystkie testowane funkcjonalności kontrolera GPIO działały poprawnie, zarówno w przypadku modelu zaimplementowanego w FPGA jak i układu ASIC.

#### SPI

Po sprawdzeniu kontrolera GPIO, aplikacja testująca przechodziła do weryfikacji kontrolera SPI. Test tego układu peryferyjnego realizowany był w oparciu o sprzężenie zwrotne, w którym wyjście danych Master Output, Slave Input (MOSI) podłączone było do wejścia danych Master Input, Slave Output (MISO). Przeprowadzone eksperymenty potwierdziły poprawność tego komponentu, zarówno w przypadku modelu zaimplementowanego w FPGA, jak i układu ASIC. Poprawność transmisji kontrolera zaimplementowanego w układzie ASIC została dodatkowo zweryfikowana poprzez analizę sygnałów wybierającego sygnał podrzędny Slave Select (SS), zegarowego Serial Clock (SCK) oraz danych wyjściowych MOSI wykonaną przy użyciu oscyloskopu.

#### Układ czasowo-licznikowy

Kolejnym testowanym układem peryferyjnym był układ czasowo-licznikowy. Komponent ten weryfikowany był przy pomocy programu, który konfigurował cykliczne przerwania i w funkcji obsługi tych przerwań wysyłał do środowiska testowego ciąg znakowy „ping". Warunkiem uznania tego testu za zakończony pomyślnie było odebranie przez środowisko testowe odpowiedniej liczby wspomniany ciągów znakowych. Układ czasowo-licznikowy działał poprawnie, zarówno w modelu zaimplementowanym w FPGA, jak i w układzie ASIC.

#### UART

Ostatnim standardowym układem peryferyjnym sprawdzanym przez aplikację testującą był kontroler UART. W czasie testów weryfikowane były zarówno podstawowe funkcjonalnosci tego komponentu, takie jak transmisja i odbiór danych, jak i również generacja przerwań wyzwalanych zakończeniem transmisji i odbiorem danych. Wszystkie przeprowadzane testy zakończyły się sukcesem, zarówno w przypadku modelu zaimplementowanego w FPGA, jak i układu ASIC.

#### Wnioski

Aplikacja testująca wchodząca w skład skonstruowanego środowiska testowego pozwoliła na zdefiniowanie testów, weryfikujących poprawność komponentów zaimplementowanych wewnątrz opracowanego układu scalonego. Przeprowadzone przy jej użyciu eksperymenty potwierdziły poprawność układów peryferyjnych zaprojektowanych przez Autora niniejszej rozprawy.

### 4.3.4 Weryfikacja kontrolera matrycy pikseli

#### Testy rejestrów kontrolno-statusowych

Weryfikacja PMC została rozpoczęta od testów CSR zaimplementowanych wewnątrz kontrolera matrycy. Ze względu na fakt, iż jednym z zastosowań tych rejestrów było sterowanie PM w bezpośrednim trybie

pracy PMC, podjęta została decyzja o weryfikacji ich poprawności za pośrednictwem testu zapisu i odczytu matrycy. Przeprowadzony eksperyment zakończył się pomyślnie, zarówno w przypadku modelu zaimplementowanego w FPGA, jak i układu ASIC, potwierdzając tym samym poprawność implementacji CSR oraz interfejsu łączącego PM z jej kontrolerem.

### Testy koprocesora

Kolejnym eksperymentem weryfikującym PMC było sprawdzenie poprawności koprocesora sterującego matrycą. Działanie PMCC zostało zweryfikowane poprzez wykonanie testu zapisu i odczytu PM, w odpowiednim trybie pracy jej kontrolera. Z racji tego, iż opracowany koprocesor wspierał dwie instrukcje umożliwiające nieliniowe wykonywanie kodu (`loop`, powodującą wykonanie bezwarunkowego skoku oraz `storeb`, pozwalającą na wykonanie skoku warunkowego), przeprowadzone zostały dwa eksperymenty, różniące się instrukcją wykorzystywaną do wykonywania skoków. Przeprowadzane testy zakończyły się sukcesem, zarówno w przypadku modelu zaimplementowanego w FPGA, jak i układu ASIC.

### Testy akceleratorów

Następnym eksperymentem sprawdzającym poprawność PMC była weryfikacja opracowanych akceleratorów. Działanie tych komponentów zostało zweryfikowane za pośrednictwem testu zapisu i odczytu PM, zrealizowanego w odpowiednim trybie pracy jej kontrolera. Pierwsze wykonanie eksperymentu zakończyło się pomyślnie dla modelu zaimplementowanego w FPGA, jednakże z układu ASIC odczytane zostały dane różniące się od uprzednio wpisanych.

W celu określenia, który z opracowanych akceleratorów działał niepoprawnie, przeprowadzone zostały dodatkowe testy zapisu i odczytu matrycy, podczas których wykorzystane zostały różne konfiguracje trybów pracy PMC. Wykonany eksperyment wykazał, że z PM odczytywane były dane różne od uprzednio wpisanych wtedy, gdy niezależnie od trybu pracy PMC wykorzystywanego podczas wpisywania danych, do odczytu wykorzystywany był tryb wykorzystujący koprocesor i akceleratory. Wyniki przeprowadzonego eksperymentu przedstawione zostały w Tab. 4.3.

Tablica 4.3: Wyniki testów zapisu i odczytmu matrycy dla wybranych konfiguracji trybów pracy PMC.

| Tryb pracy używany podczas zapisu | Tryb pracy używany podczas odczytu | Model | Układ scalony |
|---|---|---|---|
| Bezpośredni | Bezpośredni | + | + |
| Wykorzystujący koprocesor | Wykorzystujący koprocesor | + | + |
| Wykorzystujący koprocesor i akceleratory | Wykorzystujący koprocesor i akceleratory | + | - |
| Bezpośredni | Wykorzystujący koprocesor i akceleratory | + | - |
| Wykorzystujący koprocesor i akceleratory | Bezpośredni | + | + |

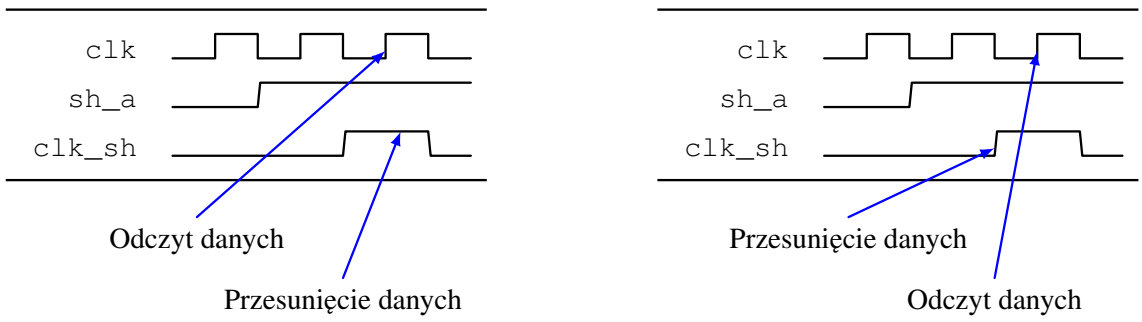
Analiza dostrzeżonego problemu została rozpoczęta od dokładniejszego porównania danych wpisywanych i odczytywanych z PM, podczas testu wykonywanego przy użyciu koprocesora i akceleratorów.

Po sprawdzeniu binarnych reprezentacji danych zebranych w czasie eksperymentu, dostrzeżona została zależność pomiędzy danymi wpisywanymi i odczytywanymi z matrycy - dane odczytywane z matrycy były przesunięte o jeden bit w lewo względem danych do niej wpisywanych. W celu określenia przyczyny niniejszego problemu, podjęta została decyzja o ponownej analizie kodu Hardware Description Language (HDL) modułu definiującego akcelerator danych wyjściowych. Na List. 4.3 przedstawiony został fragment kodu HDL tego modułu odpowiedzialny za odczyt danych z PM.

Listing 4.3: Fragment kodu HDL akceleratora danych wejściowych odpowiedzialny za odczyt danych z PM.

```
92  always_ff @(posedge clk or negedge rst_n) begin
93      if (!rst_n)
94          rdata <= {16{32'b0}};
95      else
96          rdata <= rdata_nxt;
97  end
98
99  always_comb begin
100     rdata_nxt = rdata;
101
102     case (state)
        ⋮
110     ACTIVE: begin
111         if (clk_sh) begin
112             for (int i = 0; i < 32; ++i)
113                 rdata_nxt[i][15-bits_counter] = pm_data_dout[i];
114         end
115     end
116     endcase
117 end
```

Zgodnie z definicją przedstawioną na List. 4.3, akcelerator danych wejściowych odczytywał dane wychodzące z matrycy (`pm_data_out`), gdy wysoki stan logiczny występował na linii sygnałowej `clk_sh` interfejsu kontrolnego PM. Ze względu na fakt, iż odczytywane dane były zapisywane w rejestrze `rdata` po wystąpieniu narastającego zbocza sygnału zegarowego `clk`, wartością odczytywaną przez akcelerator była wartość znajdująca się na interfejsie danych wyjściowych PM w chwili wystąpienia narastającego zbocza sygnału `clk`. Z racji tego, że warunkiem aktualizacji interfejsu danych wyjściowych matrycy było wystąpienie narastającego zbocza sygnału `clk_sh`, pierwszy odczyt danych poprzedzany był aktualizacją interfejsu wyjściowego PM, a co za tym idzie nadpisaniem nieodczytanych danych. Przedstawione zjawisko nie występowało podczas odczytu modelu zaimplementowanego w FPGA, ponieważ jego działanie było dodatkowo synchronizowane sygnałem zegarowym `clk`, co skutkowało zaktualizowaniem interfejsu danych wyjściowych matrycy dopiero po wystąpieniu narastającego zbocza tego sygnału zegarowego. Diagramy przebiegów cyfrowych przedstawiające pierwsze odczyty danych z modelu zaimplementowanego w FPGA i układu ASIC przedstawione zostały odpowiednio na Rys. 4.10a oraz Rys. 4.10b.




(a) Model zaimplementowany w FPGA.

(b) Układ ASIC.

Rysunek 4.10: Wykresy przebiegów cyfrowych prezentujące pierwsze odczyty danych z PM.

Przyczyną niepoprawnego działania PMC w trybie wykorzystującym koprocesor i akceleratory było zbyt późne wykonywanie pierwszego odczytu danych wychodzących z PM. W celu załagodzenia tego problemu, podjęta została decyzja o poprzedzeniu odczytu matrycy we wspomnianym trybie PMC, odczytem wykonanym w trybie bezpośrednim. Ten bezpośredni odczyt danych traconych podczas wykonywania opracowanej procedury odczytowej, pozwolił na programowe odtworzenie zawartości PM, a w efekcie na „naprawienie" nierzetelnego algorytmu odczytowego. W Tab. 4.4 przedstawione zostało zestawienie czasów potrzebnych na zapis i odczyt PM w poszczególnych trybach pracy PMC, w przypadku modelu zaimplementowanego w FPGA i układu ASIC.

Tablica 4.4: Zestawienie algorytmów sterujących PM, w przypadku modelu zaimplementowanego w FPGA i układu ASIC.

| Tryb pracy PMC | Algorytm | FPGA | ASIC | Różnica [%] |
|---|---|---|---|---|
| Bezpośredni | Zapis | 91510 | 91078 | −0,5 |
| | Odczyt | 89256 | 88824 | −0,5 |
| Wykorzystujący koprocesor | Zapis | 94575 | 94599 | 0,0 |
| | Odczyt | 98695 | 98725 | 0,0 |
| Wykorzystujący koprocesor i akceleratory | Zapis | 3179 | 3179 | 0,0 |
| | Odczyt | 3226 | 8350 | +158,8 |

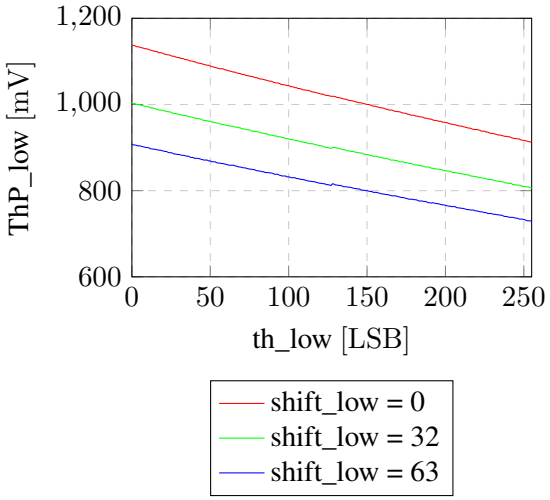
Przeprowadzone eksperymenty potwierdziły poprawność większości funkcjonalności opracowanego PMC. Dostrzeżony podczas eksperymentów problem związany z działaniem akceleratora danych wejściowych, wskazał dodatkową korzyść wynikającą z integracji mikroprocesora z PM. Mimo że, wykonywany przez CPU algorytm poprawkowy wydłużył odczyt matrycy o 158 % (z 3 226 do 8 350 cykli), odczyt PM w trybie wykorzystującym koprocesor i akceleratory był o 90 % krótszy od odczytu realizowanego w bezpośrednim trybie PMC.

### Test przetworników korygujących napięcia niezrównoważenia dyskryminatorów

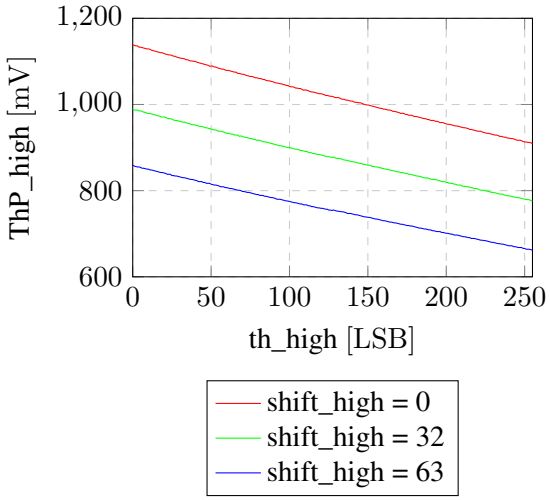
Rozrzut parametrów fizycznych pomiędzy komponentami o takich samych wymiarach geometrycznych, występujący w układach scalonych, skutkuje rozrzutem parametrów torów analogowych, wchodzących

w skład układów odczytowych dla sensorów promieniowania jonizującego. W celu minimalizacji wpływu tego zjawiska na wyniki rejestracji promieniowania, wewnątrz układów odczytowych umieszcza się komponenty dostrajające, takie jak, przetworniki cyfrowo-analogowe kompensujące parametry poszczególnych pikseli.

Jednym z pierwszych eksperymentów wykonywanych podczas testowania analogowych torów odczytowych dla sensorów promieniowania jonizującego jest weryfikacja przetworników polaryzujących tory analogowe, wspólnych dla wszystkich pikseli. W opracowanym układzie każdy piksel zawierał dwa dyskryminatory (*low* i *high*), których poziom dyskryminacji był ustalany poprzez podanie różnicowo dwóch napięć (*ThP_low* / *ThN_low* i *ThP_high* / *ThN_high*). Wspólny poziom sygnałów różnicowych jest ustawiany przez dedykowane przetworniki (*shift_low* i *shift_high*). Każde z tych napięć było programowane za pośrednictwem PMC przez odpowiednie bity globalnej konfiguracji analogowej. Wyniki pomiarów wykonanych przy użyciu przetwornika analogowo-cyfrowego wbudowanego w mikrokontroler *STM32L476RGT6* umieszczony na płytce deweloperskiej *STM32 Nucleo-L476RG [61]* zostały przedstawione na Rys. 4.11a, Rys. 4.11b, Rys. 4.12a i Rys. 4.12b.




(a) Pomiar napięcia ThP_low.



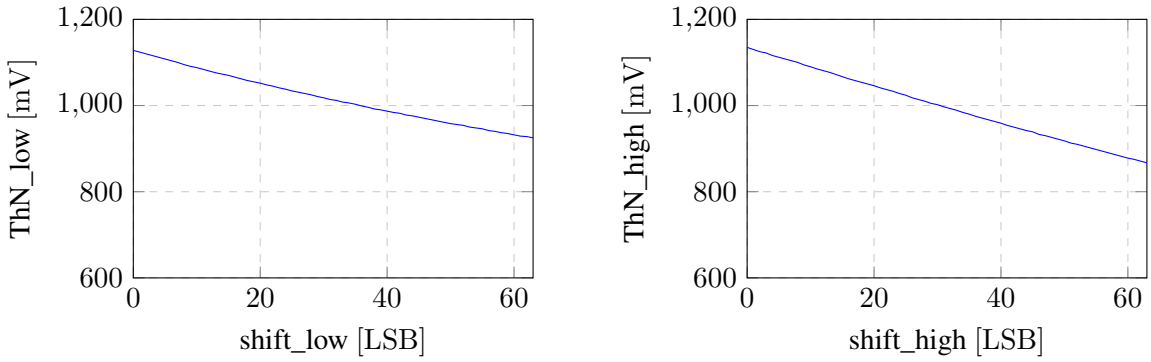

(b) Pomiar napięcia ThP_high.

Rysunek 4.11: Zmierzone charakterystyki przejściowe przetworników polaryzujących *ThP_low* i *ThP_high*.

W Tab. 4.5 przedstawione zostały zakresy pracy przetworników korygujących napięcia niezrównoważenia dyskryminatorów.

Tablica 4.5: Zmierzone zakresy pracy przetworników korygujących napięcia niezrównoważenia dyskryminatorów.

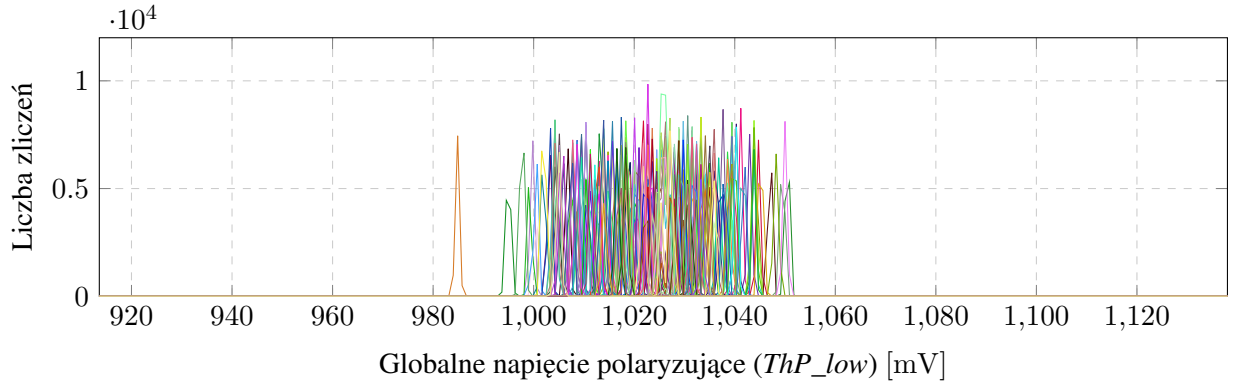
| Parametr | Napięcie | Zakres [LSB] | Zakres [mV] | LSB [mV] |
|---|---|---|---|---|
| th_low (shift_low = 0) | ThP_low | 0 - 255 | 913 - 1138 | 0,88 |
| th_high (shift_high = 0) | ThP_high | 0 - 255 | 909 - 1139 | 0,90 |
| shift_low | ThN_low | 0 - 255 | 925 - 1128 | 0,79 |
| shift_high | ThN_high | 0 - 255 | 867 - 1135 | 1,04 |




(a) Pomiar napięcia ThN_low.

(b) Pomiar napięcia ThN_high.

Rysunek 4.12: Zmierzone charakterystyki przejściowe przetworników polaryzujących *ThN_low* i *ThN_high*.




Rysunek 4.13: Wyniki pomiarów impulsów generowanych przez szumy analogowych torów odczytowych.

### Test korekcji napięć niezrównoważenia dyskryminatorów

Analogowe tory odczytowe wchodzące w skład układów odczytowych dla sensorów promieniowania jonizującego mogą być testowane przy użyciu szumów generowanych przez te tory. Podczas rejestracji promieniowania, dyskryminatory podłączane są do dedykowanych układów zliczających, których zawartość jest inkrementowana po dostarczeniu przez dyskryminatory impulsów elektrycznych o określonych parametrach. Wspomniane impulsy mogą być generowane nie tylko w wyniku wykrycia przez sensory cząstek promieniowania, ale również w wyniku występowania szumów w analogowych torach odczytowych.

Występujące w analogowych torach odczytowych napięcia niezrównoważenia dyskryminatorów powodowały, że poszczególne tory odczytowe rejestrowały maksymalne liczby zliczeń impulsów szumowych dla różnych globalnych napięć polaryzujących. Wykonanie sekwencji rejestracji i odczytów impulsów szumowych dla kolejnych wartości globalnego napięcia polaryzującego pozwoliło na określenie zakresu, w którym mieściły się napięcia zapewniające maksymalne liczby zliczeń dla poszczególnych pikseli. Na Rys. 4.13 przedstawione zostały wyniki pomiaru impulsów generowanych przez szumy analogowych torów odczytowych.

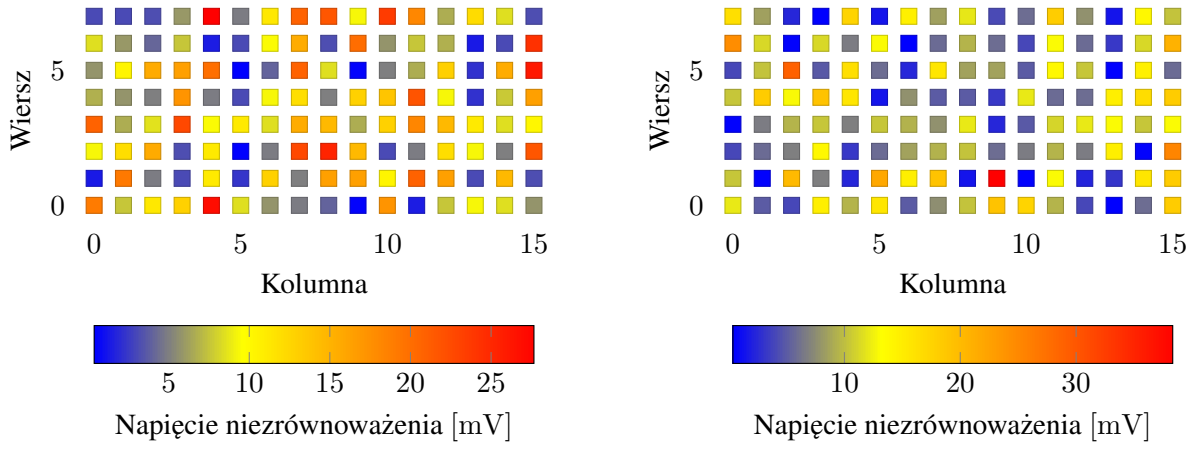
Wyniki wykonanych pomiarów impulsów generowanych przez szumy analogowego toru odczytowego zostały wykorzystane do wyznaczenia średniej i odchylenia standardowego napięć niezrównoważenia dyskryminatorów. W Tab. 4.6 przedstawione zostały wyznaczone parametry statystyczne. Ze względu na fakt,

iż w testowanym układzie scalonym PM zbudowana była z dwóch bloków o wymiarach 16 × 8 pikseli, parametry statystyczne zostały wyznaczone zarówno dla całej matrycy, jak i dla poszczególnych bloków. W dalszej części rozdziału blok znajdujący się dalej od bloku polaryzującego nazywany będzie „Blokiem A" a drugi komponent „Blokiem B".

Tablica 4.6: Wyniki pomiarów napięć niezrównoważenia.

| Parametr fizyczny | Parametr statystyczny | Blok A | Blok b | Cała matryca |
|---|---|---|---|---|
| Napięcie niezrównoważenia [mV] | Średnia | 1027,87 | 1018,57 | 1023,22 |
| | Odchylenie standardowe | 11,30 | 10,58 | 11,89 |
| Maksymalne liczby zliczeń [zliczenia] | Średnia | 6234,27 | 6256,79 | 6245,53 |
| | Odchylenie standardowe | 1177,80 | 1353,58 | 1268,79 |

Przedstawione na Rys. 4.13 przebiegi pozwalają na oszacowanie wymaganego zakresu napięć generowanych przez przetworniki korygujące w pikselach potrzebnego do skalibrowania układu. Na Rys. 4.14 przedstawione zostały bezwzględne wartości napięć niezrównoważenia dyskryminatorów dla poszczególnych pikseli.



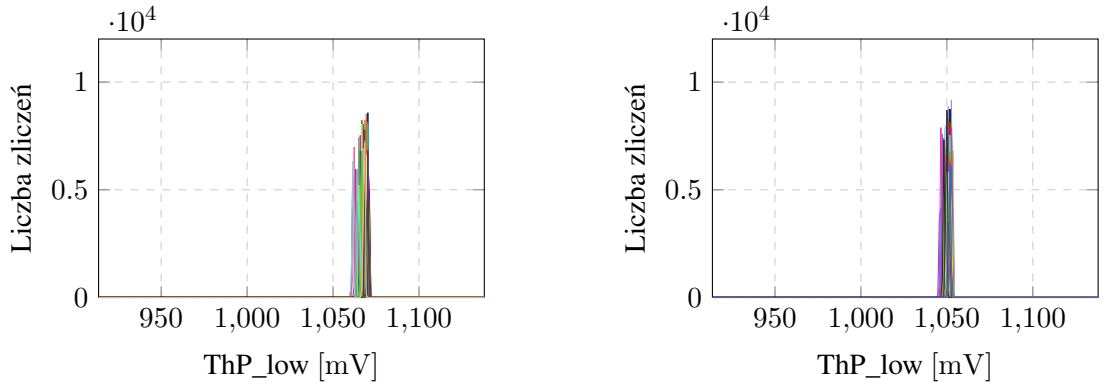

(a) Blok A.

(b) Blok B.

Rysunek 4.14: Bezwzględne wartości napięć niezrównoważenia dyskryminatorów.

W testowanym układzie scalonym, każdy dyskryminator posiada dedykowany 7-bitowy przetwornik korygujący napięcie niezrównoważenia. Ze względu na fakt, iż w docelowych aplikacjach rejestracja promieniowania realizowana jest dla jednej wartości globalnego napięcia korygującego, wykonuje się kalibrację napięć niezrównoważenia wszystkich pikseli do wspólnego poziomu. Na Rys. 4.15a oraz Rys.4.15b przedstawione zostały wyniki pomiarów impulsów szumowych zarejestrowanych dla układu skalibrowanego dla globalnych napięć polaryzujących wynoszących odpowiednio 984 mV oraz 1001,6 mV. Wartości te zostały wybrane ze względu na fakt, iż znajdowały się w środku zakresu napięciowego globalnego przetwornika polaryzującego.

Wyniki pomiarów wykonanych dla układu skalibrowanego dla globalnego napięcia polaryzującego wynoszącego 1001,6 mV zostały poddane tej samej analizie statystycznej, której poddane zostały wyniki po-




(a) Pomiar dla układu skalibrowanego dla globalnego napięcia polaryzującego 984 mV.

(b) Pomiar dla układu skalibrowanego dla globalnego napięcia polaryzującego 1001,6 mV.

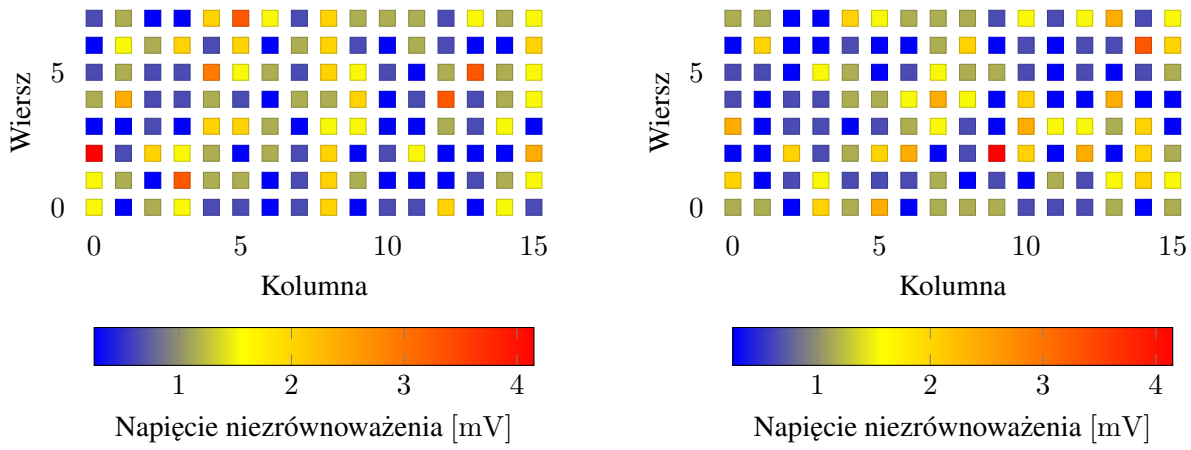
Rysunek 4.15: Wyniki pomiarów impulsów generowanych przez szumy skalibrowanego układu.

miaru układu nieskalibrowanego, a wyniki tej analizy przedstawione zostały w Tab 4.7. Porównanie zawartości Tab 4.6 i Tab 4.7 wykazuje, że przeprowadzona procedura kalibracyjna pozwoliła na znaczną redukcję odchylenia standardowego napięcia niezrównoważenia z 11,89 mV do 1,30 mV. Na Rys. 4.16 przedstawione zostały wyniki pomiaru napięć niezrównoważenia dla skalibrowanego układu.

Tablica 4.7: Wyniki pomiarów napięć niezrównoważenia dla układu skalibrowanego.

| Parametr fizyczny | Parametr statystyczny | Blok A | Blok b | Cała matryca |
|---|---|---|---|---|
| Napięcie niezrównoważenia [mV] | Średnia | 1050,66 | 1050,61 | 1050,63 |
| | Odchylenie standardowe | 1,34 | 1,27 | 1,30 |
| Maksymalne liczby zliczeń [zliczenia] | Średnia | 6242,35 | 6266,91 | 6254,63 |
| | Odchylenie standardowe | 1311,67 | 1258,47 | 1285,40 |

Wykonane eksperymenty wykazały, że prezentowany w niniejszej rozprawie scalony system mikroprocesorowy dla hybrydowych detektorów promieniowania jonizującego może być skutecznie wykorzystywany do autokalibracji układów odczytowych.




(a) Blok A.

(b) Blok B.

Rysunek 4.16: Bezwzględne wartości napięć niezrównoważenia dyskryminatorów skalibrowanego układu.

# Rozdział 5

# Podsumowanie

Celem badań przeprowadzonych przez Autora niniejszej rozprawy było wykorzystanie zalet przetwarzania brzegowego w hybrydowych detektorach promieniowania jonizującego. Paradygmat ten został wdrożony w opracowanym przez Autora systemie mikroprocesorowym dla hybrydowego pikselowego detektora promieniowania poprzez integrację mikroprocesora RISC-V i pikselowego układu odczytowego sensora we wspólnym podłożu krzemowym. Opracowane rozwiązanie pozwoliło, między innymi na odczyt wyników rejestracji, filtrację danych odczytywanych z matrycy i korekcję napięć niezrównoważenia dyskryminatorów wchodzących w skład analogowych torów odczytowych.

Wynikiem badań przeprowadzonych przez Autora niniejszej rozprawy były:

1. scalony system mikroprocesorowy integrujący mikroprocesor RISC-V i układ odczytowy dla hybrydowego pikselowego detektora promieniowania jonizującego (Rozdział 3),
2. środowisko do testowania opracowanego systemu mikroprocesorowego (Rozdział 4),
3. 32-bitowy mikroprocesor RISC-V (Sekcja 3.3).

Autor rozprawy zrealizował następujące zadania:

- zaprojektowanie architektury sprzętowej układu scalonego, integrującego mikroprocesor RISC-V z układem odczytowym dla hybrydowego detektora promieniowania jonizującego (Sekcja 3.1),
- implementacja opracowanej architektury sprzętowej w języku SystemVerilog, w szczególności układu peryferyjnego sterującego układem odczytowym dla hybrydowego detektora pikselowego (Podsekcja 3.1.4),
- opracowanie, przy wykorzystaniu języka C++, oprogramowania układowego (ang. firmware), sterującego zaprojektowanym systemem mikroprocesorowym (Sekcja 3.2),
- opracowanie środowiska projektowego, zawierającego skrypty wspomagające projektowanie i weryfikację projektowanego układu scalonego (skrypty powłoki systemowej oraz język Python),
- przygotowanie, napisanych w języku TCL, skryptów do syntezy i implementacji topografii (ang. place & route) układu scalonego,

- przeprowadzenie syntezy i implementacji topografii oraz weryfikacji układu scalonego,
- opracowanie prototypowego modelu projektowanego układu scalonego do implementacji w układzie FPGA,
- budowa zautomatyzowanego systemu testów regresyjnych opartego na platformie *GitHub*, odpowiedzialnego za automatyczne uruchamianie symulacji, syntezy i implementacji układu scalonego oraz syntezy i wgrywania prototypu do FPGA, a także za uruchomianie testów weryfikujących wgrany prototyp. Wszystkie wymienione testy były automatycznie uruchamianie po zaktualizowaniu zawartości przechowywanego na tej platformie repozytorium, zawierającego model HDL projektowanego układu scalonego oraz pliki źródłowe oprogramowania układowego,
- opracowanie architektury środowiska do testowania opracowanego systemu mikroprocesorowego, składającego się z układu FPGA, sterownika i aplikacji przestrzeni użytkownika,
- zaprojektowanie układu cyfrowego implementowanego w FPGA, pośredniczącego w komunikacji pomiędzy PC a zaprojektowanym układem scalonym, komunikującego się z PC za pośrednictwem interfejsu PCIe Gen. 3 x8 (Sekcja 4.1),
- opracowanie sterownika dla Linuksa (modułu jądra) do komunikacji z układem cyfrowym zaimplementowanym w FPGA (Podsekcja 4.2.1),
- zaprojektowanie interaktywnej aplikacji sterującej skonstruowanym środowiskiem testowym (C++ i biblioteka Qt), pozwalającej, między innymi na programowanie opracowanego układu scalonego, przesyłanie komend do tego układu i wizualizację danych transmitowanych przez testowany układ (Podsekcja 4.2.2),
- opracowanie aplikacji testowej (C++ i biblioteka GoogleTest), pozwalającej na automatyczne uruchamianie zbiorów testów weryfikujących zaprojektowany układ scalony (Podsekcja 4.2.2),
- zaprojektowanie mikroprocesora RISC-V o architekturze RV32I (Sekcja 3.3).

Realizacja badań przeprowadzonych przez Autora niniejszej rozprawy wymagała bardzo szerokiego zakresu wiedzy z dziedziny programowania (oprogramowanie układowe, sterowniki Linuksa i aplikacje przestrzeni użytkownika), projektowania systemów cyfrowych w FPGA i systemów mikroprocesorowych oraz projektowania specjalizowanych układów scalonych.

Wszystkie założone przez Autora cele zostały zrealizowane. Wyniki prac zostały zaprezentowane na konferencjach *International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)* [62] i *International Workshop on Radiation Imaging Detectors (iWoRiD)* [63, 64] oraz opublikowane w czasopiśmie *Journal of Instrumentation* [65]. Przedstawiony w niniejszej rozprawie scalony system mikroprocesorowy był, według najlepszej wiedzy Autora, pierwszym na świecie układem integrującym we wspólnym podłożu krzemowym mikroprocesor RISC-V i układ odczytowy dla hybrydowego pikselowego detektora promieniowania jonizującego.

Prezentowany w niniejszej rozprawie scalony system mikroprocesorowy dla hybrydowych detektorów promieniowania był prototypem mającym na celu sprawdzenie słuszności implementacji paradygmatu przetwarzania brzegowego w tego typu układach. W opinii Autora opracowane rozwiązanie ma wiele potencjalnych możliwości dalszego rozwoju w różnych kierunkach, takich jak:

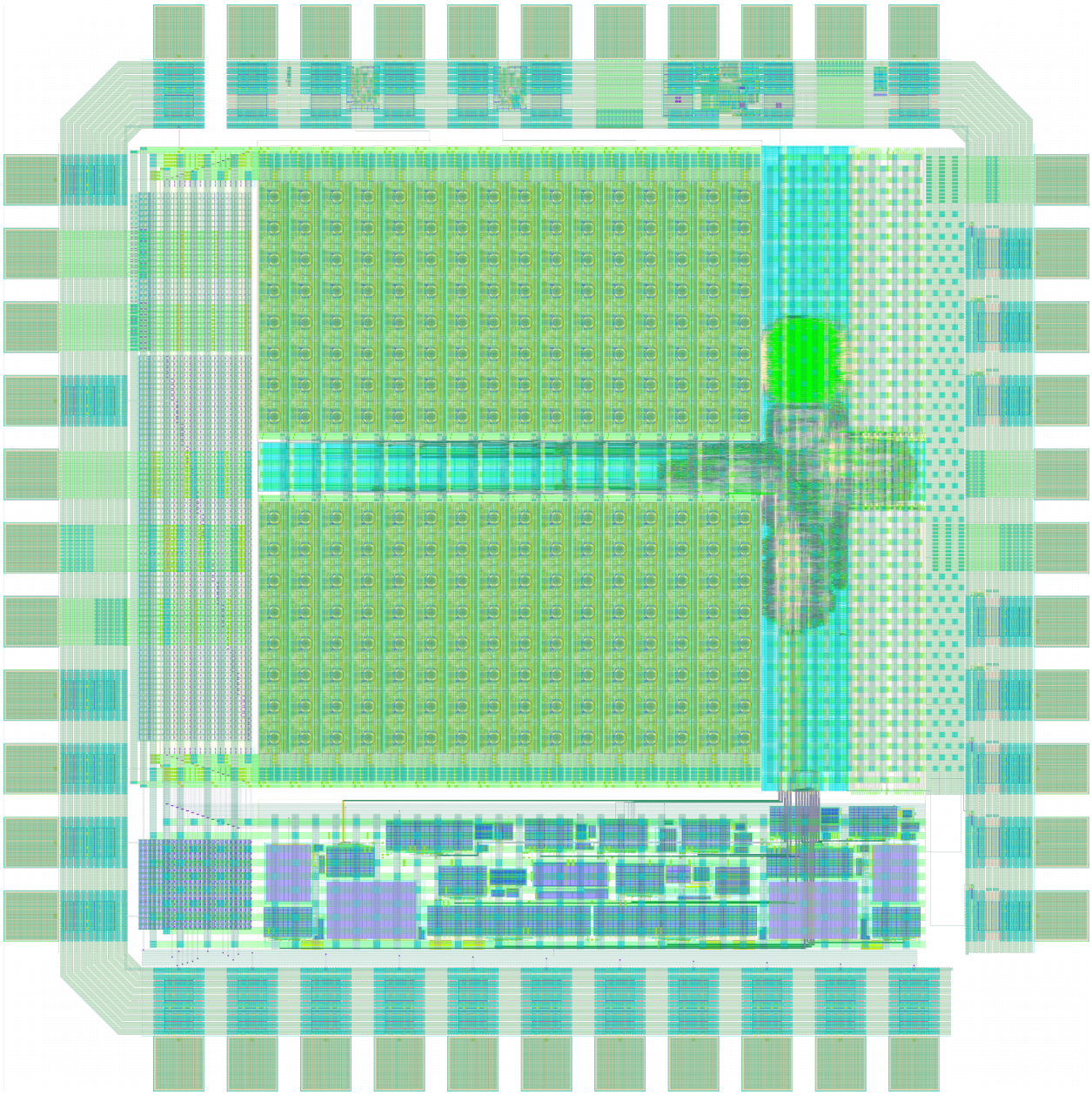
- budowa systemów do obserwacji procesów o dużej zmienności w czasie,
- konstrukcja autonomicznych systemów detekcyjnych,
- budowa systemów o zaawansowanym przetwarzaniu danych, np. integracja z sieciami neuronowymi.

Autor planuje kontynuację badań nad dalszym rozwojem systemów mikroprocesorowych w rozważanych kierunkach.

# Dodatek A

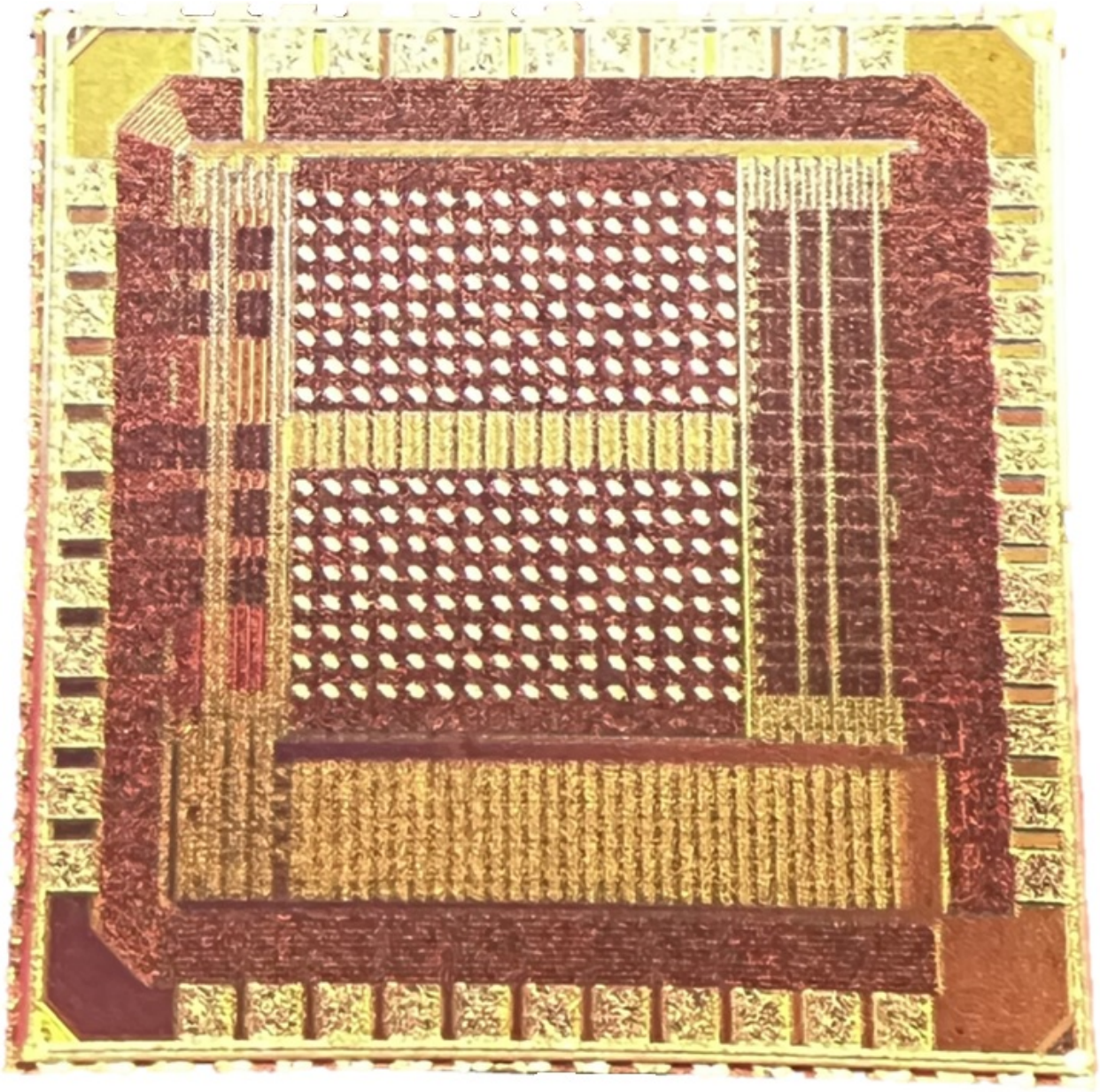
# Rysunki o wysokiej rozdzielczości

## A.1 Plan masek zaprojektowanego układu scalonego



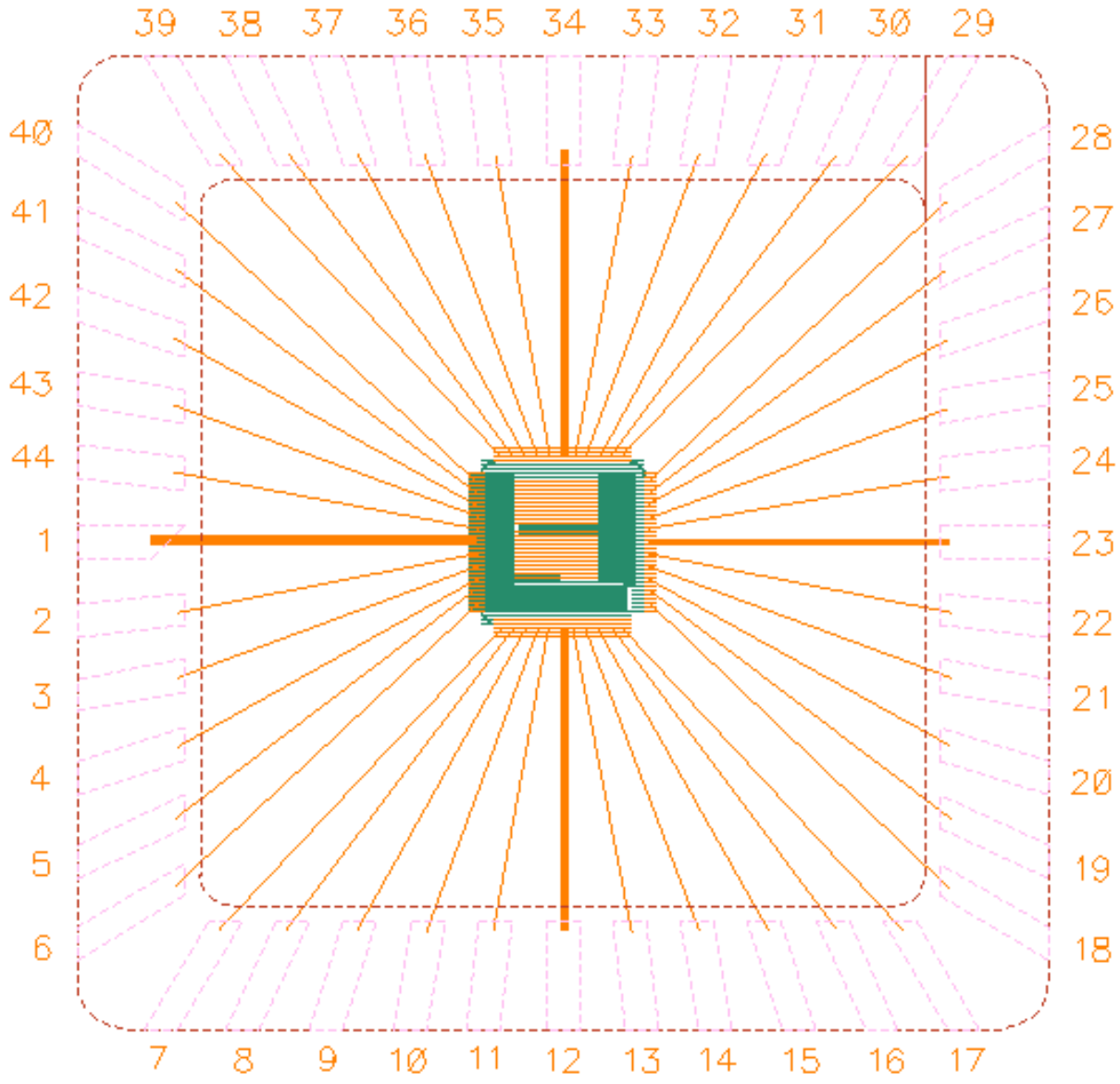
Rysunek A.1: Plan masek zaprojektowanego układu scalonego.

## A.2 Fotografia mikroskopowa zaprojektowanego układu scalonego



Rysunek A.2: Fotografia mikroskopowa zaprojektowanego układu scalonego.

## A.3 Schemat montażu i pakowania zaprojektowanego układu scalonego




Rysunek A.3: Schemat montażu i pakowania zaprojektowanego układu scalonego.

# Spis rysunków

# Spis tablic

# Spis listingów

# Bibliografia


[1] F. Faggin i in. “The history of the 4004”. W: *IEEE Micro* 16.6 (1996), s. 10–20. DOI: 10.1109/40.546561.

[2] W. Aspray. “The Intel 4004 microprocessor: what constituted invention?” W: *IEEE Annals of the History of Computing* 19.3 (1997), s. 4–15. DOI: 10.1109/85.601727.

[3] Arijit Biswas. “Sapphire Rapids”. W: *2021 IEEE Hot Chips 33 Symposium (HCS)*. 2021, s. 1–22. DOI: 10.1109/HCS52781.2021.9566865.

[4] Sarah Harris i David Harris. *Digital Design and Computer Architecture: RISC-V Edition*. 1st. Morgan Kaufmann, 2021. ISBN: 9780128200643.

[5] B. Razavi. *Fundamentals of Microelectronics, 2nd Edition*. Wiley, 2013. ISBN: 9781118559574. URL: https://books.google.pl/books?id=Lvr8ngEACAAJ.

[6] David Patterson i John Hennessy. *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. 1st. Morgan Kaufmann, 2017. ISBN: 9780128122754.

[7] Brian Grayson i in. “Evolution of the Samsung Exynos CPU Microarchitecture”. W: *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 2020, s. 40–51. DOI: 10.1109/ISCA45697.2020.00015.

[8] Junichiro Kadomoto, Hidetsugu Irie i Shuichi Sakai. “Evaluation of Different Microarchitectures for Energy-Efficient RISC-V Cores”. W: *2022 IEEE 15th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. 2022, s. 78–84. DOI: 10.1109/MCSoC57363.2022.00022.

[9] A.D. George. “An overview of RISC vs. CISC”. W: *[1990] Proceedings. The Twenty-Second Southeastern Symposium on System Theory*. 1990, s. 436–438. DOI: 10.1109/SSST.1990.138185.

[10] Emily Blem, Jaikrishnan Menon i Karthikeyan Sankaralingam. “Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures”. W: *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. 2013, s. 1–12. DOI: 10.1109/HPCA.2013.6522302.

[11] Herbert Bos Andrew S. Tanenbaum. *Modern Operating Systems*. 5th. Pearson, 2023. ISBN: 9780137618873.

[12] E. Koutroulis, K. Kalaitzakis i N.C. Voulgaris. "Development of a microcontroller-based, photovoltaic maximum power point tracking control system". W: *IEEE Transactions on Power Electronics* 16.1 (2001), s. 46–54. DOI: 10.1109/63.903988.

[13] E. Koutroulis i K. Kalaitzakis. "Design of a maximum power tracking system for wind-energy-conversion applications". W: *IEEE Transactions on Industrial Electronics* 53.2 (2006), s. 486–494. DOI: 10.1109/TIE.2006.870658.

[14] Suayb Cagri Yener i Resat Mutlu. "A microcontroller-based ECG signal generator design utilizing microcontroller PWM output and experimental ECG data". W: *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*. 2018, s. 1–4. DOI: 10.1109/EBBT.2018.8391465.

[15] Artur Cyba, Hubert Szolc i Tomasz Kryjak. "A simple vision-based navigation and control strategy for autonomous drone racing". W: *2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2021, s. 185–190. DOI: 10.1109/MMAR49549.2021.9528463.

[16] Brad Burres i in. "Intel's Hyperscale-Ready Infrastructure Processing Unit (IPU)". W: *2021 IEEE Hot Chips 33 Symposium (HCS)*. 2021, s. 1–16. DOI: 10.1109/HCS52781.2021.9567455.

[17] Naru Sundar i in. "9.4 An In-depth Look at the Intel IPU E2000". W: *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. 2023, s. 162–164. DOI: 10.1109/ISSCC42615.2023.10067333.

[18] Debendra Das Sharma. "Compute Express Link®: An open industry-standard interconnect enabling heterogeneous data-centric computing". W: *2022 IEEE Symposium on High-Performance Interconnects (HOTI)*. 2022, s. 5–12. DOI: 10.1109/HOTI55740.2022.00017.

[19] Donghyun Gouk i in. "Memory Pooling With CXL". W: *IEEE Micro* 43.2 (2023), s. 48–57. DOI: 10.1109/MM.2023.3237491.

[20] Andrew Waterman i in. "The RISC-V instruction set". W: *2013 IEEE Hot Chips 25 Symposium (HCS)*. 2013, s. 1–1. DOI: 10.1109/HOTCHIPS.2013.7478332.

[21] RISC-V International. *History of RISC-V*. 2023. URL: https://riscv.org/about/history/ (term. wiz. 12. 08. 2023).

[22] RISC-V International. *History of RISC-V*. 2023. URL: https://riscv.org/members/ (term. wiz. 12. 08. 2023).

[23] Google. *Popularność hasła „risc-v" w okresie 01.01.2010 - 12.08.2023*. 2023. URL: https://trends.google.com/trends/explore?date=2010-01-01%202023-08-12&q=risc-v&hl=pl (term. wiz. 12. 08. 2023).

[24] A. Waterman i K. Asanovic. *The RISC-V Instruction Set Manual*. 2019. URL: https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf (term. wiz. 01. 06. 2023).

[25] A. Dawiec i in. “New Fast Photon Counting Hybrid Pixel Detector for Synchrotron Applications Developed at SOLEIL Synchrotron”. W: *2021 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. 2021, s. 1–5. DOI: 10.1109/NSS/MIC44867.2021.9875574.

[26] P. Maj i in. “HyPix-3000 - a large area single-photon counting detector with two discriminator thresholds”. W: *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. 2014, s. 1–4. DOI: 10.1109/NSSMIC.2014.7431095.

[27] Jan Dudak i in. “Applicability of Large-Area Single-Photon Counting Detectors Timepix for High-Resolution and High-Contrast X-Ray Imaging of Biological Samples”. W: *IEEE Transactions on Nuclear Science* 69.4 (2022), s. 753–760. DOI: 10.1109/TNS.2022.3140396.

[28] R. Ballabriga i in. “Review of hybrid pixel detector readout ASICs for spectroscopic X-ray imaging”. W: *Journal of Instrumentation* 11.01 (sty. 2016), P01007. DOI: 10.1088/1748-0221/11/01/P01007. URL: https://dx.doi.org/10.1088/1748-0221/11/01/P01007.

[29] P Maj. “Fast and precise algorithms for calculating offset correction in single photon counting ASICs built in deep sub-micron technologies”. W: *Journal of Instrumentation* 9.07 (lip. 2014), s. C07009. DOI: 10.1088/1748-0221/9/07/C07009. URL: https://dx.doi.org/10.1088/1748-0221/9/07/C07009.

[30] Paweł Skrzypiec i Robert Szczygieł. “Development of On-Chip Calibration for Hybrid Pixel Detectors”. W: *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. 2021, s. 133–136. DOI: 10.1109/DDECS52668.2021.9417021.

[31] P. Grybos i in. “32k Channel Readout IC for Single Photon Counting Pixel Detectors with 75 $\mu$m Pitch, Dead Time of 85 ns, 9 $e^-$ rms Offset Spread and 2 % rms Gain Spread”. W: *IEEE Transactions on Nuclear Science* 63.2 (2016), s. 1155–1161. DOI: 10.1109/TNS.2016.2523260.

[32] Bartosz Tutro, Kacper Urbański i Robert Szczygieł. “Design of Matrix Controller for Hybrid Pixel Detectors”. W: *2018 25th International Conference "Mixed Design of Integrated Circuits and System"(MIXDES)*. 2018, s. 140–144. DOI: 10.23919/MIXDES.2018.8436909.

[33] Bartosz Tutro i Kacper Urbański. *Design of pixel matrix controller*. Bachelor’s thesis. Kraków, PL, sty. 2018.

[34] T Poikela i in. “Timepix3: a 65K channel hybrid pixel readout chip with simultaneous ToA/ToT and sparse readout”. W: *Journal of Instrumentation* 9.05 (maj 2014), s. C05013. DOI: 10.1088/1748-0221/9/05/C05013. URL: https://dx.doi.org/10.1088/1748-0221/9/05/C05013.

[35] Roberto Dinapoli i in. “EIGER: Next generation single photon counting detector for X-ray applications”. W: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 650.1 (2011). International Workshop on Semiconductor Pixel Detectors for Particles and Imaging 2010, s. 79–83. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2010.12.005. URL: https://www.sciencedirect.com/science/article/pii/S0168900210027427.

[36] M. Bochenek i in. “IBEX: Versatile Readout ASIC With Spectral Imaging Capability and High Count Rate Capability”. W: *IEEE Transactions on Nuclear Science* 65.6 (2018), s. 1285–1291. DOI: 10.1109/TNS.2018.2832464.

[37] lowRISC. *Ibex RISC-V Core*. 2023. URL: https://github.com/lowRISC/ibex (term. wiz. 01. 06. 2023).

[38] The Apache Software Foundation. *Apache License, version 2.0*. 2004. URL: https://www.apache.org/licenses/LICENSE-2.0 (term. wiz. 01. 06. 2023).

[39] lowRISC. *Ibex User Manual*. 2023. URL: https://ibex-core.readthedocs.io/en/latest (term. wiz. 01. 06. 2023).

[40] lowRISC. *Ibex Demo System*. 2023. URL: https://github.com/lowRISC/ibex-demo-system (term. wiz. 01. 06. 2023).

[41] P. Grybos i in. “32k Channel Readout IC for Single Photon Counting Pixel Detectors with 75 µm Pitch, Dead Time of 85 ns, 9 e- rms Offset Spread and 2 % rms Gain Spread”. W: *IEEE Transactions on Nuclear Science* 63.2 (2016), s. 1155–1161. DOI: 10.1109/TNS.2016.2523260.

[42] Intel Corporation. *Intel® Arria® 10 GX Development Kit*. 2023. URL: https://www.intel.com/content/www/us/en/products/details/fpga/development-kits/arria/10-gx.html (term. wiz. 01. 06. 2023).

[43] Intel Corporation. *Intel® Arria® 10 GX 1150 FPGA*. 2023. URL: https://www.intel.com/content/www/us/en/products/sku/210381/intel-arria-10-gx-1150-fpga/specifications.html (term. wiz. 01. 06. 2023).

[44] Intel Corporation. *Procesor Intel® Core™ i3-6300*. 2023. URL: https://www.intel.pl/content/www/pl/pl/products/sku/90731/intel-core-i36300-processor-4m-cache-3-80-ghz/specifications.html (term. wiz. 01. 06. 2023).

[45] Linux Kernel Organization. *The Linux Kernel Archives*. 2023. URL: https://www.kernel.org (term. wiz. 01. 06. 2023).

[46] Fedora Community. *Fedora Linux*. 2023. URL: https://fedoraproject.org (term. wiz. 01. 06. 2023).

[47] Intel Corporation. *Intel® Arria® 10 and Intel® Cyclone® 10 PCIe Hard IP*. 2023. URL: https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/pcie-a10-c10-hard-ip.html (term. wiz. 01. 06. 2023).

[48] Intel Corporation. *Oprogramowanie Intel® Quartus® Prime*. 2023. URL: https://www.intel.pl/content/www/pl/pl/products/details/fpga/development-tools/quartus-prime.html (term. wiz. 01. 06. 2023).

[49] Intel Corporation. *Avalon® Interface Specifications*. 2023. URL: https://www.intel.com/content/www/us/en/docs/programmable/683091/22-3/introduction-to-the-interface-specifications.html (term. wiz. 01. 06. 2023).

[50] Intel Corporation. *IOPLL Intel® FPGA IP Core User Guide*. 2023. URL: https://www.intel.com/content/www/us/en/docs/programmable/683285/18-1/core-user-guide.html (term. wiz. 01. 06. 2023).

[51] The kernel development community. *Devres - Managed Device Resource*. 2023. URL: https://docs.kernel.org/driver-api/driver-model/devres.html (term. wiz. 01. 06. 2023).

[52] The kernel development community. *PCI Bus Subsystem*. 2023. URL: https://docs.kernel.org/PCI/index.html (term. wiz. 01. 06. 2023).

[53] The kernel development community. *Miscellaneous Devices*. 2023. URL: https://docs.kernel.org/driver-api/misc_devices.html?highlight=miscellaneous+devices (term. wiz. 01. 06. 2023).

[54] Michael Kerrisk. *syscalls(2) — Linux manual page*. 2023. URL: https://man7.org/linux/man-pages/man2/syscalls.2.html (term. wiz. 01. 06. 2023).

[55] The Qt Company. *Qt*. 2023. URL: https://www.qt.io (term. wiz. 01. 06. 2023).

[56] Emanuel Eichhammer. *QCustomPlot*. 2023. URL: https://www.qcustomplot.com (term. wiz. 01. 06. 2023).

[57] Google. *GoogleTest User's Guide*. 2023. URL: http://google.github.io/googletest (term. wiz. 01. 06. 2023).

[58] W. Zubrzycka. *Printed Circuit Board for the PXRV chip*. 2023.

[59] Keithley Instruments. *2231A-30-3 195W Triple Channel DC Power Supply*. 2014.

[60] Keithley Instruments. *2110 5½-Digit Dual-Display Digital Multimeter*.

[61] STMicroelectronics. *NUCLEO-L476RG*. 2023. URL: https://www.st.com/en/evaluation-tools/nucleo-l476rg.html (term. wiz. 20. 09. 2023).

[62] Paweł Skrzypiec i Robert Szczygieł. "Development of On-Chip Calibration for Hybrid Pixel Detectors". W: *24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. 2021, s. 133–136. DOI: 10.1109/DDECS52668.2021.9417021.

[63] Paweł Skrzypiec i Robert Szczygieł. "Readout chip with RISC-V microprocessor for hybrid pixel detectors". W: *23rd International Workshop on Radiation Imaging Detectors*. 2022.

[64] Paweł Skrzypiec, Robert Szczygieł i Paweł Gryboś. "System for the Fast Readout and Tests of Pixel IC Operating in Single Photon Counting Mode using PCIe-based FPGA". W: *24th International Workshop on Radiation Imaging Detectors*. 2023.

[65] P. Skrzypiec i R. Szczygieł. "Readout chip with RISC-V microprocessor for hybrid pixel detectors". W: *Journal of Instrumentation* 18.01 (sty. 2023), s. C01030. DOI: 10.1088/1748-0221/18/01/C01030. URL: https://dx.doi.org/10.1088/1748-0221/18/01/C01030.