dr inż. Grzegorz Bazydło
G.Bazydlo@iee.uz.zgora.pl, staff.uz.zgora.pl/gbazydlo

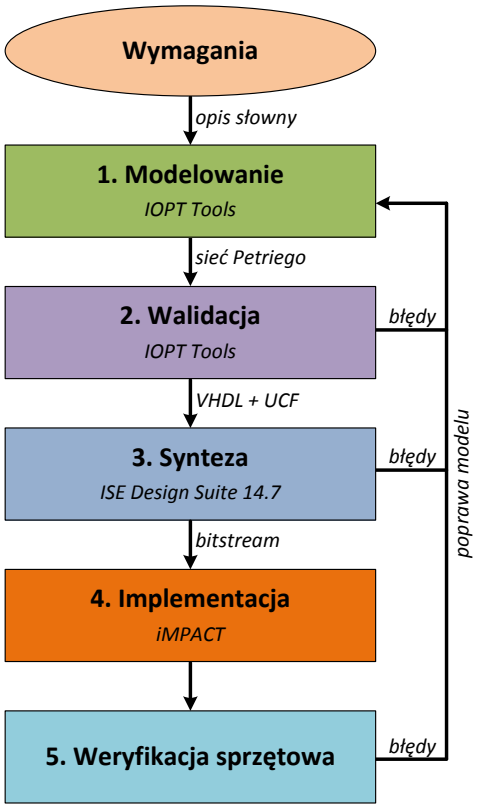
Lista zadań nr 1

Zagadnienia

- stosowanie sieci Petriego (ang. *Petri net*) jako narzędzia do modelowania algorytmów sterowania procesami dyskretnymi,
- implementacja algorytmów sterowania w układach cyfrowych FPGA (*Field-Programmable Gate Array*).

Ścieżka projektowa

Realizacja kolejnych zadań odbywać się będzie zgodnie z poniższą ścieżką projektową (rys. 1):



Rys. 1

Środowisko laboratoryjne

W trakcie realizacji listy zadań będziesz korzystać z następujących narzędzi informatycznych i sprzętu:

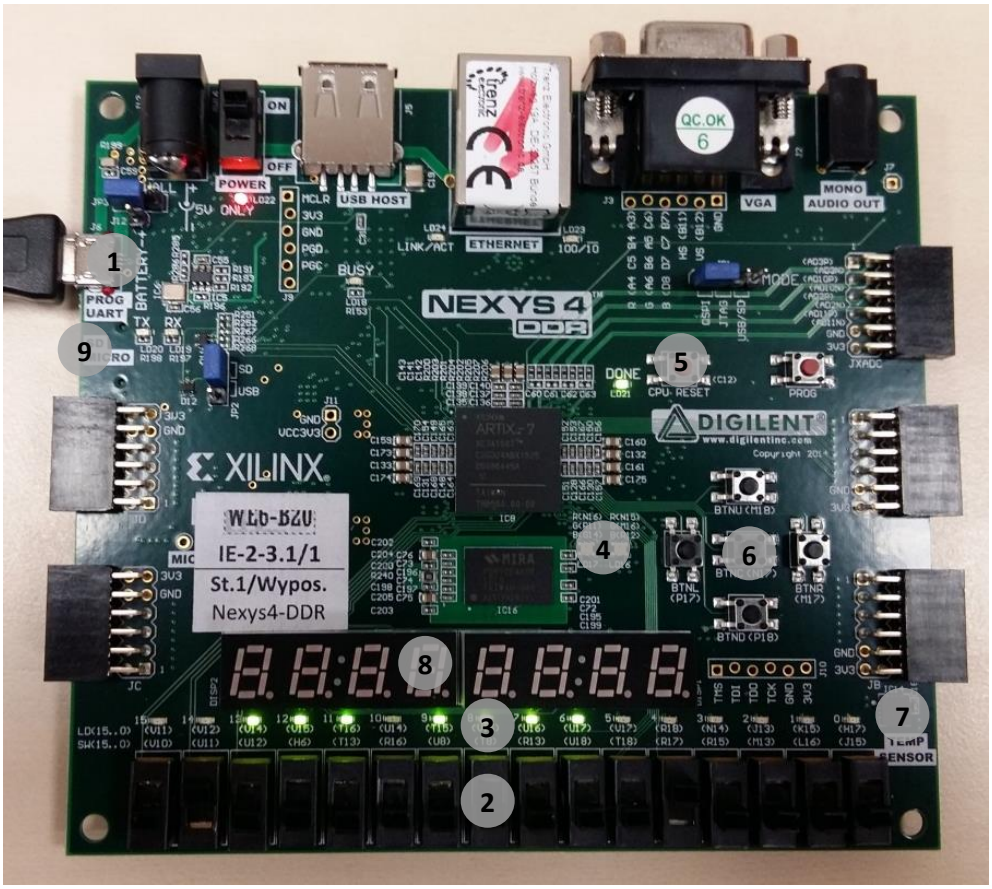
- do tworzenia modelu sieci Petriego (krok 1) oraz symulacji (krok 2) wykorzystaj *IOPT Tools*: *gres.uninova.pt/IOPT-tools* (login: *guest*, hasło: *guest* lub załóż swoje własne konto),
- do syntezy (krok 3) algorytmu sterowania w układzie cyfrowym wykorzystaj środowisko *Xilinx ISE Design Suite 14.7*,
- do fizycznej implementacji (krok 4) w układzie cyfrowym typu FPGA (programowanie układu) użyj narzędzia *iMPACT* (z pakietu narzędzi firmy Xilinx).

Sprzęt

Do implementacji sprzętowej (krok 4) oraz weryfikacji algorytmu sterowania w sprzęcie (krok 5) wykorzystaj zestaw uruchomieniowy Nexys4 DDR (rys. 2) z układem FPGA Artix-7, który oferuje zbiór portów i urządzeń peryferyjnych, w tym:

- port USB-JTAG Digilent do komunikacji i programowania FPGA ①,
- 16 przełączników użytkownika ②,
- 12-bitowe wyjście VGA,

- 3-osiowy akcelerometr,
- DDR2 128 MB,
- moduł peryferyjny Pmod dla sygnałów XADC,
- 16 diod LED użytkownika ③,
- dwie trójkolorowe diody LED ④,
- jeden programowalny przycisk CPU_RESET ⑤,
- pięć programowalnych przycisków ⑥,
- wyjście audio PWM,
- czujnik temperatury ⑦,
- szeregowa pamięć Flash,
- dwa 4-cyfrowe wyświetlacze 7-segmentowe ⑧,
- złącze karty Micro SD ⑨,
- mikrofon PDM,
- Ethernet PHY 10/100 Mb/s,
- cztery porty Pmod,
- port USB HID Host dla myszy, klawiatury i przenośnej pamięci USB (pendrive).

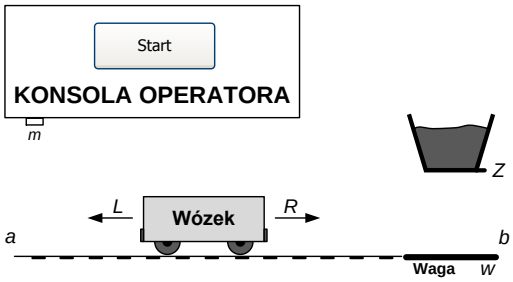


Rys. 2

Zadanie 1

Wymagania

Celem projektu jest utworzenie systemu sterowania ruchem wózka (rys. 3). Przebieg sterowania jest następujący: w chwili początkowej wózek znajduje się w lewym, skrajnym położeniu. Po naciśnięciu przycisku *Start* (aktywny sygnał *m*) wózek jedzie do w prawo (aktywny sygnał *R*), aż dojedzie do prawego, skrajnego punktu (aktywny sygnał *b*). Następnie otwierany jest zsyp (aktywny sygnał *Z*) i wózek jest ładowany. Po uzyskaniu przez wózek wymaganej wagi (aktywny sygnał *w*) zsyp jest zamykany (nieaktywny sygnał *Z*) i wózek wraca (aktywny sygnał *L*) do lewego, skrajnego punktu (aktywny sygnał *a*). Wszystkie sygnały występujące w układzie scharakteryzowano w tabeli 1.



Rys. 3

Tabela 1. Opis sygnałów układu z zadania 1

Lp.	Nazwa sygnału	Rodzaj sygnału	Źródło	Znaczenie
1.	m	wejściowy	konsola operatora	Sygnał aktywny oznacza, że użytkownik wcisnął przycisk *Start* i system rozpoczyna pracę.
2.	a	wejściowy	czujnik położenia	Sygnał aktywny oznacza, że wózek znajduje się w lewym skrajnym położeniu.
3.	b	wejściowy	czujnik położenia	Sygnał aktywny oznacza, że wózek znajduje się w prawym skrajnym położeniu.
4.	w	wejściowy	waga	Sygnał aktywny oznacza, że waga wózka przekroczyła zadaną wartość.
5.	L	wyjściowy	sterownik	Sygnał aktywny oznacza, że wózek jedzie w lewą stronę.
6.	R	wyjściowy	sterownik	Sygnał aktywny oznacza, że wózek jedzie w prawą stronę.
7.	Z	wyjściowy	sterownik	Sygnał aktywny oznacza, że zsyp jest otwarty i wózek jest ładowany.

Krok 1 – Modelowanie

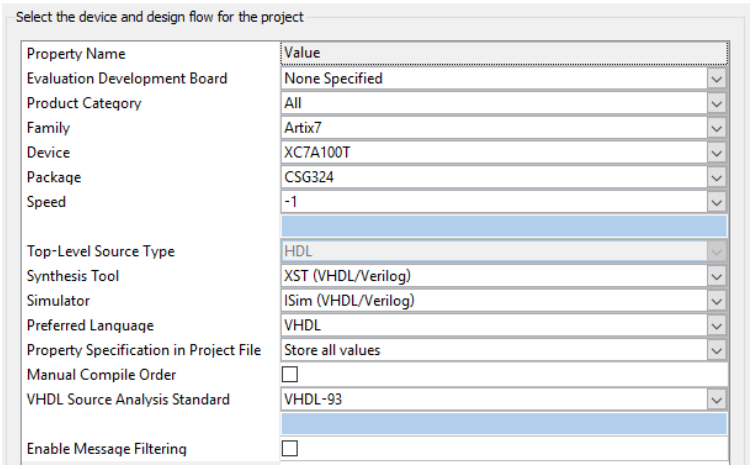
Korzystając ze środowiska *IOPT Tools* zamodeluj sieć Petriego przedstawiającą model sterowania ruchem wózka. Odpowiednio zidentyfikuj i nazwij poszczególne miejsca (np. RWP – *ruch wózka w prawo*, RWL – *ruch wózka w lewo* itp.) i tranzycje. Użyj dokładnie takich nazw sygnałów jak pokazano w tabeli 1.

Krok 2 – Walidacja

Po zaprojektowaniu modelu układu sterowania, wykonaj symulację utworzonej sieci Petriego korzystając z symulatora *IOPT Tools*. Jeżeli wyniki symulacji wykażą, że model zawiera błędy, wróć do kroku pierwszego i popraw go. Jeżeli model przeszedł poprawnie etap symulacji, wyeksportuj jego opis w syntezowalnym języku opisu sprzętu VHDL.

Krok 3 – Synteza

Następnie korzystając z wytycznych prowadzącego utwórz projekt w środowisku *Xilinx ISE Design Suite 14.7*. Parametry projektu pokazano na rys. 4.



Rys. 4

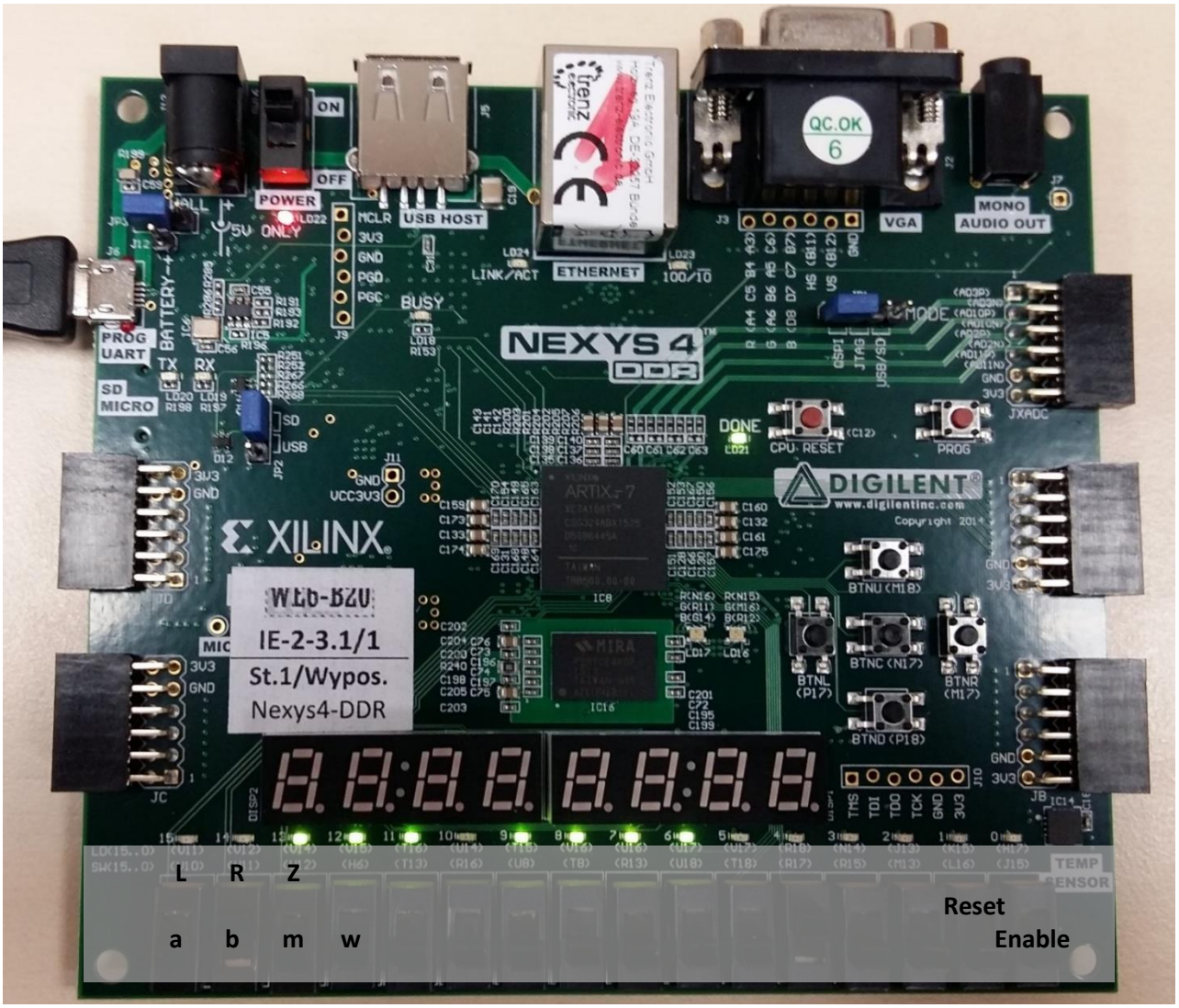
Następnie dodaj do projektu plik `smart01_zad01_nexys4ddr.ucf` dostępny na stronie z materiałami do zajęć. Plik ten zawiera przypisanie sygnałów używanych w modelu do konkretnych wyprowadzeń obudowy układu FPGA. Kolejnym etapem jest uruchomienie procesu syntezy, implementacji oraz generowania pliku wynikowego (bitstream) zawierającego dane służące do zaprogramowania układu.

Krok 4 – Implementacja

Podłącz do komputera układ *Nexys 4 DDR* i korzystając z narzędzia *iMPACT* oraz wytycznych prowadzącego wykonaj programowanie układu FPGA za pomocą wygenerowanego w poprzednim kroku bitstreamu.

Krok 5 – Weryfikacja sprzętowa

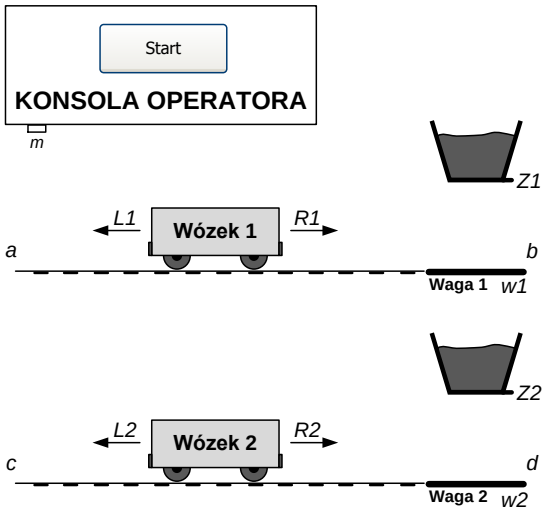
Zweryfikuj działanie układu przełączając poszczególne przełączniki i obserwując diody LED (rys. 5).



Rys. 5

Zadanie 2

Zrealizuj całą ścieżkę projektową (rys. 1) opisaną w zadaniu 1 do układu sterowania ruchem dwóch wózków (rys. 6). Odpowiednio zidentyfikuj i nazwij poszczególne miejsca (np. RW1P – *ruch wózka 1 w prawo*, RW2L – *ruch wózka 2 w lewo* itp.) i tranzycje. Przebieg sterowania jest następujący: ruch wózków zaczyna się po naciśnięciu przycisku *Start* (aktywny sygnał *m*). Wózki startują z lewego, skrajnego położenia *a* oraz *c* i dojeżdżają do punktów *b* oraz *d*, gdzie następuje otwarcie zsypów (sygnały *Z1* i *Z2*) i załadowanie wózków. Po załadowaniu każdego wózka (aktywny sygnał wagi *w1* lub *w2*) odpowiednie zsypy są zamykane. Powrót wózków następuje w kolejności: najpierw wraca wózek 1, a gdy ten dojedzie do punktu *a*, wówczas wraca wózek 2. Wszystkie sygnały występujące w układzie scharakteryzowano w tabeli 2.



Rys. 6

Tabela 2. Opis sygnałów układu z zadania 2

Lp.	Nazwa sygnału	Rodzaj sygnału	Źródło	Znaczenie
1.	m	wejściowy	konsola operatora	Sygnał aktywny oznacza, że użytkownik wcisnął przycisk *Start* i system rozpoczyna pracę.
2.	a	wejściowy	czujnik położenia	Sygnał aktywny oznacza, że wózek 1 znajduje się w lewym skrajnym położeniu.
3.	b	wejściowy	czujnik położenia	Sygnał aktywny oznacza, że wózek 1 znajduje się w prawym skrajnym położeniu.
4.	c	wejściowy	czujnik położenia	Sygnał aktywny oznacza, że wózek 2 znajduje się w lewym skrajnym położeniu.
5.	d	wejściowy	czujnik położenia	Sygnał aktywny oznacza, że wózek 2 znajduje się w prawym skrajnym położeniu.
6.	w1	wejściowy	waga 1	Sygnał aktywny oznacza, że waga wózka 1 przekroczyła zadaną wartość.
7.	w2	wejściowy	waga 2	Sygnał aktywny oznacza, że waga wózka 2 przekroczyła zadaną wartość.
8.	L1	wyjściowy	sterownik	Sygnał aktywny oznacza, że wózek 1 jedzie w lewą stronę.
9.	R1	wyjściowy	sterownik	Sygnał aktywny oznacza, że wózek 1 jedzie w prawą stronę.
10.	L2	wyjściowy	sterownik	Sygnał aktywny oznacza, że wózek 2 jedzie w lewą stronę.
11.	R2	wyjściowy	sterownik	Sygnał aktywny oznacza, że wózek 2 jedzie w prawą stronę.
12.	Z1	wyjściowy	sterownik	Sygnał aktywny oznacza, że zsyp 1 jest otwarty i wózek 1 jest ładowany.
13.	Z2	wyjściowy	sterownik	Sygnał aktywny oznacza, że zsyp 2 jest otwarty i wózek 2 jest ładowany.

Aby wykonać syntezę (krok 3) dodaj do projektu w *ISE Design Suite 14.7* plik `smart01_zad02_nexys4ddr.ucf` dostępny na stronie z materiałami do zajęć. Następnie zweryfikuj działanie układu przełączając poszczególne przełączniki i obserwując diody LED (rys. 7).

Rys. 7