Uniwersytet Warszawski

Wydzial Fizyki

Zaklad czastek i oddzialywan fundamentalnych

Testowanie prototypu procesora PAC 2

Karol Bunkowski

Praca magisterska

napisana pod kierunkiem

Prof. dr hab. Jana Królkowskiego

Warszawa 2001

Spis tresci

1 Akcelerator LHC, jego program naukowy i eksperyment CMS

1.1 Program badawczy LHC

LHC - Large Hadron Colider bedzie kolowym akceleratorem w którym przyspieszane i zderzane beda przeciwbiezne wiazki protonów lub ciezkich jonów. Jego uruchomienie planowane jest na 2006 rok. Bedzie to najpotezniejsze urzadzenie sposród wszystkich, jakie do tej pory powstaly na potrzeby fizyki wysokich energii. Powinno ono umozliwic dalsze, lepsze zrozumienie swiata czastek elementarnych.

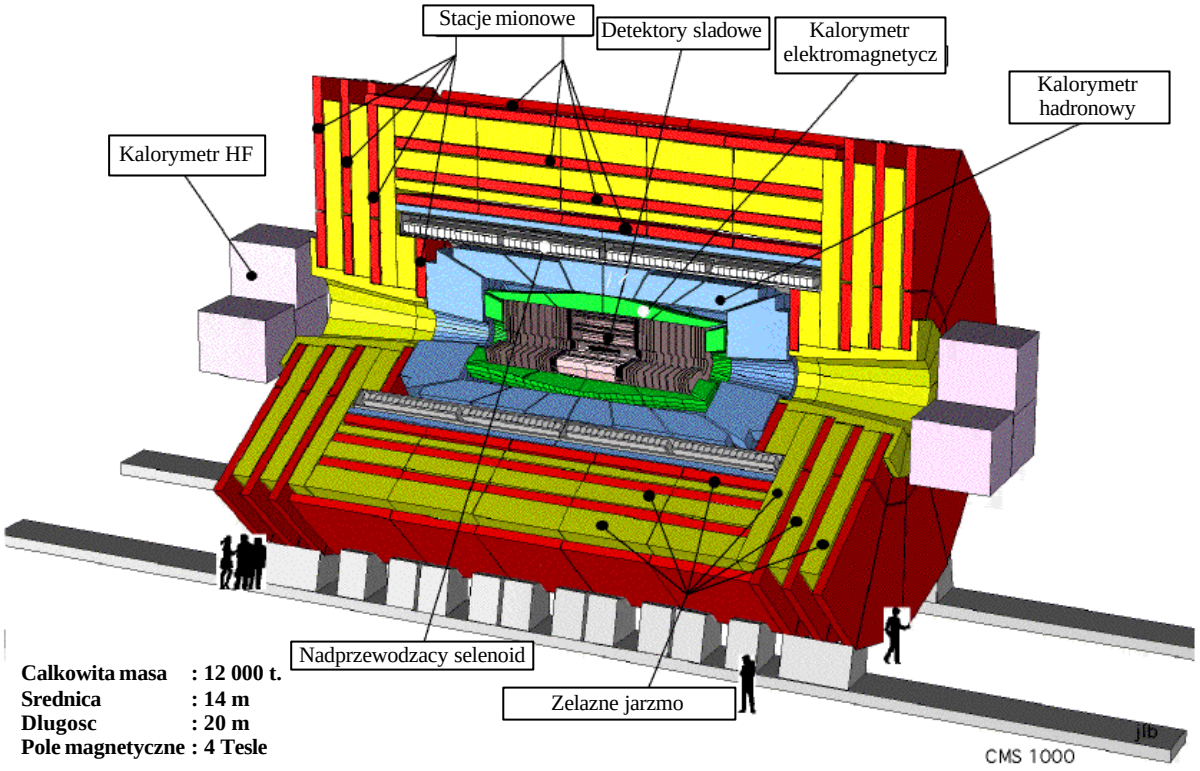
Wspólczesna wiedza o fundamentach budowy materii zawiera sie w tzw. modelu standardowym. Jego przewidywania sa znakomicie zgodne z wynikami doswiadczalnymi. Do tej pory jednak nie zostal odkryty bozon Higgsa, bardzo wazny element tego modelu, odpowiedzialny za nadawanie masy innym czastkom. Mimo swoich niewatpliwych sukcesów model standardowy posiada tez jednak pewne wady: ma duza liczbe wolnych parametrów, nie daje odpowiedzi na podstawowe pytania takie jak np.: dlaczego obserwuje sie trzy generacje fermionów, dlaczego ladunek elektryczny jest skwantowany. Dlatego rozpatrywane sa rozszerzenia modelu standardowego, np. modele supersymetryczne, lub inne nowe teorie, np. teorie wyzszych wymiarów. Konsekwencja przewidywan tych teorii jest istnienie nowych, do tej pory nie odkrytych czastek. To wlasnie odkrycie i zbadanie czastki Higgsa oraz czastek spoza modelu standardowego jest najwazniejszym celem eksperymentu LHC. Ze wzgledu na to, ze przewidywane masy tych czastek sa bardzo duze, a przekroje czynne na ich produkcje male, do ich wytworzenia potrzebna jest duza energia i swietlnosc. Dlatego w akceleratorze LHC protony przyspieszane beda do energii 7 TeV, a jego docelowa swietlnosc powinna wyniesc 10^{34} $cm^{-2}s^{-1}$. Aby uzyskac taka swietlnosc, paczki, w które beda zgrupowane protony, przecinac sie beda co 25 ns, w kazdym przecieciu zachodzic bedzie ok. 20 zderzen proton-proton, w wyniku których wyprodukowanych zostanie kilkaset czastek.

Program LHC przewiduje równiez dzialanie z wiazkami ciezkich jonów (jadra od wapnia do olowiu). Celem tych eksperymentów jest badanie innego interesujacego zagadnienia – istnienia i wlasnosci plazmy kwarkowo-gluonowej. W tym przypadku zderzenia paczek zachodzic beda co 125 ns, ale w ich wyniku produkowanych bedzie znacznie wiecej czastek.

Tak wysoka energia i czestosc zderzen stawia bardzo duze wymagania detektorom, jakie beda dzialac przy LHC: powinny one charakteryzowac sie m.in. duza granularnoscia, malym czasem martwym, odpornoscia na wysokie promieniowanie, krótkim czasem odpowiedzi.

1.2 Budowa detektora CMS

Detektor CMS – Compact Muon Selenoid bedzie jednym z czterech detektorów przy akceleratorze LHC. Zbudowany on bedzie wedlug schematu klasycznego dla wspólczesnych detektorów stosowanych w fizyce wysokich energii (Rys. 1.1) [1]. Jego najwazniejszym elementem jest nadprzewodzacy selenoid o dlugosci 13 m i srednicy 6 m. Pole magnetyczne przez niego wytwarzane bedzie tak silne (4 T wewnatrz, 2 T na zewnatrz), ze umozliwi precyzyjny pomiar pedu czastek o energii do kilku TeV.



Rys. 1.1: Detektor CMS.

Wewnatrz selenoidu znajda sie:

- detektory sladowe: pikslowe i paskowe detektory krzemowe,
- kalorymetry: elektromagnetyczny (ECAL), oparty na kryształach $PbWO_4$ i hadronowy (HCAL), zlozony z plastikowych scyntylatorów i mosiadzu.

Na zewnatrz selenoidu umieszczone beda cztery warstwy stacji minowych. Kazda ze stacji mionowych w beczce ($0 < |\eta| < 1.0$) skladac sie bedzie z komór dryfowych (ang. Drift Tube - DT) i z jednej lub dwóch plaszczyzn komór RPC (ang. Resistive Plate Chambers). Natomiast kazda ze stacji z pokryw ($1 < |\eta| < 2.1$) zlozona bedzie z komór CSC (ang. Cathode Strip Chamber) i z jednej plaszczyzny komory RPC. Stacje mionowe przedzielone beda zelaznymi plytami zamykajacymi strumien pola magnetycznego i pelniacymi role absorbenta.

Tuz przy rurze akceleratora umieszczone zostana dodatkowe kalorymetry hadronowe (Very Forward Calorimeter - HF).

Detektor bedzie umieszczony w hali eksperymentalnej znajdujacej sie ok. 100 metrów pod ziemia. Bezposrednio na detektorze bedzie umieszczona tylko niezbedna elektronika odczytujaca i na pewien czas gromadzaca sygnaly, wieksza czesc elektroniki znajdowac sie bedzie w innej podziemnej hali tzw. counting house, polaczonym z detektorem poprzez wysokoprzepustowe lacza optyczne.

2 System wyzwalania

Kluczowym elementem eksperymentu jest system wyzwalania (tryger) [2]. Detektor zbiera ok. 1 MB danych z kazdego przeciecia paczek, co daje ogromny strumien 40 GB danych na sekunde, praktycznie niemozliwy do zapisania na zadnych, dostepnych nosnikach. Jednak wiekszosc z tych zdarzen to produkcja dobrze juz znanych, a przez to malo interesujacych przypadków. Zadaniem systemu wyzwalania jest wiec wybranie z poczatkowej liczby 40 milionów przeciec na sekunde ok. 100 przypadków/s, w których mogly pojawic sie interesujace zdarzenia. Sygnatura umozliwiajaca odróznienie tych zdarzen od przypadków tla sa m.in.: wysokoenergetyczne leptony, fotony lub dzety hadronowe oraz brakujaca energia poprzeczna bedaca wynikiem produkcji slabo oddzialujacych czastek uciekajacych z detektora.

2.1 Stopnie trygera

W eksperymencie CMS system wyzwalania zostal podzielony na dwa glówne stopnie. Stopien pierwszy oparty jest na dedykowanej elektronice. Ze wzgledu na wymagana szybkosc dzialania analizuje on jedynie zgrubne dane z kalorymetrów i systemu mionowego. Chociaz decyzja o odrzuceniu przypadku musi zapadac co 25 ns, to na jej podjecie pierwszy stopien trygera, dzieki tzw. przetwarzaniu potokowemu, ma 3,2 μs. W tym czasie calosc danych o przypadku przechowywana jest w pamieciach buforowych (rejestrach przesuwnych). Jesli dany przypadek zostanie zaakceptowany przez pierwszy stopien, dane te sa przekazywane do trygera wyzszego stopnia, który bedzie realizowany przez farme kilku tysiecy komercyjnych procesorów. Przypadki zaakceptowane przez tryger wyzszego stopnia beda zapisywane w masowych pamieciach i pózniej analizowane. Zadaniem trygera pierwszego stopnia jest zredukowanie czestosci zdarzen do ok. 30 kHz (przy maksymalnej swietlnosci), tryger wyzszego stopnia powinien obnizyc ta czestosc do ok. 100 Hz.

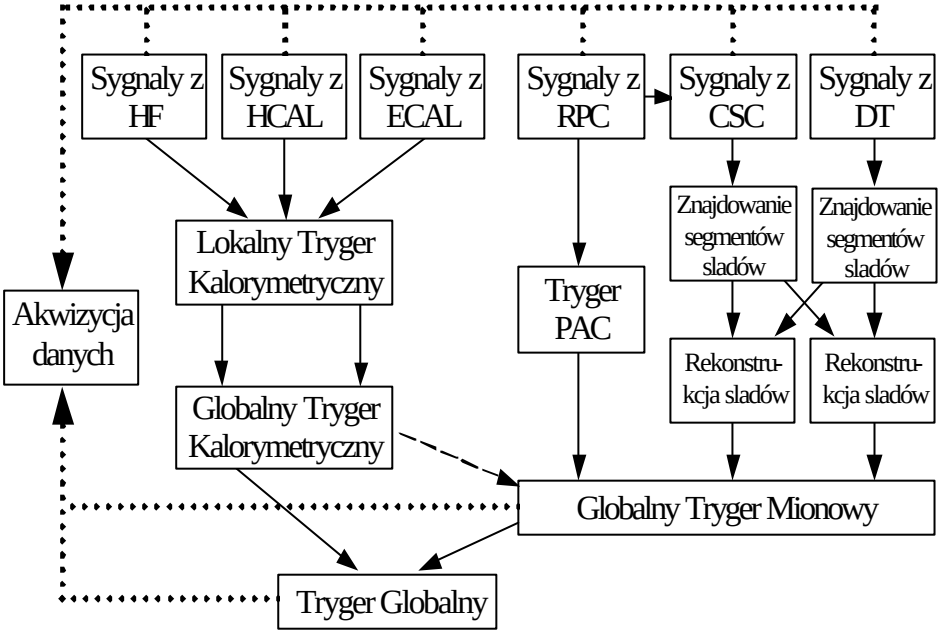
2.2 System wyzwalania pierwszego stopnia

Pierwszy stopien trygera sklada sie z trzech podstawowych podsystemów: trygera kalorymetrycznego, trygera mionowego oraz trygera globalnego (Rys. 2.1) [2]. Trygery kalorymetryczny i mionowy nie podejmuja decyzji o odrzuceniu lub zaakceptowaniu danego przypadku, ich zadaniem jest odnalezienie okreslonych obiektów i przekazanie informacji o ich energii i wspólrzednych do trygera globalnego. Tryger kalorymetryczny sortuje zidentyfikowane elektrony, fotony, leptony τ oraz dzety, sposród kazdego typu wybiera po 4 o najwyzszej energii, a takze oblicza wektor calkowitej brakujacej energii poprzecznej. Tryger mionowy natomiast odnajduje 4 miony o najwyzszym pedzie. Tryger globalny sprawdza, czy obiekty te maja energie powyzej ustalonych progów, a dzieki temu, ze zna równiez ich wspólrzedne, moze róznicowac progi w zaleznosci od obszaru detektora z którego pochodza, moze równiez uwzgledniac ich usytuowanie wzgledem siebie. Wartosci progów sa optymalizowane tak, aby uzyskac jak najwieksza efektywnosc przy równoczesnym zachowaniu wymaganego stopnia redukcji czestosci zdarzen.

Rozpady poszukiwanych nowych czastek na miony daja sygnaly, które moga byc stosunkowo latwo odróznialne od tla. Dodatkowo miony sa czastkami o duzej przenikalnosci, dlatego system komór mionowych umieszcza sie w zewnetrznych obszarach detektora, gdzie nie docieraja zadne inne czastki, gdyz sa wczesniej pochlaniane. Ulatwia to znacznie

identyfikacje mionów i pomiar ich pedu. Z tych wzgledów system mionowy jest niezwykle waznym elementem calego eksperymentu, a w szczególnosci systemu wyzwalania.

W sklad trygera mionowego wchodza dwa, uzupelniajace sie podsystemy: jeden oparty na detektorach DT i CSC, drugi na komorach RPC, dedykowanych specjalnie dla systemu wyzwalania. Detektory DT i CSC zapewniaja precyzyjny pomiar toru czastek, kluczowy dla dokladnego wyznaczenia ich pedu. Natomiast detektory RPC charakteryzuja sie (szczególnie w przeciwienstwie do DT) znakomita rozdzielczoscia czasowa, umozliwiajaca jednoznaczne przypisanie czastki do przeciecia paczek, oraz wysoka granularnoscia pozwalajaca na prace przy wielkiej liczbie czastek. Prawidlowe polaczenie tych dwóch systemów owocuje wysoka efektywnoscia i wydajnym odrzucaniem tla, umozliwia równiez wzajemne monitorowanie i kalibracje.



Rys. 2.1: Schemat pierwszego stopnia trygera

Trygery DT i CSC maja za zadanie znalezc w kazdym przecieciu paczek po 4 miony o najwyzszym pedzie, tryger RPC powinien odszukac 4 najszybsze miony w beczce i 4 w pokrywach. Informacje z kazdego z tych trygerów (ped mionów, wspólrzedne, jakosc sladów) sa przekazywane do globalnego trygera mionowego, który wybiera 4 najlepsze czastki i przekazuje je do trygera globalnego.

2.3 Tryger RPC

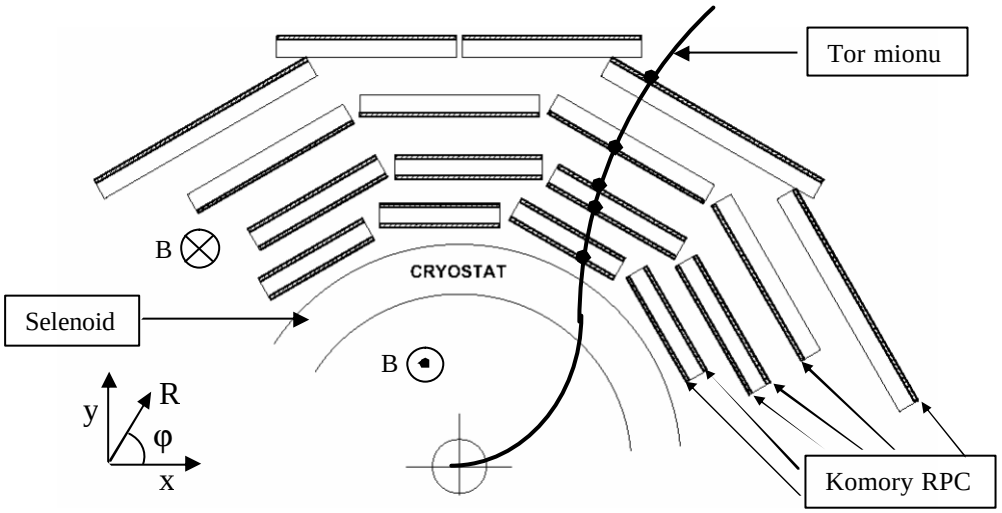
2.3.1 Zasada dzialania

Aby tryger RPC spelnil postawione przed nim zadania, tzn. potrafil wydajnie odnajdywac wysokoenergetyczne miony, musi wykonywac symultanicznie trzy podstawowe funkcje:

- z wysoka efektywnoscia identyfikowac slady mionów,
- jednoznaczne przypisac znalezione miony do przeciecia, z którego pochodza,

- na tyle dokladnie wyznaczyc ped, aby mozliwe bylo wyróznienie najbardziej energetycznych mionów.

Dla potrzeb trygera najwazniejszy jest dokladny pomiar zakrzywienia torów mionów w polu magnetycznym, a nie precyzyjne wyznaczenie ich kierunków. Ze wzgledu na konfiguracje pola magnetycznego wytworzonego przez selenoid miony zakrzywiane beda w plaszczyznie R-φ. Aby zbadac to zakrzywienie wystarczy dokladnie zmierzyc dla kilku punktów toru wspólrzedna φ (Rys. 2.2). Dlatego komory RPC zaproponowane do eksperymentu CMS beda komorami paskowymi, umozliwiajacymi odczyt punktu przejscia czastki tylko w jednym wymiarze. W beczce tworzyc one beda szesc cylindrycznych warstw, a paski odczytowe biec beda równolegle do osi wiazki. Natomiast w kazdej z dwóch pokryw komory RPC ulozone beda w cztery warstwy prostopadle do osi wiazki, a paski beda biec radialnie. Paski beda mialy szerokosc kilku centymetrów tak, aby kazdy z nich pokrywal 5/16° we wspólrzednej φ.



Rys. 2.2: Przekrój poprzeczny przez beczke detektora CMS. Tor mionu jest zakrzywiany przez pole magnetyczne wytwarzane przez selenoid. Komory RPC mierza punkty przejscia mionu. Paski komór biegna prostopadle do plaszczyzny rysunku.

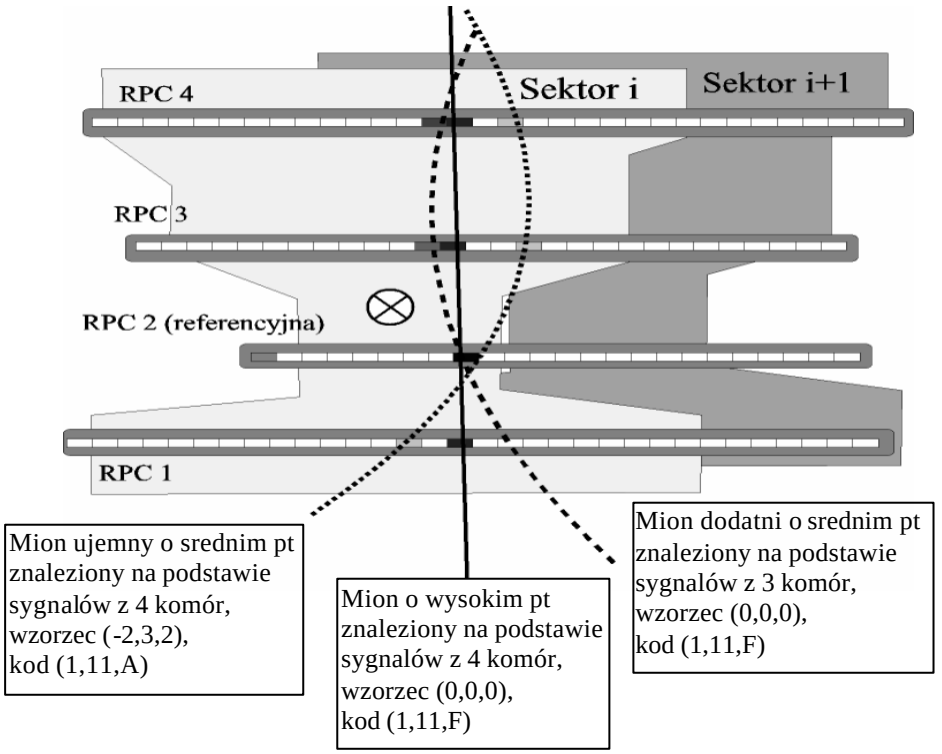
Najprostszym i najszybszym algorytmem umozliwiajacym zrekonstruowanie toru mionu na podstawie sygnalów z komór RPC jest tzw. PAtern Compataror (PAC). Idea jego dzialania polega na porównaniu sygnalów z komór z wczesniej zdefiniowanymi wzorcami (paternami) sygnalów, jakie moglyby powstac po przejsciu wysokoenergetycznych mionów (Rys. 2.3). Wzorce te zostaly znalezione z komputerowych symulacji, kazdemu z nich zostala przypisana wartosc pedu poprzecznego mionu. Oczywiscie mozliwych wzorców jest bardzo duzo, dodatkowo, ze wzgledu na wielokrotne rozpraszanie i fluktuacje energii miony wyemitowane w tym sam kierunku z jednakowym pedem moga pozostawic rózne slady. Wzorce porzadkuje sie grupujac wszystkie wzorce zawierajace jeden pasek wybranej komory, zwanej komora referencyjna.

Przeprowadzone symulacje pokazaly [3], ze aby zrekonstruowac tor mionu wystarczy odnalezc czasowa koincydencji sygnalów z czterech komór RPC (slady 4/4). Aby jednak zwiekszyc efektywnosc, uwzgledniane sa takze koincydencje sygnalów z jedynie trzech komór (slady 3/4). Sa one jednak traktowane jako slady o nizszej jakosci niz slady 4/4. Na podstawie symulacji stwierdzono równiez, ze dla zwiekszenia efektywnosci konieczne jest uwzglednienie mionów o nieco nizszym pedzi, które w wyniku zakrzywienia toru w polu

magnetycznym nie dolatuja do zewnetrznych warstw komór. Dlatego dwie wewnetrzne stacje mionowe z beczki zawieraja po dwie warstwy komór RPC tak, aby równiez te niskopedowe miony przechodzily przez cztery komory.

Do róznych obszarów detektora potrzebne sa rózne zestawy wzorców, dlatego nie moga one byc wpisane do systemu trygera na stale, lecz powinna istniec mozliwosc ich programowania.

Rozwazano rózne sposoby implementacji algorytmu PAC w urzadzeniach elektronicznych (programowalne procesory FPGA, pamieci „Look Up Table"). Zdecydowano sie jednak na zaprojektowani i wyprodukowanie dedykowanego procesora ASIC (zwanego dalej PAC), gdyz wydaje sie, ze jest to najbardziej optymalna opcja ze wzgledu na mozliwosci i koszty. Procesor, którego przetestowanie bylo celem niniejszej pracy, jest druga wersja PACa. Zostal on wyprodukowany w technologii 35 μm, co umozliwilo zmieszczenie w nim ukladów analizujacych obszar detektora zawierajacy osiem pasków komory referencyjnej.

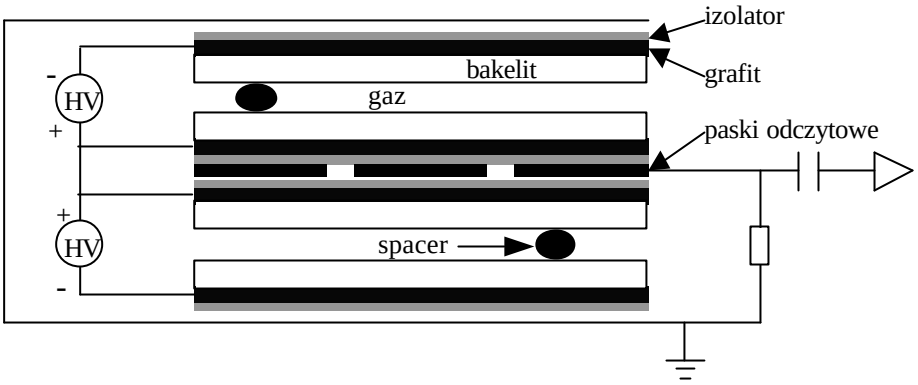


Rys. 2.3: Zasada dzialania procesora PAC – dopasowywanie torów do wzorców. Rysunek pokazuje równiez, jakie obszary poszczególnych komór RPC analizuje jeden procesor (jeden procesor analizuje jeden sektor, sasiadujace sektory czesciowo pokrywaja sie, jedynie paski stacji referencyjnej podlaczone sa tylko do jednego procesora).

2.3.2 Komory RPC

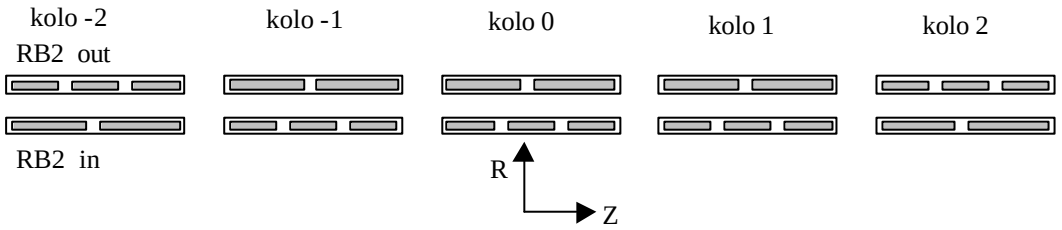
Komory RPC wykorzystywane w detektorze CMS skladaja sie z czterech bakelitowych plyt tworzacych dwa szczelne pudla gazowe o grubosci przerwy 2 mm, wypelnione mieszanka gazowa (freon z dodatkiem innych gazów) Rys. 2.4 [1]. Zewnetrze powierzchnie plyt pokryte sa warstwa grafitu, do którego podlaczone jest zródlo wysokiego napiecia. Pomiedzy wnekami znajduja sie metalowe paski umozliwiajace odczyt pozycji

powstania kaskady elektronowej wywolanej przejsciem naladowanej czastki. Efektywnosc takich komór siega 100%,W CMS komory RPC beda pracowaly w ograniczonym modzie proporcjonalnym.

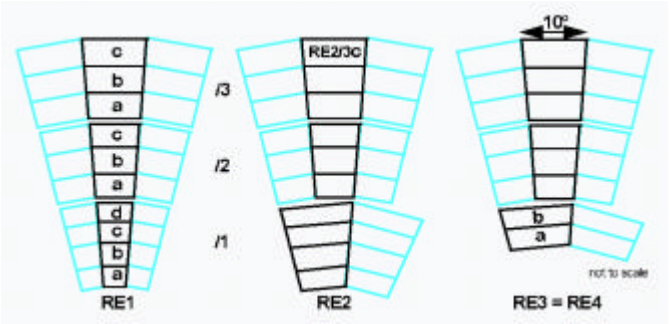


Rys. 2.4: Przekrój komory RPC o dwóch wnekach gazowych.

Stacje mionowe w beczce dziela sie na piec kól, z których kazde sklada sie z 12 sektorów obejmujacych po 30° w ϕ. Paski odczytowe komór RPC biegna równolegle do wiazki, ich szerokosc wynosi od 2 do 4 cm. Aby zapewnic prawidlowa segmentacje trygera w η, paski wewnatrz kazdego kola zostaly podzielone na dwie czesci, oprócz komór referencyjnych, gdzie zostaly podzielone na trzy czesci (Rys. 2.5) [4]. Jako komory referencyjne (patrz pp. 2.4.2) zostaly wybrane zewnetrzne RPC stacji drugiej dla kól –2 i 2 oraz wewnetrzne dla kól -1, 0, 1.

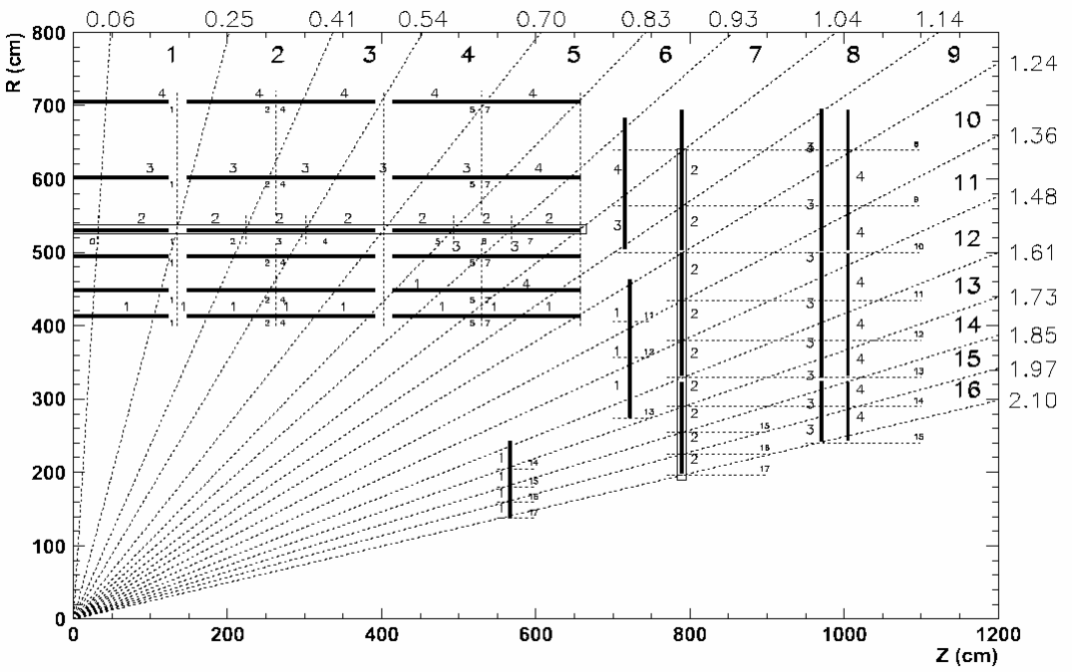


Rys. 2.5: Uklad komór RPC w stacji drugiej (RB2) w beczce. Paski komór referencyjnych podzielone sa na trzy czesci.



Rys. 2.6: Uklad komór RPC w pokrywach.

W pokrywach kazda z czterech warstw stacji mionowych dzieli sie w ϕ na 36 sektorów (oprócz najblizszych wiazce obszarów stacji 2, 3 i 4, podzielonych na 18 sektorów). W R stacje dziela sie sa na trzy czesci. Kazda czesc zawiera jedna komore RPC, której paski sa podzielone w η na 2, 3 lub 4 czesci (Rys. 2.6). Paski biegna radialnie, prostopadle do wiazki. Referencyjnymi sa komory ze stacji drugiej.

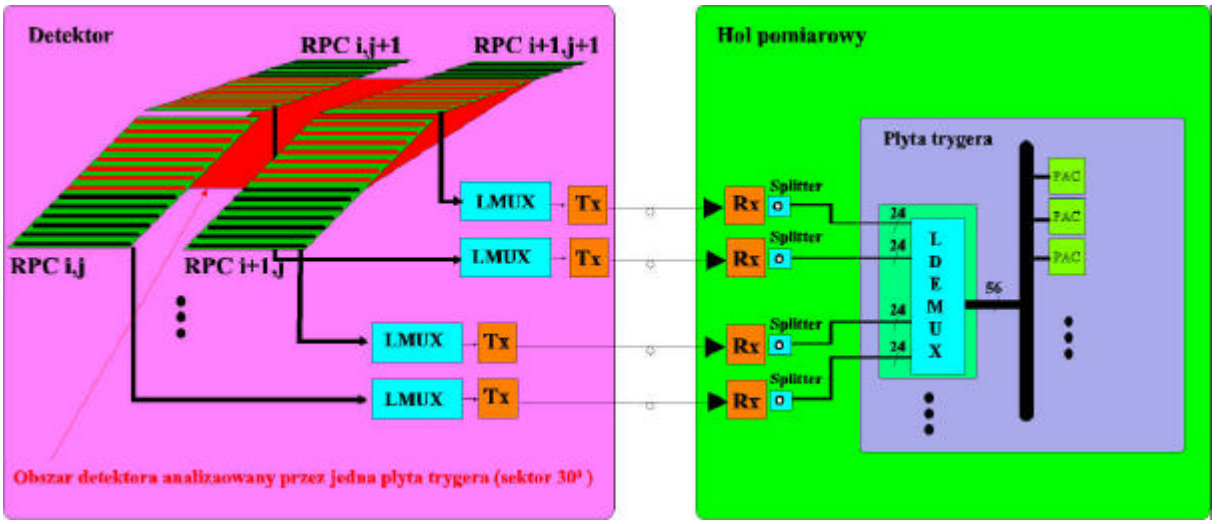


Rys. 2.7: Podzial komór RPC na wieze w η.

Caly system komór RPC zostal podzielony na potrzeby trygera na 33 wieze w η, definiowane przez dlugosc pasków komór referencyjnych (Rys. 2.7) Kazda wieza dzieli sie na 12 logicznych sektorów w ϕ.

2.3.3 System trygera RPC

Schemat ukladu trygera RPC przedstawia Rys. 2.8. Analogowe sygnaly z pasków komór konwertowane sa przez uklady FEC (Front End Chip) do postaci cyfrowej (1 – pasek „zapalony”, 0 – „nie zapalony”), nastepnie sa synchronizowane, pakowane (LMUX), urównoleglane i przesylane przez lacza optyczne do elektroniki decyzyjnej w podziemnym holu pomiarowym. Tam dane po odczytaniu przechodza przez tzw. spliter, który ma na celu tworzenie kilku kopii sygnalu wejsciowego, i nastepnie sa przekazywane do plyt trygera. Uklady LDMUX znajdujace sie na tych plytach dekompresuja i w odpowiedni sposób rozsylaja dane do procesorów PAC. Kazda plyta trygera zawiera 12 PACów i analizuje dane z jednego 30° sektora jednej wiezy. Zadaniem kazdego procesora PAC jest wyszukanie jednego sladu mionu o najwyzszym pedzie w segmencie 2.5° w ϕ na, zawierajacym osiem pasków komory referencyjnej. Slady znalezione przez wszystkie PACe z jednej plyty sa ghostbustowane i sortowane, cztery miony o najwyzszym pedzie przekazywane sa do dalszych czesci systemu. Czas przeznaczony na wykonanie wszystkich operacji to maksymalnie 2,025 μs (81 bx).



Rys. 2.8: Schemat trygera RPC.

3 Procesor PAC

3.1 Dane wejsciowe i odpowiedzi PACa

W omawianej wersji PAC podzielony jest na 9 bloków [5]. Kazdy z bloków 1 - 8 analizuje dane z obszaru zawierajacego jeden pasek komory referencyjnej, wykorzystujac do tego dane z komór: 1 (bity dalej zwane MS1_OR1), 2 (MS2_OR1), 3 (MS3_OR1) i 4 (MS4_OR1) (numeracje komór w kazdej wiezy przedstawia Rys. 2.7). Natomiast blok 9, wykorzystywany tylko w wiezach z beczki, sluzy do wyszukiwania mionów o nizszym pedzie, które ze wzgledu na zakrzywienie toru w polu magnetycznym nie dolecialy do zewnetrznych komór. Blok ten analizuje obszar zawierajacy 8 pasków komory referencyjnej, wykorzystujac dane z komór: 1 (bity MS1_OR1 oraz bity MS1_OR4 – kazdy bit odpowiada 4 paskom komory (logiczny OR4 obliczany przez LDMUXy)), 2 (MS2_OR1), 1p (MS1p_OR4) i 2p (MS2p_OR4) (Rys. 3.3).

Pelne wymuszenie PACa (dane wejsciowe) to 124 bity (Rys. 3.1). Bity te sa przekazywane do PACa w dwóch czesciach. Pierwsza czesc - bity nieparzyste, wysylane sa na rosnacym zboczu zegara, druga czesc - bity parzyste, wysylane sa na zboczu malejacym.

123 100	**76**	**66**	**58**	**46**	**32**	**0**
MS4_OR1 [24...1]	MS3_OR1 [24...1]	MS2p_OR4 [10...1]	MS2_OR1 [8...1]	MS1_OR4 [12...1]	MS1p_OR4 [14...1]	MS1_OR1 [32...1]

Rys. 3.1: Uklad bitów w wymuszeniu.

Odpowiedz PACa to 8 bitów. Pierwszy bit mówi o znaku sladu, nastepne dwa o jego jakosci wedlug schematu:

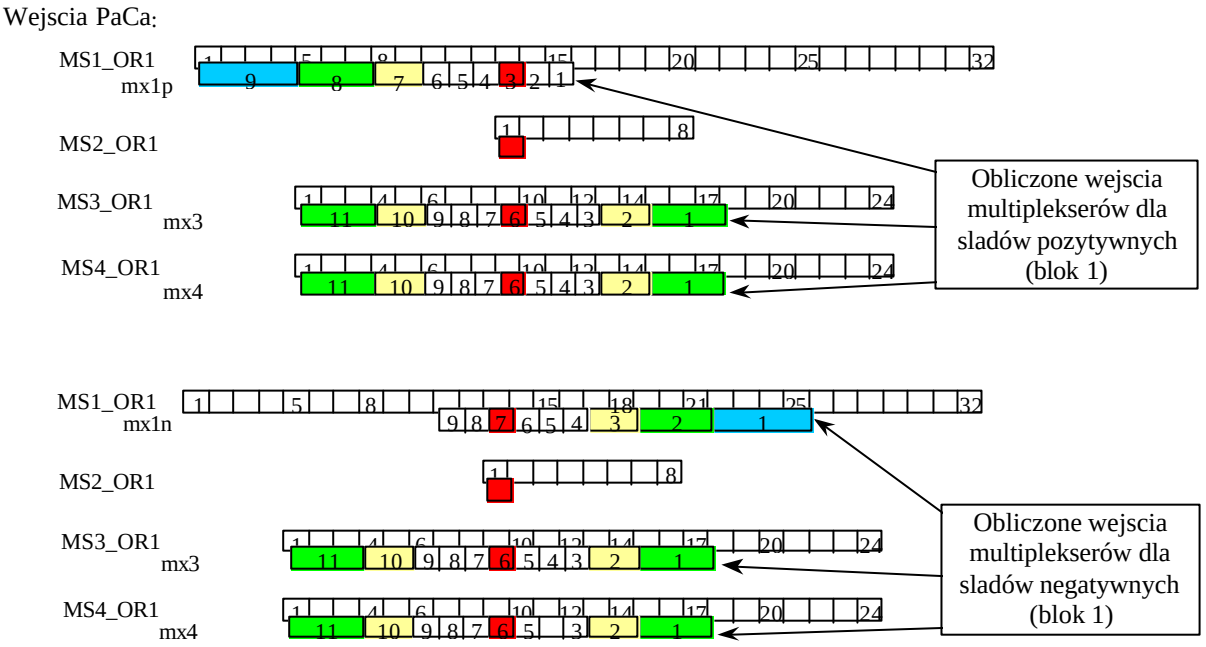
Jesli slad zostal zidentyfikowany przez bloki 1 – 8 (wysoki ped):

- 11 – slad zostal znaleziony na podstawie sygnalów z kazdej z czterech komór;
- 10 – slad zostal znaleziony na podstawie sygnalów z komór 1, 2, 3 lub 1,2,4;

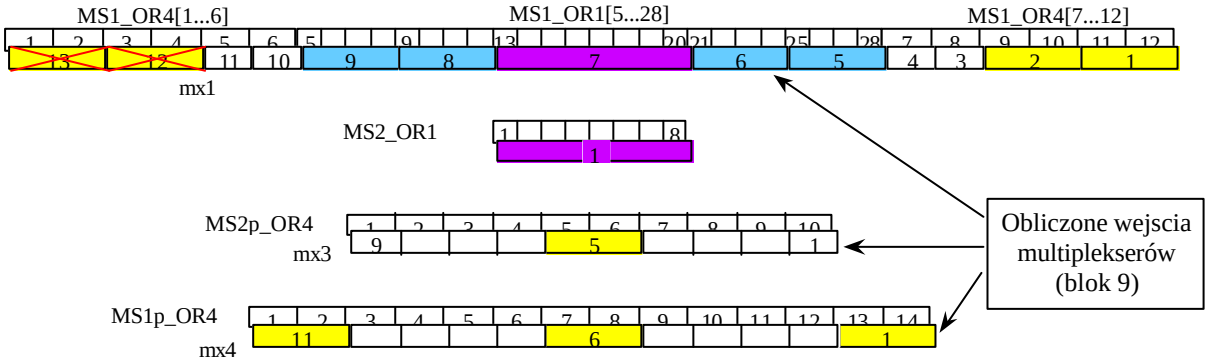
- 01 – slad zostal znaleziony na podstawie sygnalów z komór 2, 3, 4,
- 00 – slad zostal znaleziony na podstawie sygnalów z komór 1, 3, 4,

jesli slad zostal zidentyfikowany przez blok 9 (niski ped):

- 11 – slad zostal znaleziony na podstawie sygnalów z kazdej z czterech komór;
- 00 – slad zostal znaleziony na podstawie sygnalów z którychkolwiek trzech komór.



Rys. 3.2: Schemat rozsylania bitów wejsciowych do bloków 1-8. Do kazdego bloku trafia tylko czesc bitów wejsciowych, z niektórych obliczany jest OR (kolor niebieski – OR4, zielony - OR3, zólty - OR2). Rysunek przedstawia sposób obliczania wejsc bloku 1. Dla bloku drugiego schemat jest taki sam, jedynie przesuniety o jeden bit w prawo, dla bloku trzeciego o dwa bity, itd.



Rys. 3.3: Schemat obliczania wejsc bloku 9 z bitów wejsciowych. Kolor fioletowy oznacza OR8, niebieski – OR4, zólty OR2. W wyniku bledu w konstrukcji PACa wejscia nr 12 i 13 multiplekserów mx1 nie sa aktywne.

Wsród sladów o wysokim pedzie gradacja jakosci jest nastepujaca: najwyzsza jakosc to 11, potem 10, 01, i najnizsza 00.

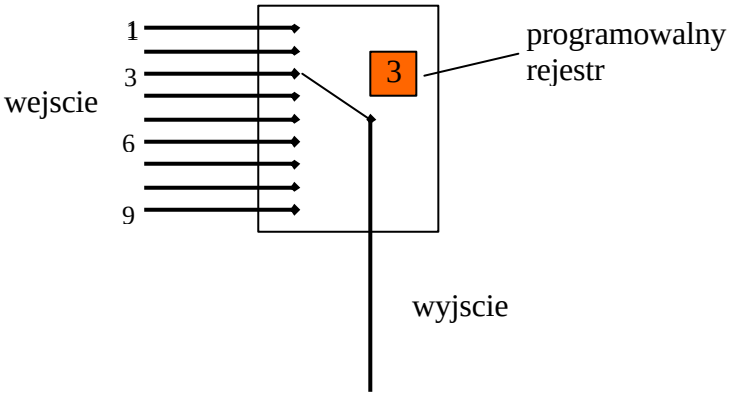
Pozostale piec bitów odpowiedzi to kod odpowiadajacy pedowi. Uklad, schemat oznaczenia i opis funkcji nózek PACa zawiera dodatek B.

3.2 Zasada dzialania procesora PAC

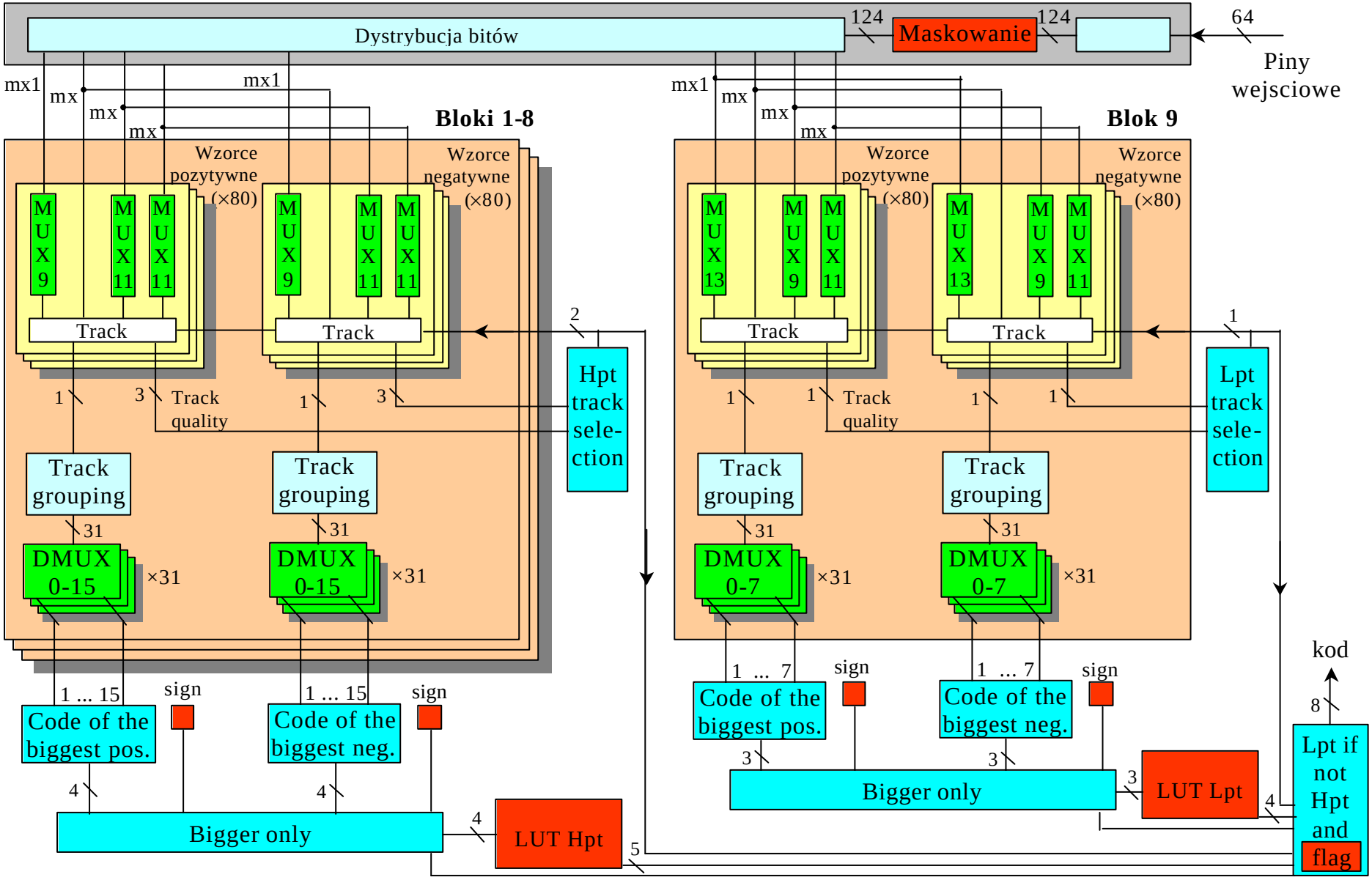
Ogólny schemat PACa przedstawia Rys. 3.5. Przed wlasciwym uzyciem PACa nalezy zaprogramowac, to znaczy wpisac w niego wzorce oraz kilka innych danych konfigurujacych jego dzialanie. W kazdy z dziewieciu bloków mozna wpisac 160 wzorców, po 80 dla sladów mionów dodatnich i ujemnych.

Bity wejsciowe jednego wymuszenia z dwóch zboczy zegara sa najpierw synchronizowane i do dalszych ukladów wysylane sa jednoczesnie. Kazdy bit moze zostac zablokowany przez programowalny uklad tzw. masek. Jesli bit maski ustawiony jest na „1”, wówczas odpowiadajacy mu bit wejsciowy jest zawsze ustawiany na „0”. Nastepnie bity wejsciowe sa dystrybuowane do bloków wedlug schematu z Rys. 3.2 i Rys. 3.3. Z bitów skrajnych obliczane sa logiczne OR, gdyz odpowiadaja one bardziej zakrzywionym sladom, dla których nie jest wymagana tak duza dokladnosc.

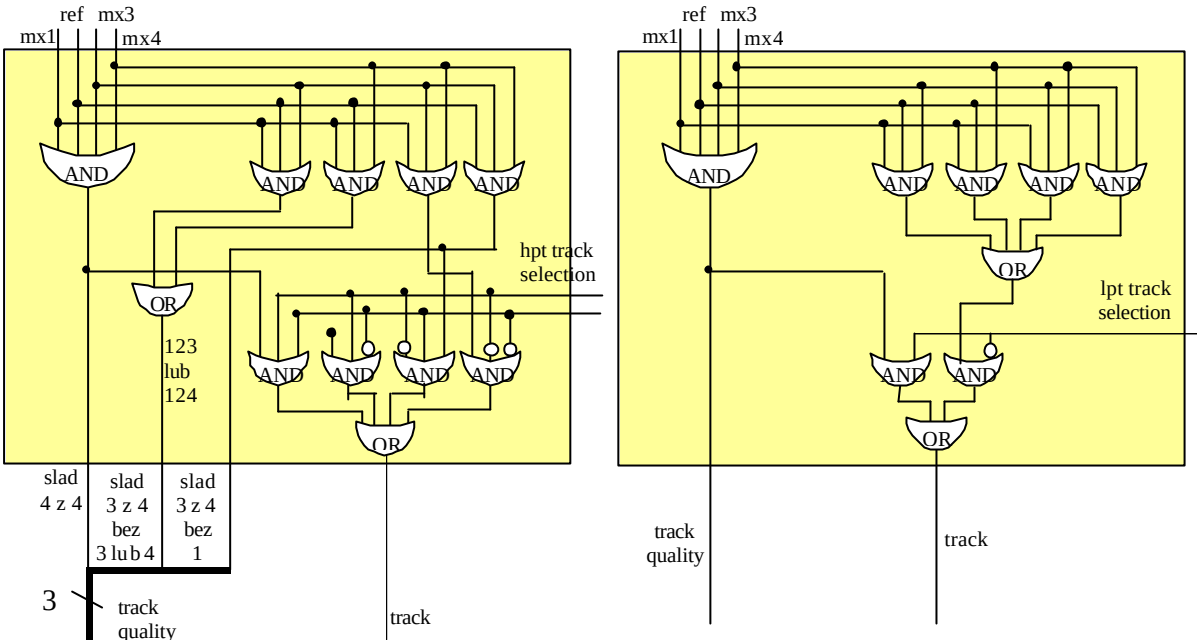
Wewnatrz bloków odbywa sie porównywanie bitów wejsciowych z zaprogramowanymi wzorcami. Jeden wzorzec to trzy liczby wpisane w programowalne rejestry trzech tzw. multiplekserów (MUX). Do kazdego multipleksera trafiaja bity z jednej plaszczyzny. Jesli bit wejsciowy wskazywany przez rejestr przyjmie wartosc „1”, to na wyjsciu multipleksera pojawi sie „1” (Rys. 3.4).



Rys. 3.4: Zasada dzialania multipleksera. Na wyjsciu pojawia sie „1” tylko wtedy, gdy wartosc bitu, którego numer zapisany jest w programowalnym rejestrze, jest „1”.



Rys. 3.5: Ogólny schemat procesora PAC. Uklady zaznaczone na zielono i czerwono sa programowalne (opis ukladów znajduje sie w tekscie)



Rys. 3.6: Schemat ukladu definicji torów (track signal circuit) w blokach 1 – 8 (rysunek po lewej) i w bloku 9 (rysunek po prawej).

Wyjscia trzech multiplekserów z danego wzorca oraz bit komory 2 wchodza do ukladu definicji sladu (Track signal circuit). Jego dzialanie przedstawia Rys. 3.6. Bity jakosci sladu (track quality) sa wynikiem logicznych AND z odpowiednich wyjsc multiplekserów i bitu komory 2. Natomiast bit „slad” („track”) uzyskuje wartosc „1” jesli znaleziony w danym wzorcu slad ma jakosc niemniejsza niz slad znaleziony w jakimkolwiek innym wzorcu. Z bitów jakosci sladu wszystkich wzorców (oddzielnie dla bloków 1-8 i bloku 9) obliczane sa przez uklad „Hpt track selection” lub „Lpt track selection” dwa bity „wybór sladu” (track selection). Sa to te same bity, które pojawiaja sie w odpowiedzi PACa (pp. 3.1). Ich wartosc odpowiada jakosci najlepszego znalezionego sladu (jednego lub wielu). Sygnaly „slad” sa nastepnie grupowane (oddzielnie w kazdym bloku, oddzielnie slady pozytywne i negatywne) wedlug zasady:

- wzorce 1-16 – OR16,
- wzorce 17-24 – OR8,
- wzorce 25-32 – OR8,
- wzorce 33-36 – OR4,
- wzorce 37-40 – OR4,
- wzorce 41-44 – OR4,
- wzorce 45-48 – OR4,
- wzorce 49-64 – z kazdych kolejnych dwóch obliczany jest OR2,
- wzorce 65-80 – bez zmian.

Pozwala to przypisac kilku róznym wzorcom ten sam kod pedu. W ten sposób 80 bitów odpowiadajacych sygnalom sladów jest redukowanych do 31. Wyniki kazdego OR trafiaja do programowalnych tzw. demultiplekserów (DMUX), w których, jesli w grupie

pojawil sie choc jeden slad, przypisywana jej jest (wczesniej zaprogramowana) liczba o wartosci 0-15 dla sladów wysokopedowych (bloki 1-8) i 0-7 dla sladów niskopedowych (blok9). Liczby te powinny byc w takie, aby sladom o wyzszym pedzie przypisana byla wieksza liczba. Nastepnie sposród liczb odpowiadajacych wszystkim sladom pozytywnym z bloków 1-8 wybierana jest najwieksza, podobnie wsród sladów negatywnych (uklady „Code of the biggest"). Kolejny uklad „Bigger only" porównuje te dwie liczby i wybiera wieksza. Wybrana liczba jest teraz numerem komórki w ukladzie pamieci LookUp Table (LUT) (tabela 16 pieciobitowych liczb o programowalnej wartosci), z której odczytywany jest zaprogramowany kod. Podobna procedura jest stosowana do sladów z bloku 9, z tym, ze w tym przypadku LUT jest tabela osmiu czterobitowych liczb. O tym, który z kodów (Hpt czy Lpt) znajdzie sie w koncowej odpowiedzi PACa decyduje uklad wyboru kodu („Lpt if not Hpt and mask"). Jesli programowalny bit „flag" ustawiony jest na „0", kod odpowiadajacy sladowi Lpt przechodzi wtedy, gdy ma wieksza jakosc niz slad Hpt. Jesli „flag" = „1", kod Lpt przechodzi jedynie wtedy, gdy nie zostal znaleziony zaden slad Hpt.

Bit znaku jest programowalny oddzielnie dla sladów pozytywnych i negatywnych czesci Hpt i Lpt. W koncowej odpowiedzi znajduje sie ten odpowiadajacy wybranemu przez PACa sladowi.

3.3 Programowalnosc i uklad sciezki brzegowej

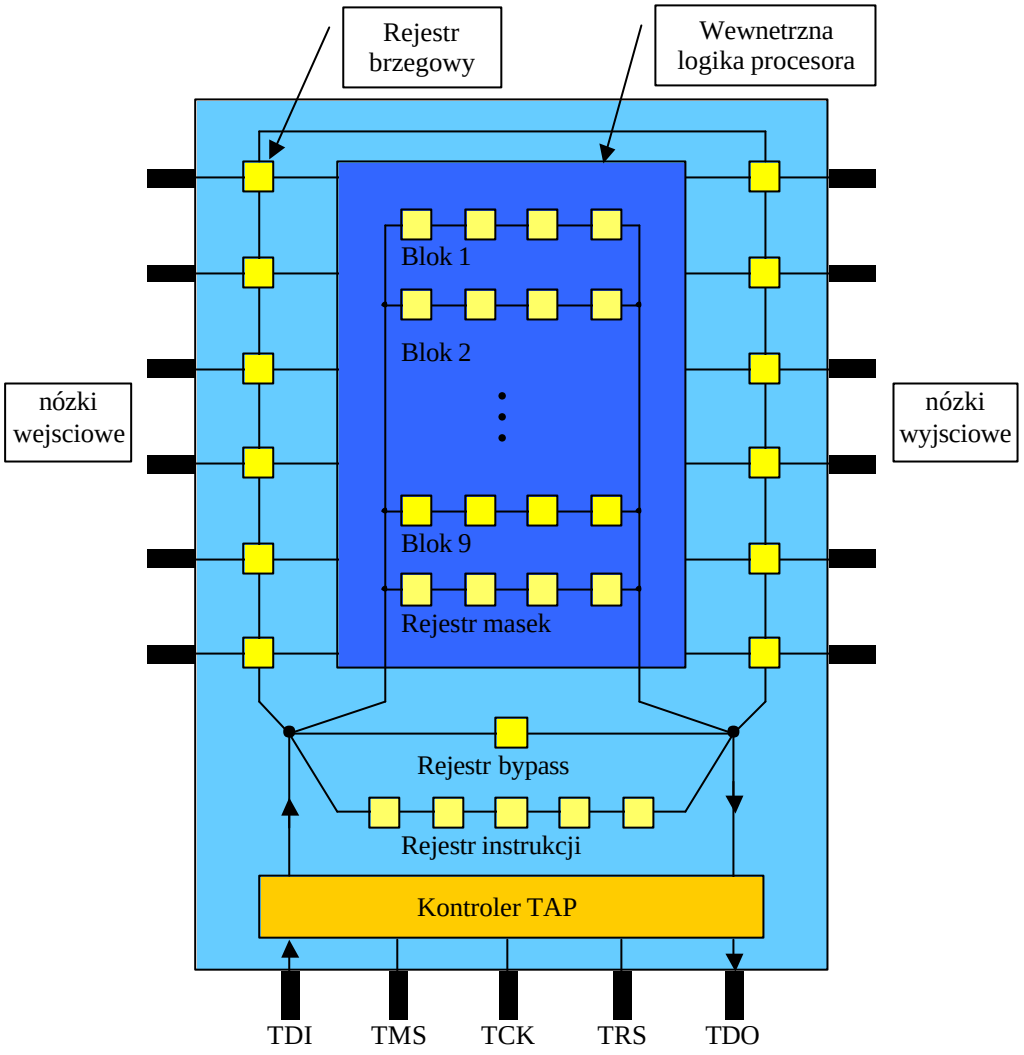
Programowanie PACa jest realizowane przy pomocy szeregowego interfejsu sciezki brzegowej [6]. System ten (ang. Test Acces Port and Boundary Scan Architecture, w skrócie Boundary Scan - BS) jest elektronicznym standardem stworzony pierwotnie na potrzeby latwego testowania ukladów elektronicznych i plytek (odczytywania i wymuszania stanów nózek ukladu scalonego), ale dzieki swojej uniwersalnosci moze byc równiez stosowany do innych celów. Podstawowymi elementami tego systemu wbudowywanymi w uklad scalony sa (Rys. 3.7) [7]:

- Cztery lub piec dodatkowych nózek: TDI (Test Data In), TDO (Test Data Out), TMS (Test Mode Select) i TCK (Test Clock) oraz opcjonalna nózka TRST (Test Reset) tworzace standardowa szyne TAP (Test Access Port);

- Kontroler TAP bedacy prosta maszyna stanów sterowana sygnalami TMS oraz TCK, który generuje sygnaly kontrolujace pozostale elementy Boundary-Scan;

- Rejestr Instrukcji, który jest uzywany do ustawienia zadanego trybu pracy i wybrania odpowiedniego rejestru danych. Moze on zostac podlaczony przez kontroler TAP pomiedzy nózki TDI oraz TDO w celu szeregowego wprowadzenia instrukcji lub odczytu danych kontrolnych (tzw. Capture Value). Czesc instrukcji jest narzucona przez standard (SAMPLE, EXTEST i BYPASS), czesc jest proponowana, ale opcjonalna (np. INTEST, IDCODE, HIGHZ, RUNBIST itd.). Istnieje równiez mozliwosc dodania wlasnych instrukcji.

- Grupa rejestrów okreslanych jako rejestry danych. W kazdym ukladzie musza byc przynajmniej 2 rejestry danych: Rejestr Brzegowy (Boundary Register), sluzacy do odczytywania i wymuszania stanów nózek wejsciowych oraz Rejestr Bypass (Bypass Register). Standard pozwala równiez na umieszczenie rejestrów opcjonalnych lub dodanie wlasnych. Rejestry danych umieszczane sa pomiedzy TDI oraz TDO w zaleznosci od instrukcji dekodowanej przez Rejestr Instrukcji oraz stanu kontrolera TAP.



Rys. 3.7: Schemat ukladu sciezki brzegowej w PACu.

W PACu umieszczono 11 dodatkowych rejestrów: 9 sluzacych do programowania wzorców w blokach, jeden tzw. rejestr masek sluzacy do programowania m.in. LUT, oraz rejestr Intscan, dzieki któremu mozna odczytac wartosc sygnalów „track" (bitów informujacych, czy pojawil sie slad odpowiadajacy wpisanemu wzorcowi, patrz pp. 3.2).

Zaprogramowanie PACa sklada sie z nastepujacych bitów:

Rejestr bloków:

- MUX – $9 \times 160 \times 3$ czterobitowych liczb;
- DMUX – 8×62 czterobitowych liczb (bloki 1 – 8);

  62 czterobitowe liczby (wykorzystywane 3 bity) (blok 9);

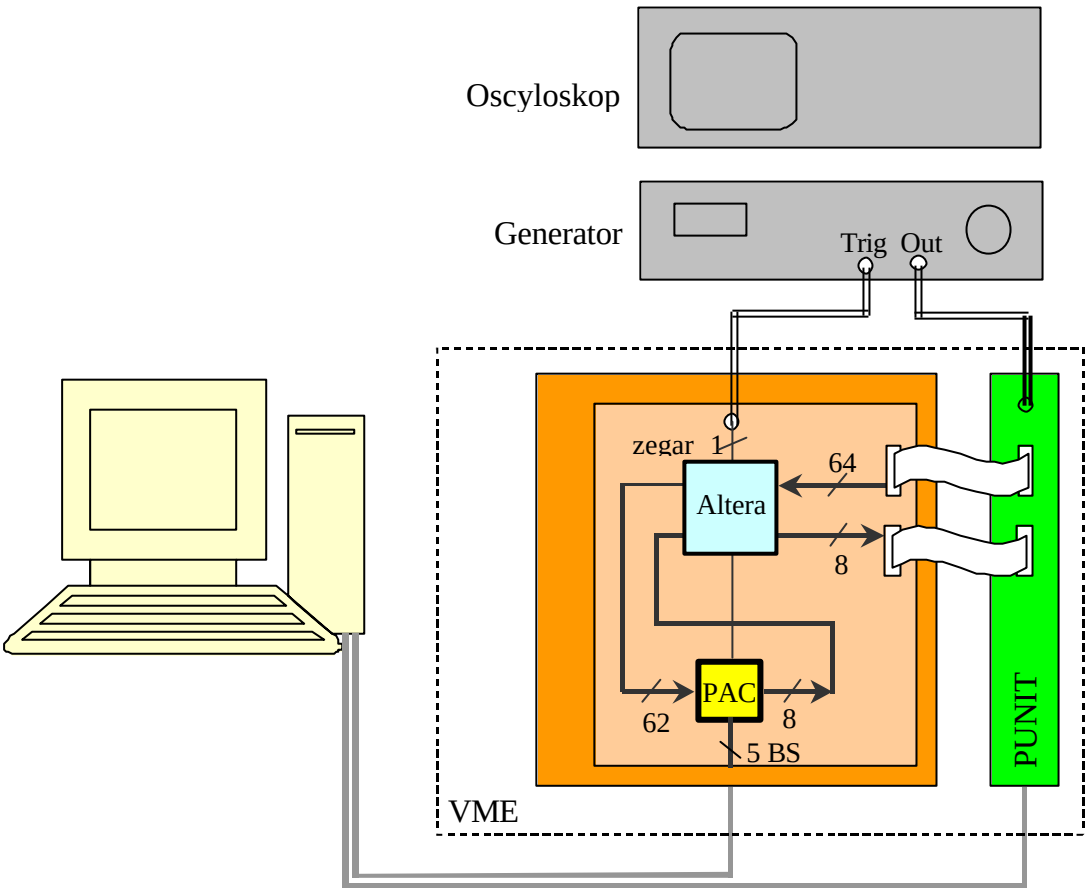
Rejestr masek:

- LUT Hpt – 16 czterobitowych liczb;
- LUT Hpt – 8 trzybitowych liczb;
- Maskowanie bitów wejsciowych – 124 bity;

- Znak – 4 bity;
- Flaga – 1 bit.

4 Uklad testowy

Schemat ukladu testowego przedstawia Rys. 4.1. PUNIT oraz plyta z kontrolerem sciezki brzegowej, na której umieszczona jest plytka testowa PACów znajduja sa w kasecie VME – standardowym urzadzeniu umozliwiajacym współprace ukladów elektronicznych z komputerem. Na plytce testowej znajduja sie procesor PAC oraz Altera ACEX (programowalny procesor FPGA). Zaprogramowania PACa sa ladowane z komputera bezposrednio przez plyte kontrolera BS, natomiast wymuszenia wysylane sa do PUNITa i poprzez Altere trafiaja do PACa, podobnie odpowiedzi PACa sa przesylane poprzez Altere do PUNITa.



Rys. 4.1: Schemat ukladu testowego.

4.1 PUNIT (Logic Patern Unit)

PUNIT jest blokiem VME skladajaca sie z osmiu chipów FIFO (kolejek First in First Out) o pojemnosci 4K 16-bitowych słów [8]. Kazdy z nich moze zostac zdefiniowany jako wejsciowy (IN) lub wyjsciowy (OUT). Do FIFO zdefiniowanych jako OUT mozna przy pomocy komputera wpisac sekwencje bitów. Po przekazaniu komendy START sekwencje te wysylane sa przez panel frontowy do podlaczonego urzadzenia (w tym przypadku plytki testowej), a jednoczesnie odpowiedzi urzadzenia sa sczytywane przez FIFO zdefiniowane

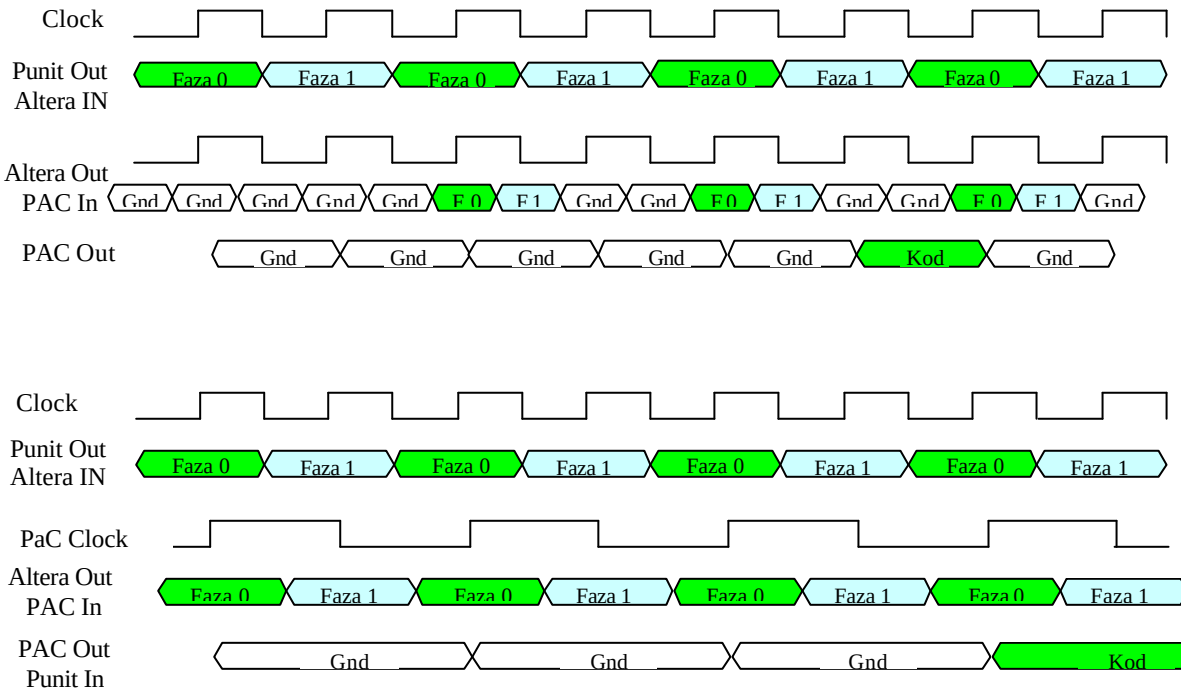
jako IN. Po zakonczeniu tej operacji bity z FIFO IN moga zostac odczytane przez komputer. W omawianym ukladzie testowym wykorzystywane zostaly cztery FIFO zdefiniowane jako wyjsciowe i jeden zdefiniowany jako wejsciowy.

Aby umozliwic synchronizacje PUNITa z podlaczonymi do niego urzadzeniami mozliwe jest opóznienie sygnalu danych wychodzacych i wchodzacych do PUNITa wzgledem sygnalu zegara w zakresie 0 – 32 ns.

4.2 Tryby pracy ukladu testowego

Jak zostalo wczesniej wspomniane, PAC oczekuje danych wejsciowych na opadajacym i rosnacym zboczu zegara, natomiast PUNIT moze je wysylac jedynie na zboczach rosnacych. Zadaniem Altery jest wiec odpowiednie przygotowanie dla Pac danych wejsciowych wysylanych przez PUNIT. Mozliwe sa dwa rozwiazania (Rys. 4.2):

- PAC pracuje z taka sama czestoscia jak PUNIT, jednak wlasciwe wymuszenia sa przekazywane przez Altere do PACa w co drugim takcie zegara, na przemian z wymuszeniami zerowymi,
- PAC pracuje z czestoscia o polowe mniejsza niz PUNIT (sygnal z generatora jest mnozony przez Altere), ale wymuszenia sa przekazywane w kazdym takcie zegara.



Rys. 4.2: Tryby pracy ukladu testowego. Tryb pierwszy (u góry) – wymuszenia sa wysylane do PACa w co drugim takcie zegara, tryb drugi (u dolu) – wymuszenia wysylane sa w kazdym takcie zegara, ale PAC pracuje z czestoscia dwa razy mniejsza niz PUNIT.

Tryb pierwszy wykorzystywany byl do testowania PACa przy czestosci 50 MHz (czestosc o 10 MHz wieksza, niz bedzie w eksperymencie). Poniewaz PUNIT moze pracowac stabilnie z maksymalna czestoscia ok. 60 MHz, dlatego nie mozna wykorzystac trybu

drugiego z czestoscia 100 MHz. Tryb drugi uzywany byl przy testach wstepnych z nizsza czestoscia.

4.3 Synchronizacja danych z sygnalem zegara

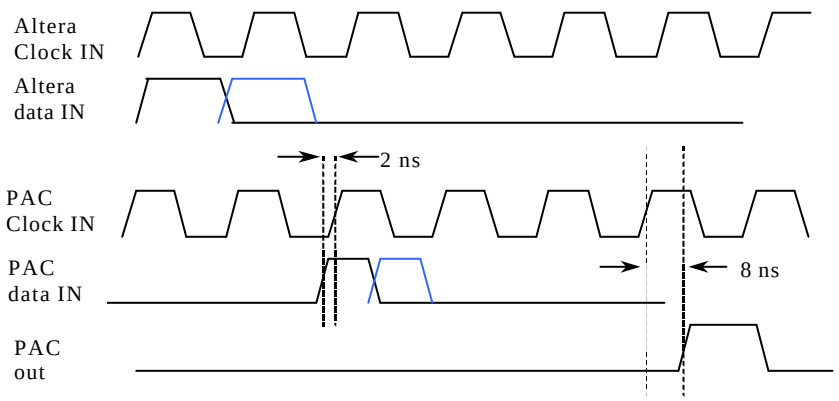
Dla poprawnego dzialania ukladu testowego niezwykle wazna jest synchronizacja sygnalu zegara z danymi wchodzacymi do Altery, PACa oraz PUNITa. Jej uzyskanie mozliwe jest poprzez ustawianie:

- opóznienia miedzy dwoma sygnalami generatora: Out – podawanym do plytki testowej i Tryg – podawanym do PUNITa,
- w PUNICie - opóznienia sygnalów wyjsciowych i wejsciowych,
- opózniania sygnalu zegara podawanego na PACa poprzez odpowiednie programowanie Altery.

Przy czestosci 50 MHz prawidlowe dzialanie ukladu uzyskano uzywajac nastepujacych ustawien:

- 8,1 ns – opóznienie w generatorze,
- 9 ns – wyjsciowe opóznienie PUNITa,
- 6 ns – wejsciowe opóznienie PUNITa,
- 5,7 ns – opóznienie sygnalu zegara przekazywanego do PACa przez Altere.

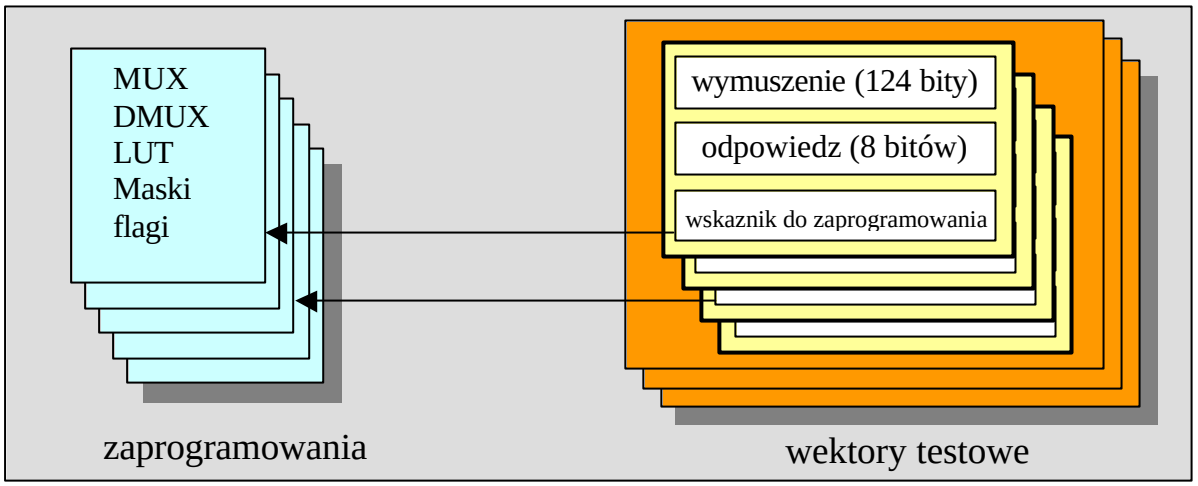
Zbadane przy pomocy oscyloskopu ustawienia sygnalów wzgledem zegara przedstawia Rys. 4.3.



Rys. 4.3: Usytlowanie syganlu zegara wzgledem sygnalów danych urzyte w testach z czestosia 50 MHz.

4.4 Baza danych

Testowe zaprogramowania PACa oraz zestawy wymuszen zgromadzone sa w komputerowej bazie danych typu SQL (Rys. 4.4). Podstawowym jej obiektem jest tzw. Vector Item, skladajacy sie z jednego wymuszenia, wskaznika do zaprogramowania oraz wlasciwej odpowiedzi PACa na to wymuszenie przy takim zaprogramowaniu. Vector Items zgrupowane sa w wektory testowe. Zaprogramowania zapisane sa w zrozumialym dla czlowieka formacie, na odpowiedni ciag bitów tlumaczone sa przez program testujacy.



Rys. 4.4: Schemat bazy danych, w której zapisane sa zaprogramowania i wektory testowe.

4.5 Program testujacy

Program obslugujacy uklad testowy zostal stworzony przy pomocy pakietu Borland C++ Builder. Umozliwia on edycje wektorów testowych i zaprogramowan (wpisywanie, kopiowanie, usuwanie) oraz ladowanie zaprogramowan, wysylanie pojedynczych wymuszen albo jednego lub calej serii wektorów testowych. Poniewaz jeden wektor testowy moze wymagac kilku róznych zaladowan, program sprawdza kazdy Vector Item i jesli wskazuje on na inne zaprogramowanie, niz aktualnie znajdujace sie w PACu, automatycznie go przeladowuje. Program porównuje odpowiedzi PACa z zapisanymi w bazie danych i informuje o ewentualnych bledach. Program umozliwia, poprzez interfejs sciezki brzegowej, odczytanie stanów nózek PACa w dowolnym momencie. Zawiera równiez symulator PACa, tzn. procedure obliczajaca odpowiedz PACa na zadane wymuszenie przy ustalonym zaprogramowaniu.

Poniewaz do przetestowania PACa potrzeba bylo duzej liczby zaprogramowan i wymuszen, stworzone zostaly procedury umozliwiajace ich automatyczna generacje.

5 Testy PACa

Procesor PAC jest ukladem bardzo skomplikowanym, zawiera duza liczbe róznych podukladów, z których czesc jest programowalna. Nie jest mozliwe zbadanie odpowiedzi PACa na kazde mozliwe wymuszenie przy dowolnym zaprogramowaniu – liczba mozliwych kombinacji jest astronomicznie wielka. Dlatego aby miec pewnosc, ze PAC funkcjonuje prawidlowo, szczególowo przetestowano dzialanie kazdego podukladu PACa.

Przeprowadzone testy PACa mozna podzielic na dwie zasadnicze kategorie: testy mechanizmu programowania, wykonywany głównie przy pomocy narzedzi ukladu sciezki brzegowej oraz testy sprawdzajace poprawnosc funkcjonowania logiki PAC, oparte na analizie odpowiedzi PACa na wymuszenia przy odpowiednim zaprogramowaniu.

5.1 Testy mechanizmu programowania

Pierwsze testy, jakim zostal poddany PAC, mialy na celu sprawdzenie dzialania ukladu sciezki brzegowej, przez który realizowane jest programowanie PACa. Sprawdzono, czy kontroler TAP poprawnie wykonuje instrukcje zapisane rejestrze instrukcji, czy wlasciwie wybiera kazdy rejestr. Zbadano tez, czy w kazdy z rejestrów da sie wpisac bity przez TDI i poprawnie je odczytac przez TDO. Poniewaz zauwazono pojawianie sie przeklaman bitów, do programu testujacego dodano funkcje umozliwiajaca wielokrotne wpisywanie i jednoczesne odczytywanie ciagu bitów w wybrany rejestr i automatyczne porównanie bitów odczytanych z wpisanymi. Pozwolilo to na dokladniejsza analize tych bledów i warunków ich powstawania.

Spróbowano tez uzyskac odpowiedzi PACa wymuszajac stany nózek wejsciowych przy pomocy rejestru brzegowego przy pewnym prostym zaprogramowniu.

5.2 Testy poprawnosci dzialania logiki PACa

Testy logiki PACa mialy za zadanie sprawdzenie dzialania kazdego podukladu procesora poprzez analize odpowiedzi na wysylane wymuszenia przy odpowiednim zaprogramowaniu. Aby ulatwic zlokalizowanie ewentualnych bledów, testy starano sie ukladac tak, aby wynik testu danego ukladu byl jak najmniej zalezny od bledów w pozostalych ukladach. PACa testowano „od tylu”, tzn. poczawszy od ukladów znajdujacych sie najblizej wyjscia, gdyz od nich zalezy kazda odpowiedz procesora. Ewentualne bledy w tych ukladach zafalszowalyby wyniki testów wszystkich wczesniejszych podukladów. Testy podukladów programowalnych mialy za zadanie sprawdzenie ich dzialania przy kazdym mozliwym zaprogramowaniu. Testy pozostalych podukladów powinny zbadac ich odpowiedzi na wszystkie mozliwe sygnaly na ich wejsciach. Zaprogramowania i wymuszenia powinny byc tak konstruowane, aby na podstawie analizy odpowiedzi mozna bylo stwierdzic nature i umiejscowienie ewentualnego bledu.

W PACu znajduje sie wielka liczba podukladów niektórych rodzajów, np. multiplekserów jest kilka tysiecy. Nie ma sensu tworzenie jednego zaprogramowania do przetestowania jednego takiego podukladu. Testy konstruowano wiec w ten sposób, aby jedno zaprogramowanie pozwalalo przetestowac wszystkie poduklady tego samego typu, natomiast wymuszenie powodowalo zadzialanie tylko jednego z tych podukladów.

Przy tworzeniu testów pomocne okazalo sie postawienie pytan, na które kazdy z testów powinien odpowiedziec:

- Test ukladu „Lpt if not Hpt & flag”.

  „Czy uklad dziala poprawnie przy obu wartosciach bitu flagi?”

- Test LUTów.

„Czy w kazda komórke LUT da sie wpisac kazda mozliwa wartosc? Czy jest wybierany wlasciwa komórka LUT?”

- Test ukladu „Bigger only”.

  „Czy sposród sladów pozytywnego i negatywnego wybierany jest slady z wyzszym kodem?”

- Test ukladów „Code of the biggest”.

  „Czy wybierany jest slad z najwyzszym kodem?”

- Test demultiplekserów (DMUX).

  „Czy kazdej grupie sladów da sie przypisac kazda mozliwa wartosc DMUX?”

- Test multiplekserów (MUX) i ukladu grupowania sladów (Track grouping).

  „Czy kazdy multiplekser dziala poprawnie po wpisaniu kazdej mozliwej wartosci? Czy kazdy wzorzec jest kierowany do wlasciwego demultipleksera?”

- Test ukladów definicji sladu („Track signal”).

  „Czy odpowiedz ma wlasciwa jakosc? Czy wybierany jest slad z wyzsza jakoscia?”

- Test ukladu dystrybucji bitów wejsciowych do bloków.

  „Czy z bity wejsciowych sa wlasciwie obliczane logiczne OR?”

- Test ukladu maskowania bitów wejsciowych.

  „Czy bity wejsciowe sa prawidlowo maskowane?”

Dokladny opis sposobu konstrukcji zaprogramowan i wymuszen dla poszczególnych testów znajduje sie w Dodatku A.

Testami tymi przebadano 20 procesorów PAC. Testy byly wykonywane automatycznie przy pomocy odpowiedniej funkcji programu testowego. Funkcja ta porównywala otrzymane odpowiedzi PACa z poprawnymi odpowiedziami zapisanymi w bazie danych i wyniki zapisywala do pliku. Jesli w odpowiedziach na którys z wektorów testowych pojawily sie bledy, PAC byl jeszcze raz ladowany tym samym zaprogramowaniem i dany test byl powtarzany. Dzieki temu mozna bylo stwierdzic charakter bledu: jesli po przeprogramowaniu nadal obserwowano takie same bledy, mozna przypuszczac, ze wynikaly one z wad konstrukcyjnych procesora. Natomiast, jesli po przeprogramowaniu obserwowano inne bledy lub nie obserwowano ich wcale, znaczylo to, ze byly one skutkiem przeklaman bitów programujacych.

Przetestowanie jednego procesora zajmowalo ok. pól godziny, najwiecej czasu pochlanialo odczytanie wektorów testowych i zaprogramowan z bazy danych.

Zanim przystapiono do wykonywania powyzszych testów sprawdzono, czy PAC reaguje na wszystkie bity wejsciowe. Po odpowiednim zaprogramowaniu PACa wysylano wymuszenia zawierajace po jednym bicie z kazdej plaszczyzny, tak aby przynajmniej jeden raz kazdy bit otrzymal wartosc „1”. Zauwazono brak odpowiedzi na kilka bitów. Po przebadaniu plytki testowej oscyloskopem stwierdzono, ze zostaly na niej niewlasciwie

wykonane dwie sciezki. Blad ten poprawiono. Pozostale bity prawidlowo dochodzily do nózek PACa, mimo to na niektóre z nich procesor nie reagowal. Przyczyna okazaly sie bledy w konstrukcji PACa.

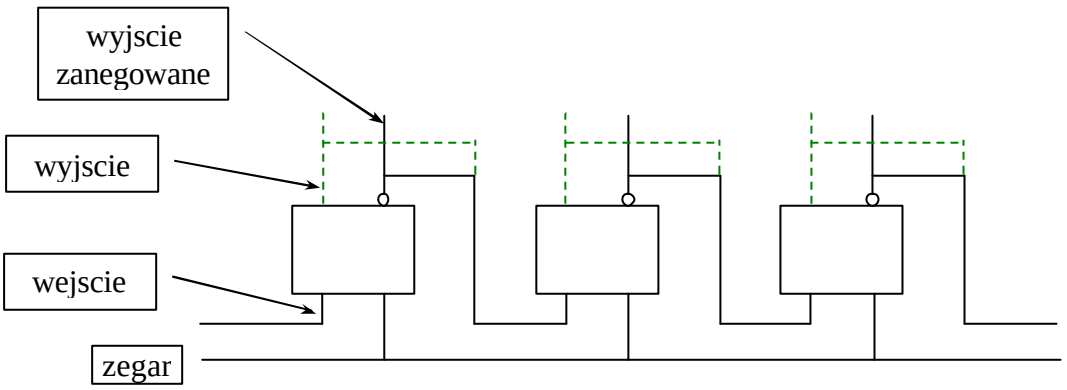
Po stwierdzeniu bledów konieczne bylo „reczne" dokladne ich zbadanie przy pomocy specjalnie konstruowanych zaprogramowan i wymuszen.

6 Stwierdzone bledy

Stwierdzone bledy wystepuja we wszystkich dwudziestu przebadanych procesorach i wynikaja z bledów w projekcie ukladu scalonego. Nie zauwazono zadnych wad wystepujacych tylko w pojedynczych PACach. Wszystkie procesory dzialaja jednakowo stabilnie przy czestosci 50 MHz. Zauwazono natomiast róznice miedzy poszczególnymi PACami w czestosci pojawiania sie przeklaman bitów rejestru masek przy programowaniu z wlaczonym zegarem (pp. 6.2). W pieciu PACach przeklamania takie pojawialy sie w ok. 30% zaprogramowan, podczas gdy w innych pieciu zle zaprogramowania stanowily mniej niz 5%. W przypadku pozostalych dziesieciu procesorów przeklamania wystepowaly w kilkunastu procentach zaprogramowan.

6.1 Zle polaczone przerzutniki w rejestrze masek

Przy pierwszych próbach wpisywanie bitów w rejestr masek stwierdzono blad w jego konstrukcji. Rejestry ukladu sciezki brzegowej zbudowane sa z szeregowo polaczonych przerzutników (Rys. 6.1).



Rys. 6.1: Schemat polaczenie przerzutników sciezki brzegowej. Liniami przerywanymi oznaczone sa prawidlowe polaczenia, liniami ciaglymi – wykonane w rejestrze masek.

Bity podawane na poczatku lancucha sa kolejno po nim przesuwane. Po wpisaniu calego ciagu bitów przewidzianego dla danego rejestru, na wyjsciach przerzutników powinny pojawic sie bity tworzace zaprogramowanie. Kazdy przerzutnik ma dwa wyjscia: zwykle i zanegowane. Standardowo wejscie jednego przerzutnika laczy sie ze zwyklym wyjsciem poprzedniego. W PACu jednak, w czesci lancucha rejestru masek, wejscia zostaly podlaczone do zanegowanego wyjscia. Przez to, po wpisaniu calego ciagu, co drugi bit jest negowany parzysta liczbe razy, w wyniku czego jego koncowa wartosc jest taka sama, jak poczatkowa, a co drugi jest negowany nieparzysta liczbe razy i jego koncowa wartosc jest negacja wartosci poczatkowej. Tak wiec aby na wyjsciach przerzutników pojawil sie ciag np. 11110000 trzeba

wpisac: 01011010. Zostalo to uwzgledniony w programie testujacym, w procedurze przekladajacej zaprogramowanie zapisane w bazie na ciag bitów wysylanych do PACa.

6.2 Przeklamania bitów wpisywanych w rejestr masek

Wpisujac i odczytujac bity z rejestru masek zauwazono pojawianie sie przeklaman, tzn. w odczytanym ciagu kilka bitów zmienionych bylo z „1” na „0” lub odwrotnie. Przeklamania te mialy przypadkowy charakter, tzn. na wszystkich pozycjach w ciagu pojawialy sie z podobna czestoscia. Przy przeprowadzaniu testów z uzyciem wymuszen równiez zauwazono problemy z programowaniem rejestru masek: zamaskowywanie niektóre bitów wejsciowych lub zmiane wartosci kodów wpisanych w LUTy. Sugerowalo to, ze te przypadkowe przeklamania pojawiaja sie juz przy wpisywaniu bitów do rejestru, a nie przy ich odczytywaniu. Po szczególowych badaniach, polegajacych na wielokrotnym wpisywaniu i odczytywaniu ciagu bitów w rejestr masek, okazalo sie, ze przeklamania pojawiaja sie wtedy, gdy podczas programowania PACa podawany jest na niego sygnal zegara.

6.3 Wplyw wysylania wymuszen na dzialanie ukladu sciezki brzegowej

Zauwazono równiez, ze na dzialanie ukladu sciezki brzegowej ma wplyw to, czy na PACa podawane sa wymuszenia. Na nózki wejsciowe PACa, który byl wczesniej zaprogramowany pustym zaprogramowaniem (tzn. skladajacym sie z samych zer), wysylane byly z Altery wymuszenia skladajaca sie z samych jedynek. Bity wchodzace przez TDI do ukladu sciezki brzegowej przesylane byly przez rejestr BYPASS, dzieki czemu omijaly pozostale rejestry i nie zmienialy zaprogramowania. Stwierdzono, ze ok. 1/3 ciagów bitów odczytanych z TDO zawierala bledy. Zmniejszenie liczby jedynek wysylanych na nózki wejsciowe powodowalo zmniejszenie czestosci wystepowania bledów.

Natomiast, jesli wczesniej wpisano w PACa zaprogramowanie zawierajace przynajmniej jeden wzorzec sladu, to, jesli na nózki wejsciowe wysylane byly same jedynki, uklad sciezki brzegowej calkowicie sie blokowal, tzn. nózka TDO ciagle pozostawala w stanie wysokim, niezaleznie od tego, co bylo wysylane na TDI.

Takie zablokowanie pojawialo sie równiez przy próbie przesylania bitów przez rejestry bloków lub masek, niezaleznie od tego, jakie zaprogramowanie bylo wczesniej wpisane.

	Na nózki wejsciowe wysylane same zera	Na nózki wejsciowe wysylane same jedynki
PAC zaprogramowany pustym zaprogramowaniem	*Sporadyczne przeklamania bitów*	*Bledy pojawiaja sie w ok. 1/3 ciagów*
PAC zaprogramowany zaprogramowaniem zawierajacym przynajmniej jeden wzorzec	*Sporadyczne przeklamania bitów*	*Zablokowanie ukladu sciezki brzegowej, nózka TDO pozostaje stale w wysokim stanie*

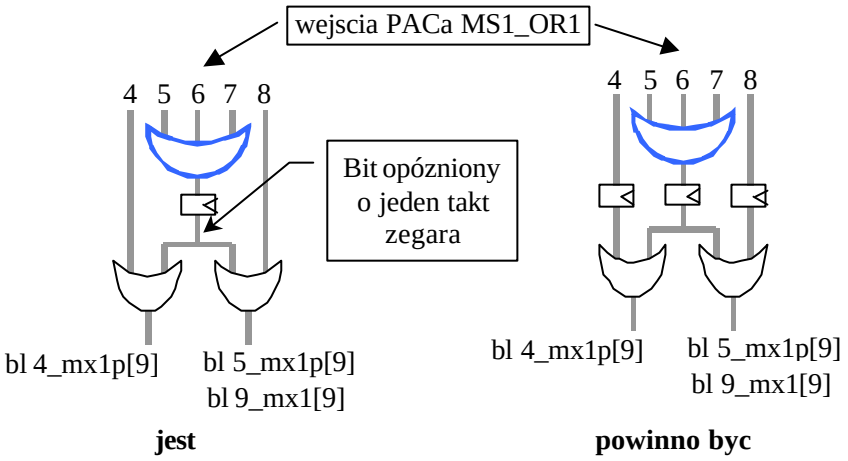
Tabela 1: Wyniki przesylania bitów w trybie BYPASS w zaleznosci od zaprogramowania PACa i wysylanych wymuszen.

Nalezy zauwazyc, ze wysylanie sygnalu zegara ma wplyw jedynie na programowanie rejestru masek, podczas gdy wysylanie jedynek na wejsciowe nózki PACa powoduje bledy w dzialaniu calego ukladu sciezki brzegowej.

Aby podczas testów logiki PACa ominac te problemy, program testujacy zmodyfikowano w ten sposób, ze przed rozpoczeciem programowania z PUNITa wysylany jest jeden bit o wartosci „1” (pozostale maja wartosc „0”), po otrzymaniu którego Altera blokuje przekazywanie sygnalu zegara do PACa W momencie wysylania wymuszen wartosc tego bitu ustawiana jest na „0”,co jest sygnalem dla Altery do wznowienia przekazywania sygnalu zegara.

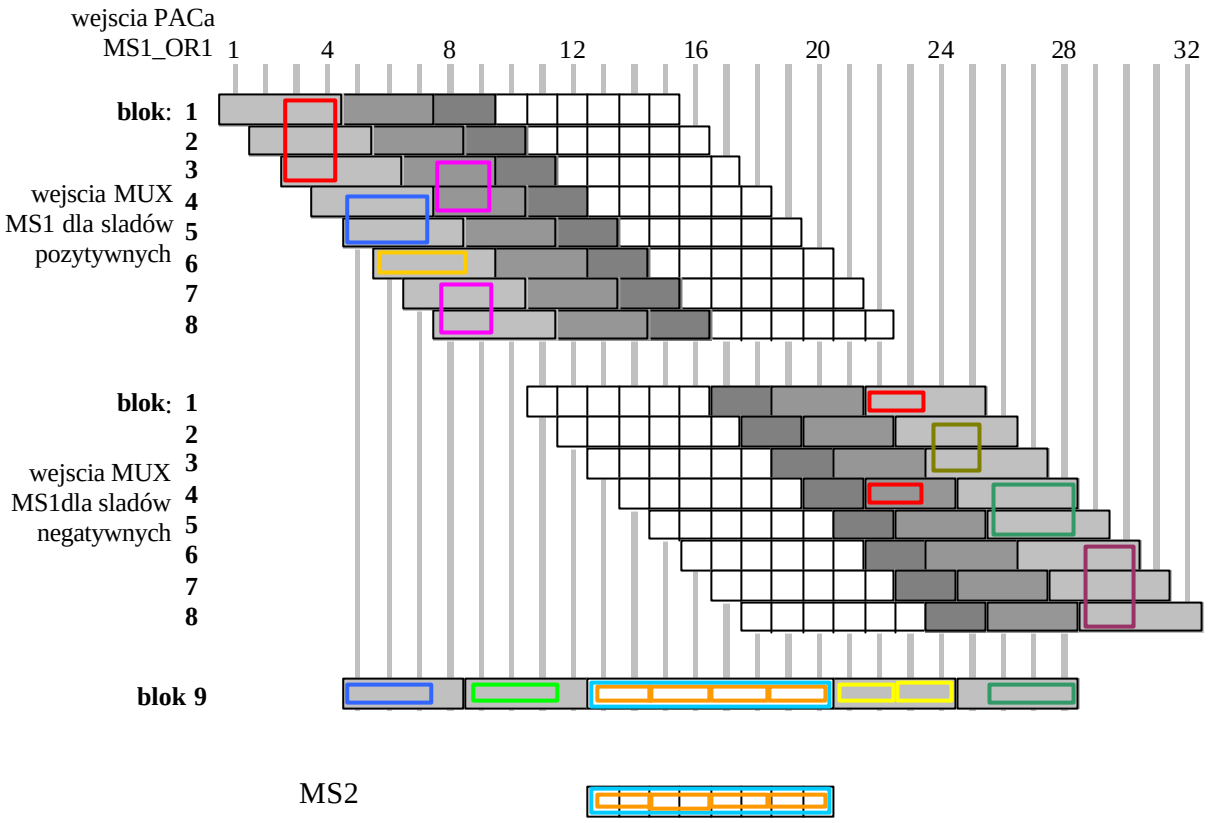
6.4 Nieprawidlowe obliczanie wejsc niektórych MUXów

Jak wczesniej zostalo wspomniane, bity wejsciowe nie sa bezposrednio przesylane na wejscia MUXów, lecz sa odpowiednio rozdzielane do bloków, a z niektórych sa obliczane logiczne OR. Dla kazdego bloku inne bity wymagaja obliczenia z nich OR, jednak aby zmniejszyc zlozonosc ukladu (liczbe bramek wykonujacych te operacje), czesc obliczen wykonuja wspólne dla kilku bloków bramki.



Rys. 6.2: Sposób obliczania wejsc MUXów powodujacy, ze niektóre bity sa opóznione o jeden takt zegara.

I tak na przyklad bity MS1_OR1 nr 4, 5, 6, 7 po zsumowaniu sa dziewiatym wejsciem MUXów bloku 4, natomiast bity MS1_OR1 nr 5, 6, 7, 8 sa dziewiatym wejsciem MUXów bloku 5 oraz bloku 9. Wspólne bity 5, 6, 7 sumowane sa przez jedna bramke, której wyjscie podlaczone jest do dwóch bramek sumujacych pozostale dwa bity 4 i 8 (Rys. 6.2). Niestety jednak, nie zostaly one podlaczone bezposrednio, lecz przez przerzutnik, opózniajacy sygnal o jeden takt zegara. W podobny sposób, tzn. przez dwa stopnie bramek, obliczanych jest kilka innych bitów MS1_OR1 (Rys. 6.3), przez co sygnaly im odpowiadajace pojawiaja sie na wejsciach MUXów o jeden takt zegara za pózno, w momencie, gdy analizowane sa juz bity z nastepnego wymuszenia. Natomiast w bloku 9, wejscie nr 7 MUXa MS1 oraz OR bitów MS2 obliczane sa przez trzy stopnie bramek, w zwiazku z tym opóznione sa o dwa takty zegara.



Rys. 6.3: Schemat orowania bitów MS1_OR1 i MS2. Bity zaznaczone kolorowymi prostokatami docieraja do wejsc MUXów o jeden takt zegara za pózno. Bity zaznaczone dwoma prostokatami opóznione sa o dwa takty.

6.5 Blad w projekcie MUXów mx1 bloku dziewiatego

PAC nie reaguje na bity MS1_OR4[1…4] (wykorzystywane sa one jedynie przez blok 9) chociaz na pewno dochodza one do nózek (ich prawidlowa wartosc mozna sczytac przy pomocy Rejestru Brzegowego) (Rys. 3.3). Przyczyna jest blad w projekcie MUXów mx1 bloku dziewiatego: ustawienie w programowalnym rejestrze drugiego bitu na 0 powoduje zwarcie do GND galezi prowadzacych do wejsc nr 12 i 13, do których trafiaja sygnaly z bitów MS1_OR4[1…4].

7 Podsumowanie

Stwierdzone bledy wykluczaja uzycie testowanej wersji PACa w eksperymencie. Konieczne bedzie poprawienie projektu i wyprodukowanie nowej wersji. Bedzie ona prawdopodobnie wykonana w nowej technologii 25 µm. Rozwazane sa równiez modyfikacje algorytmu, tak aby PAC analizowal koincydencje ze wszystkich szesciu warstw komór (zastosowanie nowej technologii pozwoli na zmieszczenie takiego bardziej zlozonego ukladu).

Trwaja równiez badania alternatywnych mozliwosci zaimplementowania algorytmu PAC w programowalne procesory FPGA.

Dodatek A. Sposób konstruowania testów poprawnosci dzialania logiki PACa

1. Testy ukladu „Lpt if not Hpt & flag”

Test sprawdza odpowiedz na kazdy mozliwy stan wejsciowy ukladu przy obu wartosciach programowalnego bitu flagi.

2. Testy LUTów

„Czy w kazda komórke LUT da sie wpisac kazda mozliwa wartosc? Czy jest wybierany wlasciwa komórka LUT?”

Zaprogramowania stworzono w ten sposób, aby w kolejne komórki LUTów Hpt wpisywane byly kolejne liczby:

zaprogramowanie „Test LUT 1”: LUT[0]=0, LUT[1]=1, LUT[2]=2,...,LUT[15]=15;

W kolejnych zaprogramowaniach ciag liczb jest przesuwany :

„Test LUT 2”: LUT[0]=0, LUT[1]=2, LUT[2]=3,...,LUT[15]=16;

:

:

„Test LUT 31”: LUT[0]=0, LUT[1]=31, LUT[2]=0,...,LUT[15]=13;

„Test LUT 32”: LUT[0]=0, LUT[1]=0, LUT[2]=1,...,LUT[15]=14;

(w LUT[0] wpisane bylo zawsze 0).

W kazdym zaprogramowaniu wpisano tez 16 wzorców, DMUXy zaprogramowano kolejnymi liczbami tak, aby wzorce te wskazywaly na wszystkie komórki LUTów. Wymuszeni odpowiadaly wpisanym wzorcom.

W podobny sposób przebadano LUT Lpt. Poniewaz sklada sie on jedynie z 8 czterobitowych komórek, potrzeba do tego 16 zaprogramowan.

3. Testy ukladu „Bigger only”

„Czy sposród sladów pozytywnego i negatywnego wybierany jest slady z wyzszym kodem?”

Komórki LUT Hpt zaprogramowano kolejnymi liczbami od 0 do 15. W jeden z bloków wpisano po 15 wzorców pozytywnych i negatywnych, odpowiadajace im DMUXy zaprogramowano kolejnymi liczbami 1-15. Kazde wymuszenie zawieralo bity odpowiadajace dwóm wzorcom – jednemu pozytywnemu i jednemu negatywnemu. Wymuszenia zostaly zrobione tak, aby kazdy wzorzec negatywny (a wiec kazdy mozliwy kod negatywny) zostal porównany z kazdym wzorcem pozytywnym.

Podobnie sprawdzono uklad Lpt.

4. Testy ukladów „Code of the biggest”

„Czy wybierany jest slad z najwyzszym kodem?”

Kazdy z czterech ukladów przetestowano podobnie jak uklad „Bigger only”, z tym, ze wymuszenia zawieraly bity odpowiadajace dwóm wzorcom tego samego znaku. Dodatkowo stworzono kilka wymuszen, w których porównywanych bylo wiecej niz dwa slady

5. Test demultiplekserów (DMUX)

„Czy kazdej grupie sladów da sie przypisac kazda mozliwa wartosc DMUX?”

W PACu znajduje sie 9 × 62 DMUXów. Aby odpowiedziec na powyzsze pytanie nalezy sprawdzic, czy w kazdy z nich mozna wpisac kazda mozliwa wartosc, tzn. liczbe z zakresu 0 – 15 (bloki 1-8) lub 0 – 7 (blok 9). Nalezy tez skonstruowac takie wzorce, aby wymuszenia im odpowiadajace pasowaly tylko do jednego wzorca, prowadzacego do demultipleksera, który ma byc przez to wymuszenie testowany.

Zaprogramowania stworzono w nastepujacy sposób: w kazdym zaprogramowaniu w LUTy wpisano kolejne liczby. W kazdy blok wpisano nastepujace wzorce i wartosci DMUX:

Zaprogramowanie nr 1:

Wzorce pozytywne, bloki 1-8.

nr wzorca	**16**	**24**	**...**	**44**	**48**	**50**	**52**	**54**	**56**	**58**	**60**	**62**	**64**	**65**	**66**	**...**	**71**	**72**	**73**	**74**	**...**	**80**
mx1	2	3		7	8	6	7	8	4	5	6	7	8	1	2		7	8	1	2		8
mx3	3	4		8	9	5	6	7	4	5	6	7	8	1	1		1	1	3	4		10
mx4	1	2		6	7	1	2	3	1	2	3	4	5	1	2		7	8	2	3		9
Nr DMUX	1	2		6	7	8	9	10	11	12	13	14	15	16	17		22	23	24	25		31
Wartosc DMUX	15	14		10	9	8	7	6	5	4	3	2	1	1	15		10	9	8	7		1

Wzorce negatywne, bloki 1- 8.

Nr wzorca	**16**	**24**	**...**	**44**	**48**	**50**	**52**	**54**	**56**	**58**	**60**	**62**	**64**	**65**	**66**	**...**	**71**	**72**	**73**	**74**	**...**	**80**
mx1n	8	7		3	2	4	3	2	6	5	4	3	2	9	8		3	2	9	8		2
mx3	9	8		4	3	7	6	5	8	7	6	5	4	11	11		11	11	9	8		2
mx4	11	10		6	5	11	10	9	11	10	9	8	7	11	10		5	4	10	9		3
Nr DMUX	1	2		6	7	8	9	10	11	12	13	14	15	16	17		22	23	24	25		31
Wartosc DMUX	15	14		10	9	8	7	6	5	4	3	2	1	1	15		10	9	8	7		1

Blok 9, wzorce pozytywne.

Nr wzorca	**16**	**24**	**...**	**50**	**52**	**54**	**56**	**58**	**...**	**67**	**68**	**69**	**70**	**71**	**72**	**...**	**78**	**79**	**80**
mx1p	11	11		11	11	11	11	11		11	11	11	11	11	11		11	11	11
mx3	7	7		7	7	8	8	8		8	8	8	9	9	9		9	9	9
mx4	3	4		10	11	1	2	3		9	10	11	1	2	3		9	10	11
Nr DMUX	1	2		8	9	10	11	12		18	19	20	21	22	23		29	30	31
Wartosc DMUX	1	7		1	7	6	5	4		7	6	5	4	3	2		3	2	1

Blok 9, wzorce negatywne.

Nr wzorca	16	24	...	50	52	54	56	58	...	67	68	69	70	71	72	...	78	79	80
mx1n	1	1		1	1	1	1	1		1	1	1	1	1	1		1	1	1
mx3	3	3		3	3	2	2	2		2	2	2	1	1	1		1	1	1
mx4	9	8		2	1	11	10	9		3	2	1	11	10	9		3	2	1
Nr DMUX	1	2		8	9	10	11	12		18	19	20	21	22	23		29	30	31
Wartosc DMUX	1	7		1	7	6	5	4		7	6	5	4	3	2		3	2	1

Taka konstrukcja umozliwia przetestowanie jednym zaprogramowaniem wszystkich demultiplekserów. Poniewaz w demultipleksery Hpt mozna wpisac liczby od 0 do 15, potrzeba 15 takich zaprogramoawan. W kolejnych zaprogramowaniach zwiekszano o jeden (modulo 15 w blokach 1 – 8, modulo 7 w bloku 9) wartosc liczby wpisanej w kazdego DMUXa, natomiast wzorce pozostawiono bez zmian. Dla kazdego zaprogramowania wygenerowano wymuszenia odpowiadajace kolejnym wzorcom.

6. Test multiplekserów i ukladu grupowania sladów (Track grouping)

„Czy kazdy multiplekser dziala poprawnie po wpisaniu kazdej mozliwej wartosci? Czy kazdy wzorzec jest kierowany do wlasciwego demultipleksera?”

W PACu jest $9 \times 3 \times 160$ multiplekserów. Udzielenie odpowiedzi na te pytania wymaga wpisania w kazdy z nich kazdej mozliwej wartosci. Aby mozna bylo sprawdzic, czy slady sa poprawnie grupowane i kierowane do wlasciwych demultiplekserów, kazdy z demultiplekserów zaprogramowano inna wartoscia, a w LUTy wpisano kolejne liczby. W kazdy z bloków wpisano nastepujace wzorce:

Zaprogramowanie nr 1.

Wzorce pozytywne, bloki 1 - 8

Nr wzorca	1	2	…	9	10	11	12	13	…	19	20	21	22	…	67	68	…	75	76	77	78	79	80
Msp1p	1	2		9	1	1	2	3		9	1	2	2		7	8		6	7	7	8	9	1
Msp3	2	3		10	11	1	2	3		9	10	11	1		2	3		10	11	1	2	3	4
Msp4	1	2		9	10	11	1	2		8	9	10	11		1	2		9	10	11	1	2	3
Wartosc DMUX	1	1		1	1	1	1	1		2	2	2	2		3	4		11	12	13	14	15	1

Wzorce negatywne bloki 1 - 8

Nr wzorca	1	2	…	9	10	11	12	13	…	19	20	21	22	…	67	68	…	75	76	77	78	79	80
Msp1n	1	2		9	1	1	2	3		9	1	2	2		7	8		6	7	7	8	9	1
Msp3	1	2		9	10	11	1	2		8	9	10	11		1	2		9	10	11	1	2	3
Msp4	2	3		10	11	1	2	3		9	10	11	1		2	3		10	11	1	2	3	4
Wartosc Dmux	1	1		1	1	1	1	1		2	2	2	2		3	4		11	12	13	14	15	1

Wzorce pozytywne, blok 9

Nr wzorca	1	2	…	9	10	11	12	13	14	15	...	25	26	…	66	67	…	75	76	77	78	79	80
Msp1p	1	2		9	10	11	12	13	1	2		12	13		1	2		10	11	12	13	1	2
Msp3	1	2		9	1	2	3	4	2	3		4	5		6	7		6	7	8	9	7	8
Msp4	1	2		9	10	11	1	2	1	2		1	2		1	2		10	11	1	2	2	3
Wartosc DMUX	1	1		1	1	1	1	1	1	1		3	3		3	4		5	6	7	1	2	3

Wzorce negatywne, blok 9

Nr wzorca	**1**	**2**	**…**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**...**	**25**	**26**	**…**	**66**	**67**	**…**	**75**	**76**	**77**	**78**	**79**	**80**
Msp1p	1	2		9	10	11	12	13	1	2		12	13		1	2		10	11	12	13	1	2
Msp3	1	2		9	1	2	3	4	2	3		4	5		6	7		6	7	8	9	7	8
Msp4	2	3			11	1	2	3	2	3		2	3		2	3		11	1	2	3	3	4
Wartosc DMUX	1	1		1	1	1	1	1	1	1		3	3		3	4		5	6	7	1	2	3

W ten sposób mozna jednym zaprogramowaniem przetestowac wszystkie multipleksery. Aby w kazdy MUX przynajmniej raz byla wpisana kazda mozliwa wartosc, w kolejnych zaprogramowniach schemat wzorców w kazdym bloku nalezy przesunac o jedna kolumne w prawo (wzorzec 77 przechodzi we wzorzec pierwszy) (wzorce 78, 79, 80 bloków 1 – 8 i 79, 80 bloku 9 wymagaja odrebnego traktowania). Wartosci DMUXów pozostaja bez zmian. W ten sposób skonstruowano 13 zaprogramowan.

7. Testy ukladów definicji sladu („Track signal”)

„Czy odpowiedz ma wlasciwa jakosc? Czy wybierany jest slad z wyzsza jakoscia?”

Przetestowanie kazdego z 9×160 ukladów definicji sladu wymaga zaprogramowania wszystkich multiplekserów. Wzorce musza byc jednak takie, aby kazdy slad o dowolnej jakosci byl jednoznacznie zdefiniowany, tzn. pasowal tylko do jednego wzorca. Poniewaz takich jednoznacznych wzorów nie da sie zrobic zbyt wielu, dlatego w jednym zaprogramowaniu zapisywano jedynie wzorce jednego rodzaju (pozytywne lub negatywne) jednego bloku. Stworzono 16 zaprogramowan testujacych bloki 1-8, w kazdym z nich w jeden blok wpisano nastepujace wzorce:

Wzorce wpisywane w bloki 1-8 blok 8 (1)

Nr wzorca	**1**	**2**	**…**	**9**	**10**	**11**	**12**	**13**	**…**	**20**	**21**	**22**	**…**	**67**	**68**	**…**	**75**	**76**	**77**	**78**	**79**	**80**	**80**
Msp1p	1	1		1	1	1	2	2		2	2	2		7	7		7	7	7	8	8	8	1
Msp3	1	2		9	10	11	1	2		9	10	11		1	2		9	10	11	1	2	3	1
Msp4	1	2		9	10	11	2	3		10	11	1		7	8		4	5	6	8	9	10	1
Wartosc DMUX	2	2		2	2	2	2	2		2	2	2		2	2		2	2	2	2	2	2	1

Dodatkowy wzorzec w bloku 8 (dla testów bloków 1-4) lub w bloku 1 (dla testów bloków 5-8) sluzyl do porównywania z nim sladów o róznej jakosci. Przetestowanie bloku 9 wymagalo stworzenia 4 dodatkowych zaprogramowan o nieco innym schemacie.

Dla kazdego wzorca zrobiono po cztery wymuszenia, z których kazde zawieralo slad o innej jakosci. Stworzono tez wymuszenia zawierajace jeden slad odpowiadajacym testowanemu wzorcowi i jeden odpowiadajacy dodatkowemu wzorcowi z bloku 8 (lub bloku 1). Dla kazdego wzorca stworzono 20 takich wymuszen, tak, aby przetestowac wszystkie kombinacje jakosci tych dwu sladów.

Dodatkowo zrobiono tez wymuszenia zawierajace po jednym bicie z tylko dwóch plaszczyzn, odpowiedz na nie powinna skladac sie z samych zer.

8. Test ukladu dystrybucji bitów wejsciowych do bloków.

„Czy z bity wejsciowych sa wlasciwie obliczane logiczne OR?”

W kazdym bloku wpisano kilka wzorców takich, aby multipleksery wskazywaly na wejscia, które sa wynikiem sumowani kilku bitów wymuszenia.

9. Test ukladu maskowania bitów wejsciowych.

„Czy bity wejsciowe sa prawidlowo maskowane?”

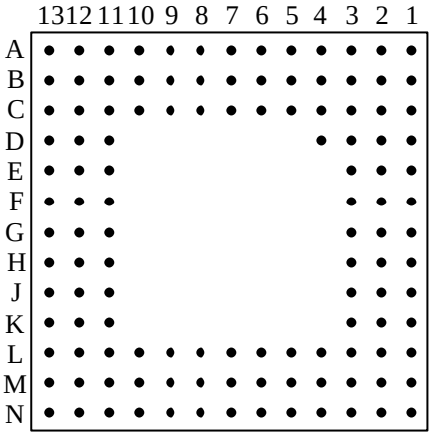
Stworzono cztery zaprogramowania, w kazdym zamaskowano wszystkie bity z jednej plaszczyzny. Wpisano tez takie wzorce, aby mozna bylo sprawdzic kazdy bit wejsciowy.

Dodatek B. Nózki PACa

	1	2	3
1	C3	ms1p_or4*<5>	x_in<27>
2	B2	vdd	
3	B1	gnd	
4	D3	ms1p_or4*<6>	x_in<28>
5	C2	ms2p_or4*<1>	x_in<33
6	C1	ms2p_or4*<2>	x_in<34
7	D2	ms2p_or4*<3>	x_in<35
8	E3	ms2p_or4*<4>	x_in<36
9	D1	ms2p_or4*<5>	x_in<37
10	E2	ms1_or4*<1>	x_in<16>
11	E1	gnd	
12	F3	vdd	
13	F2	gnd	
14	F1	vdd	
15	G2	ms1_or4*<2>	x_in<17>
16	G3	ms1_or4*<3>	x_in<18>
17	G1	ms1_or4*<4>	x_in<19>
18	H1	ms1_or4*<5>	x_in<20>
19	H2	ms1_or4*<6>	x_in<21>
20	H3	ms1_or4*<7>	x_in<22
21	J1	tdi	(JTAG)
22	J2	vdd	
23	K1	gnd	
24	J3	tms	(JTAG)
25	K2	tdo	(JTAG)
26	L1	tck	(JTAG)
27	M1	trst	(JTAG)
28	K3	gnd	
29	L2	vdd	
30	N1	empty	
31	L3	code<7>	code<7>
32	M2	code<6>	code<6>
33	N2	code<5>	code<5>
34	L4	code<4>	code<4>
35	M3	vdd	
36	N3	gnd	
37	M4	code<3>	code<3>
38	L5	code<2>	code<2>
39	N4	code<1>	code<1>
40	M5	code<0>	code<0>
41	N5	ms2_or*<4>	x_in<29>
42	L6	vdd	
43	M6	gnd	
44	N6	ms2_or*<3>	x_in<31>
45	M7	ms4_or1*<12>	x_in<61>
46	L7	ms4_or1*<11>	x_in<60>
47	N7	ms4_or1*<10>	x_in<59>
48	N8	ms4_or1*<9>	x_in<58>
49	M8	vdd	
50	L8	gnd	
51	N9	ms4_or1*<8>	x_in<57>
52	M9	ms4_or1*<7>	x_in<56>
53	N10	ms4_or1*<6>	x_in<55>
54	L9	ms4_or1*<5>	x_in<54>
55	M10	ms4_or1*<4>	x_in<53>
56	N11	vdd	
57	N12	gnd	
58	L10	ms4_or1*<3>	x_in<52>
59	M11	ms4_or1*<2>	x_in<51>
60	N13	ms4_or1*<1>	x_in<50>

	1	2	3
61	L11	empty	
62	M12	empty	
63	M13	empty	
64	K11	vdd	
65	L12	gnd	
66	L13	vdd	
67	K12	gnd	
68	J11	ms3_or1*<1>	x_in<38>
69	K13	ms3_or1*<2>	x_in<39>
70	J12	ms3_or1*<3>	x_in<40>
71	J13	gnd	
72	H11	vdd	
73	H12	ms3_or1*<4>	x_in<41>
74	H13	ms3_or1*<5>	x_in<42>
75	G12	clock	
76	G11	ms3_or1*<6>	x_in<43>
77	G13	ms3_or1*<7>	x_in<44>
78	F13	vdd	
79	F12	gnd	
80	F11	vdd	
81	E13	gnd	
82	E12	ms3_or1*<8>	x_in<45>
83	D13	ms3_or1*<9>	x_in<46>
84	E11	ms3_or1*<10>	x_in<47>
85	D12	ms3_or1*<11>	x_in<48>
86	C13	ms3_or1*<12>	x_in<49>
87	B13	gnd	
88	D11	vdd	
89	C12	empty	
90	A13	empty	
91	C11	ms1_or1*<1>	x_in<0>
92	B12	ms1_or1*<2>	x_in<1>
93	A12	ms1_or1*<3>	x_in<2>
94	C10	ms1_or1*<4>	x_in<3>
95	B11	ms1_or1*<5>	x_in<4>
96	A11	gnd	
97	B10	vdd	
98	C9	ms1_or1*<6>	x_in<5>
99	A10	ms1_or1*<7>	x_in<6>
100	B9	ms1_or1*<8>	x_in<7>
101	A9	ms1_or1*<9>	x_in<8>
102	C8	ms1_or1*<10>	x_in<9>
103	B8	gnd	
104	A8	vdd	
105	B7	ms1_or1*<11>	x_in<10>
106	C7	ms1_or1*<12>	x_in<11>
107	A7	ms1_or1*<13>	x_in<12>
108	A6	ms1_or1*<14>	x_in<13>
109	B6	ms1_or1*<15>	x_in<14>
110	C6	gnd	
111	A5	vdd	
112	B5	ms1_or1*<16>	x_in<15>
113	A4	ms2_or*<1>	x_in<29>
114	C5	ms2_or*<2>	x_in<29>
115	B4	ms1p_or4*<1>	x_in<23
116	A3	ms1p_or4*<2>	x_in<24>
117	A2	gnd	
118	C4	vdd	
119	B3	ms1p_or4*<3>	x_in<25>
120	A1	ms1p_or4*<4>	x_in<26>

Tabela zawiera opis funkcji nózek PACa. W kolumnie 1 znajduja sie nazwy nózek PACa (wedlug rysunku ponizej), w kolumnie 2 opis ich funkcji, w kolumnie 3 nazwa nózki Alatery, z która dana nózka PACa sie laczy.



Rys B.1: Uklad i oznaczenie nózek PACa.

PODZIEKOWANIA

Chcialbym serdecznie podziekowac wszystkim, którzy przyczynili sie do powstania niniejszej pracy. W szczególnosci Panu profesorowi dr hab. Janowi Króliowskiemu za skierowanie mnie na ten temat i wspomaganie mnie swoimi cennymi radami i wskazówkami.

Dziekuje tez bardzo Panu Maciejowi Kudle za nieoceniona pomoc, bez której powstanie tej pracy byloby niemozliwe oraz Michalowi Pietrusinskiemu za jego wklad w stworzenie oprogramowania i pomoc przy testach.

Dziekuje równiez Panu Krzysztofowi Kierzkowskiemu oraz Michalowi Wieji, który stworzyl symulator PACa.

Bibliografia

[1] The Muon Project Technical Design Report, **CERN/LHCC 97-32, 1997.**

[2] CMS The Trigger and Data Acquisition project, Volume I. The Level-1 Trigger, **CERN/LHCC 2000-038.**

[3] M. Koneci, J. Królikowski, G. Wrochna "Simulation Study of the RPC Based, Single Muon Trigger for CMS", **CMS TN/92 - 039**

[4] K. Banzuzi et.al. „Optimisation of the Create Layout of the RPC Pattern Comparator Trigger", **CMS IN 2001/xxx.**

[5] M. Konecki, J. Królikowski, I. M. Kudla, G. Wrochna, P Zalewski „PAC version_2 Specification",

[6] IEEE 1149.1 Tutorial, *http://www.geopel.com/english/prod/bsc/tutorial.htm*

[7] Michal Pietrusinski, praca magisterska: „Systemowe podejscie do testowania elektroniki wyzwalania w eksperymentach wysokiej energii" (Uniwersytet Warszawski 1998).

[8] "Logic Pattern Unit II (Generator/Readout) – 128 Channel VME Board", Technical Report, Bologna 2000.