

## Wprowadzenie:

Edytor maszyny stanów pozwala w prosty sposób graficzny projektować układy cyfrowe. Ponieważ projekt maszyny stanów można w prosty sposób przenieść na inny sprzęt, edytory stanów stają się coraz bardziej popularne wśród projektantów którzy cenią niezależnie się od platformy sprzętowej. Ten tutorial jest dedykowany użytkownikom programu Active-HDL, którzy chcą się nauczyć metody projektowania układów cyfrowych z wykorzystaniem „State Diagram Editor” (edytor maszyny stanów).

## Cel tej instrukcji:

Nauczenie użytkownika jak wykorzystać State Diagram Editor do wprowadzania diagramów maszyny stanów oraz jak przeprowadzić jej symulację i logiczną syntezę.

## Opis przykładu:

W trakcie realizacji ćwiczenia powstanie maszyna stanów symulująca grę w Blackjack. Celem gry jest pobranie kart których suma wag będzie jak najbliższa liczbie 21. Każda karta ma wartość od 2 do 11, gdzie karta o wadze 11 jest zwana ACE i może zostać policzona jako 1. Gracz który osiągnie najwyższy wynik ale mniejszy lub równy 21 będzie zwycięzcą.

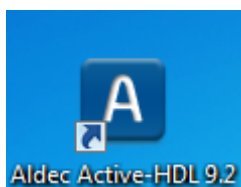
Żądanie otrzymania dodatkowej karty jest sygnalizowane przez sygnał wyjściowy SAY\_CARD (ma wartość logiczną '1' gdy jest aktywny). Żądanie to jest wysyłane gdy suma kart jest mniejsza od 17. SAY\_CARD nie zostanie nigdy wygenerowane gdy suma kart przewyższa 21.

Otrzymanie nowej karty jest sygnalizowane zmianą wartości logicznej sygnału NEW\_CARD z '0' na '1'. Maszyna stanów powinna zasygnalizować moment uzyskania sumy kart przekraczającej 21 na wyjściu SAY\_BUST. Przypadek gdy suma kart jest z zakresu od 17 do 21 jest sygnalizowany na wyjściu SAY\_HOLD.

Suma kart jest widoczna na wyjściu TOTAL. Po osiągnięciu warunku SAY\_BUST lub SAY\_HOLD, maszyna powinna pozostać w ostatnim stanie aż do momentu gdy pojawi się aktywny sygnał NEW\_GAME. Sygnał NEW\_GAME ponadto powoduje zresetowanie maszyny stanów.

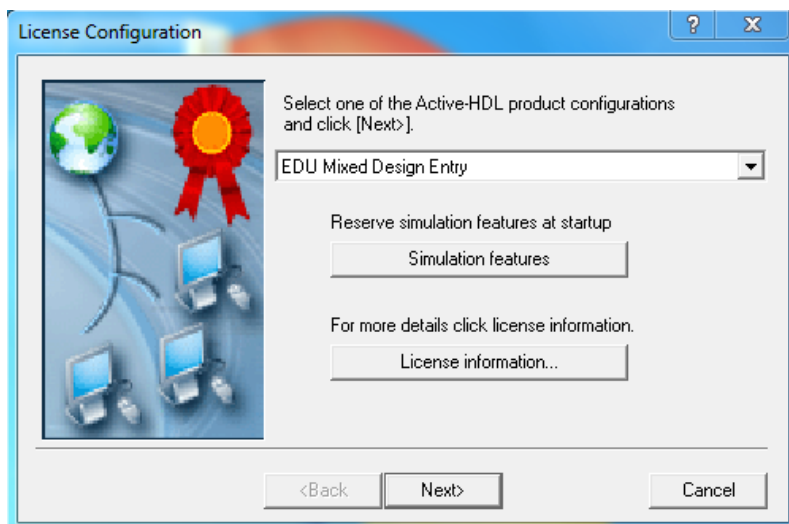
Maszyna stanów zostanie zaimplementowana jako graficzny graf. Diagram stanów zostanie automatycznie zamieniony na odpowiadający jej kod VHDL. Następnie będzie można sprawdzić poprawne działanie automatu poprzez symulację kodu VHDL.

1. Uruchomić „Aldec Active-HDL 9.2” (skrót na pulpicie). Program uruchamia się dosyć długo – cierpliwości.



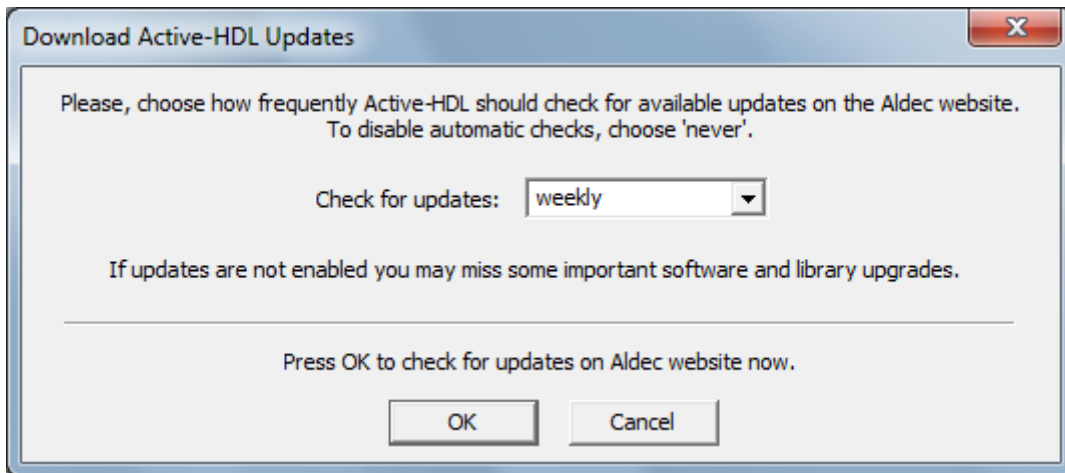
Rys. 1. Ikona na pulpicie – skrót do Aldec Active-HDL.

2. Pojawi się okno z pytaniem o licencję. Kliknąć „Next”.



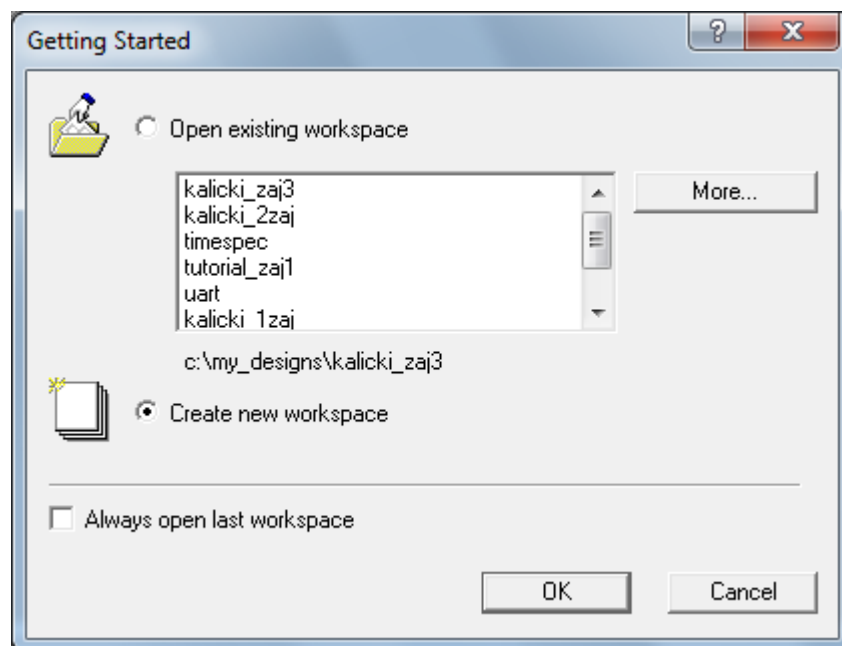
Rys. 2. Okno konfiguracji licencji.

3. Jeżeli program zapyta się o sprawdzenie aktualizacji, to wybrać „Cancel”.



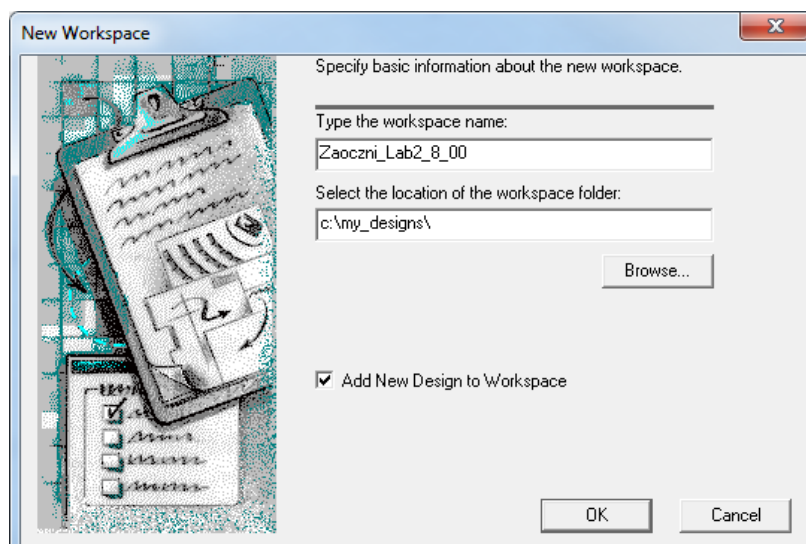
Rys. 3. Okno aktualizacji programu.

4. W oknie wybrać „Create new workspace” i kliknąć „OK”.



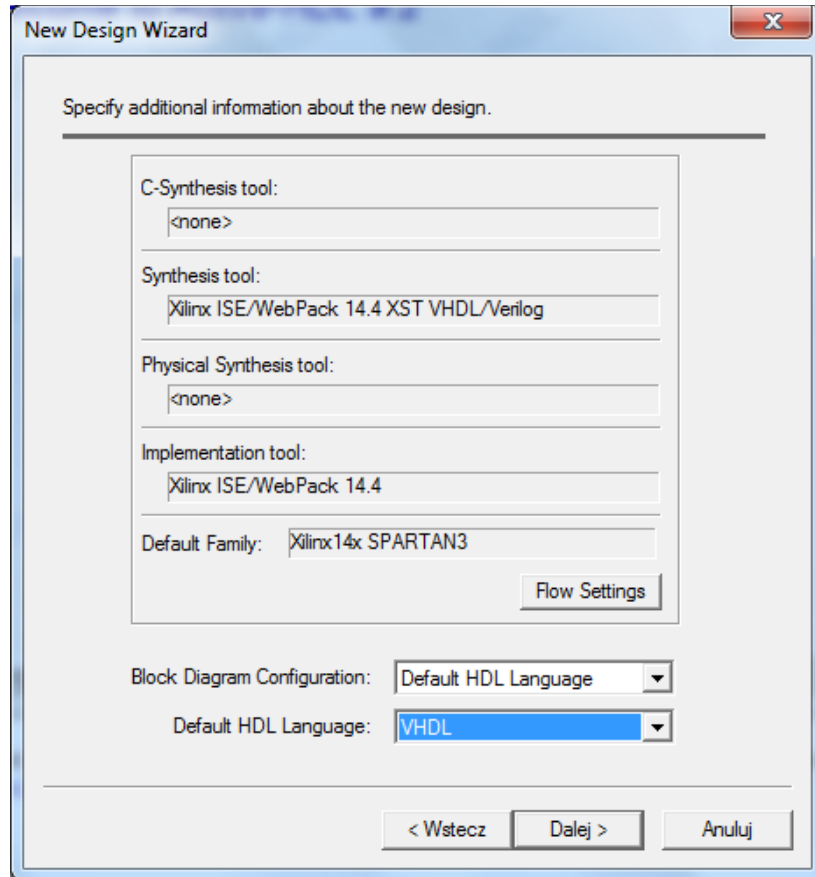
Rys. 4. Wybór przestrzeni roboczej.

5. Wprowadzić nazwę przestrzeni projektowej (tekst bez odstępów, bez polskich liter oraz bez znaków specjalnych) i kliknąć „OK”.



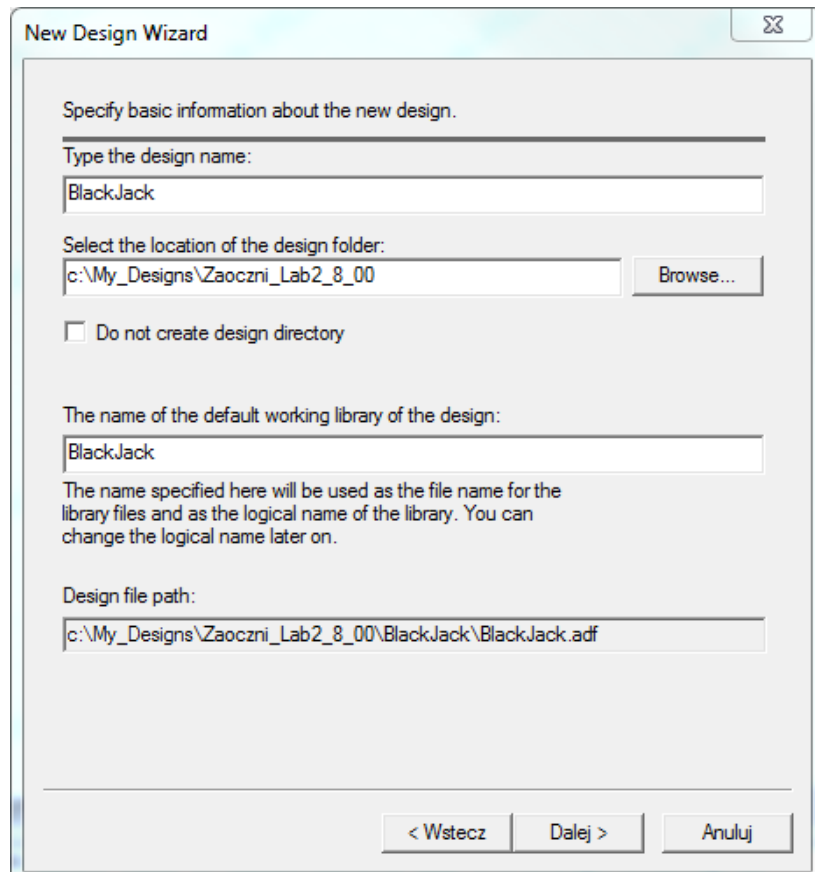
Rys. 5. Nadanie nazwy przestrzeni roboczej.

6. W kolejnym oknie wybrać „Create an Empty Design with Design Flow” i kliknąć „Dalej”.
7. Sprawdzić czy ustawienia Design Flow są takie jak na poniższym rysunku. W przypadku różnic kliknąć „Flow Settings” i dokonać odpowiednich zmian. Jeżeli wszystko się zgadza, kliknąć „Dalej”.



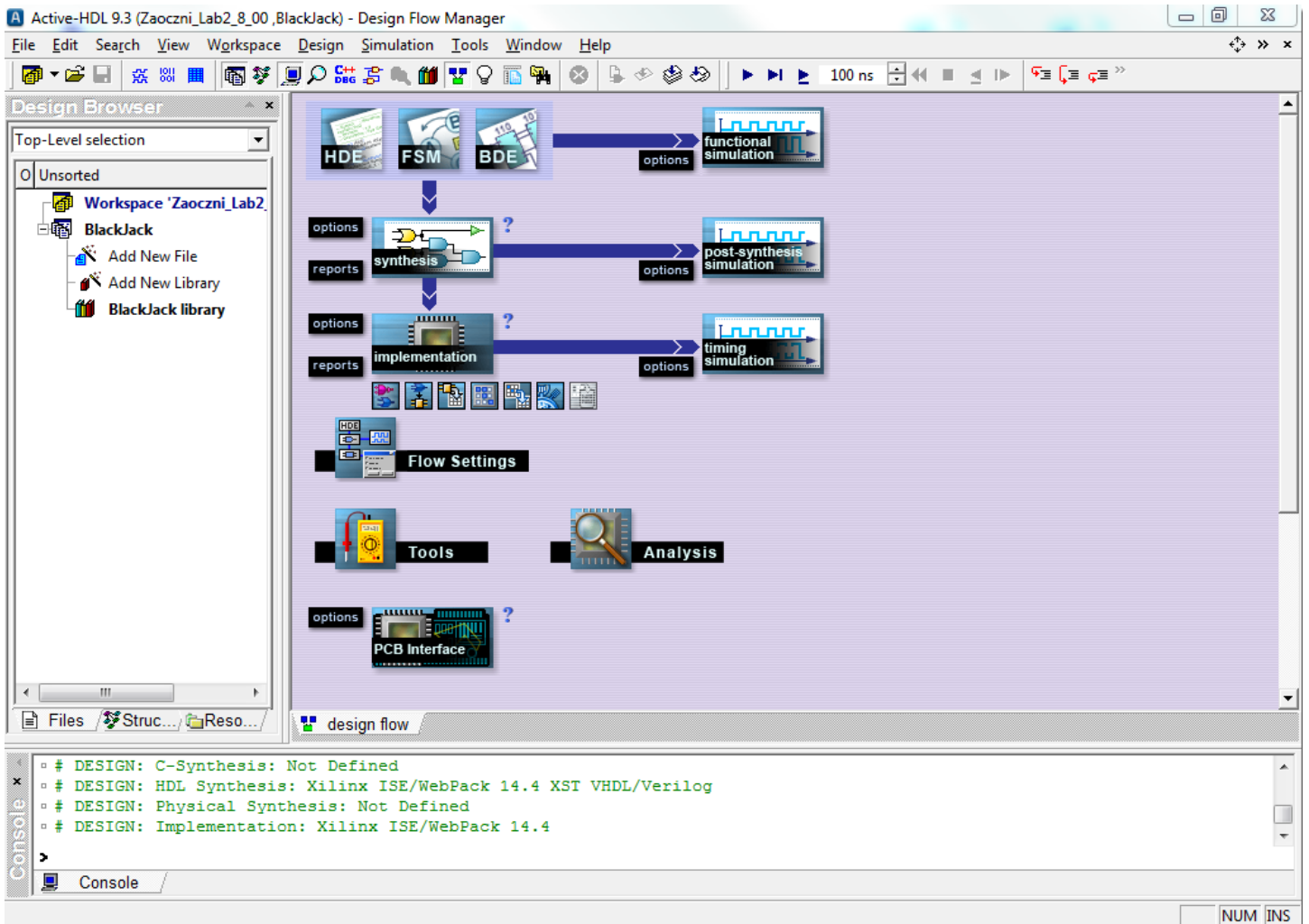
Rys. 6. Ustawienia syntezy i implementacji.

8. Wprowadzić nazwę projektu (tekst bez odstępów, bez polskich liter i znaków specjalnych) i kliknąć „Dalej”.



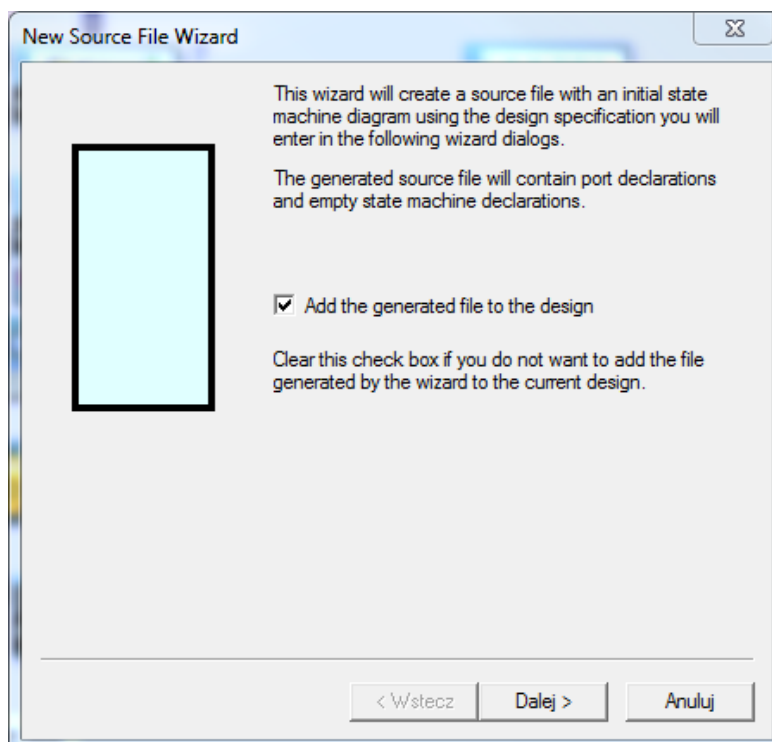
Rys. 7. Nadanie nazwy projektowi.

9. W kolejnym oknie wybrać „Zakończ”. Powinniśmy na ekranie otrzymać widok jak na poniższym rysunku.



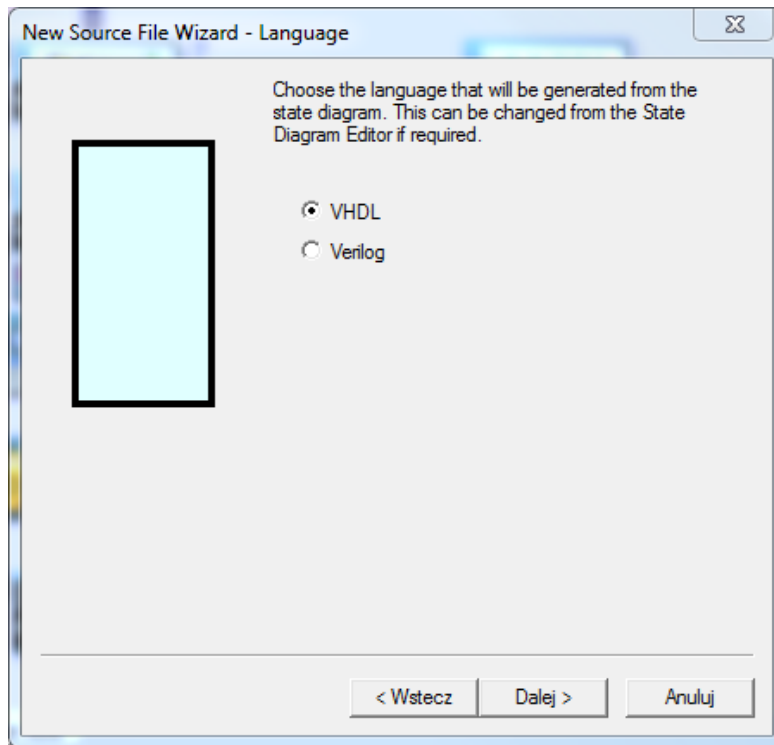
Rys. 8. Główne okno programu po założeniu projektu.

10. Z menu programu wybierz File -> New -> State Diagram. Otworzy się okno kreatora nowych plików źródłowych. W pierwszym oknie zostawiamy domyślnie zaznaczoną opcję „Add the generated file to the design” i klikamy „Dalej”.



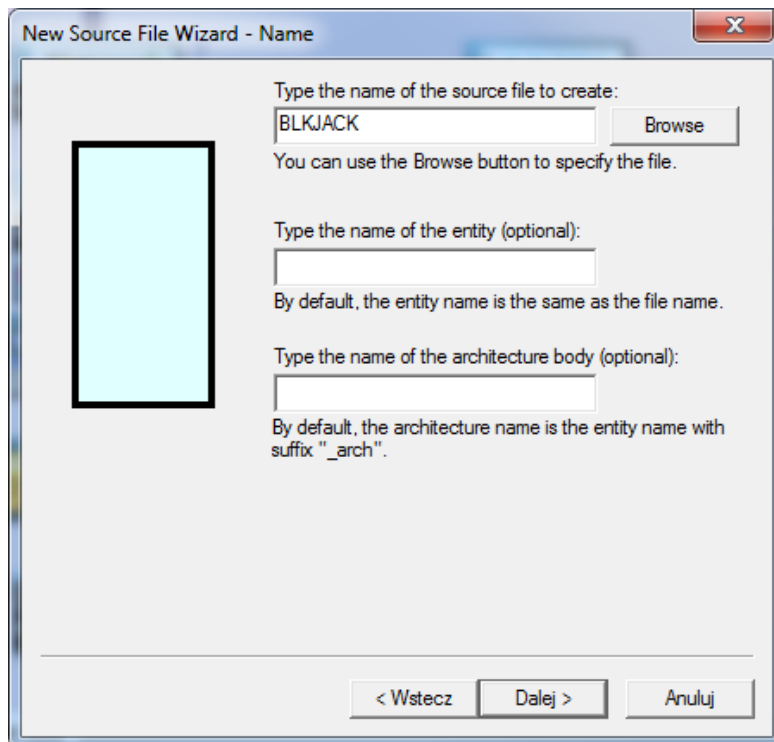
Rys. 9. Pierwsze okno kreatora diagramu maszyny stanów.

11. Wskaż język VHDL dla generowanego kodu.



Rys. 10. Wybór języka generowanego kodu.

12. Wpisz nazwę pliku „BLKJACK” (jeżeli podasz inną, możesz mieć później problemy z wykonaniem ćwiczenia) i kliknij „Dalej”.



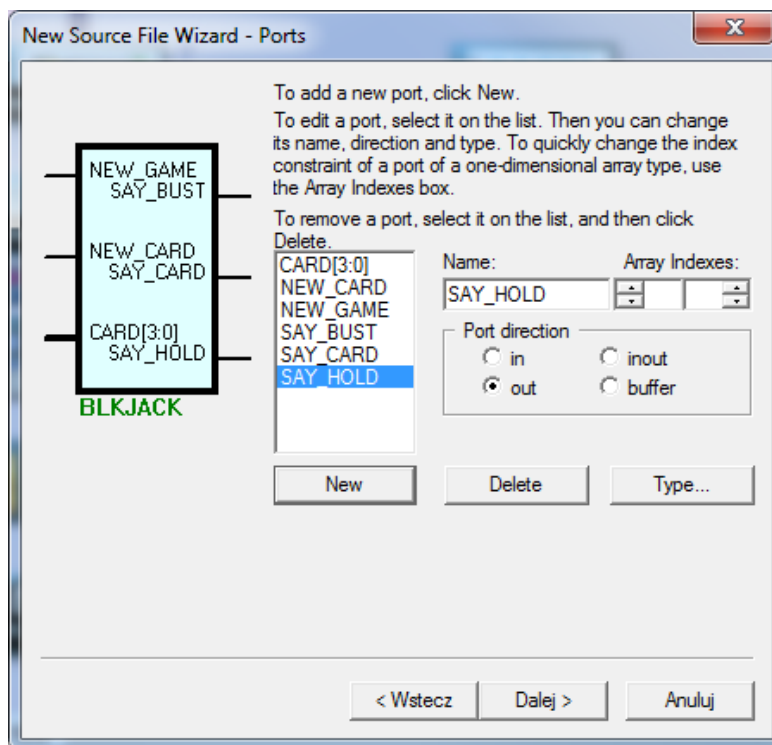
Rys. 11. Wybór nazwy pliku diagramu.

Diagramy maszyny stanów mają rozszerzenie \*.asf. Pełna nazwa tworzonego pliku to „BLKJACK.asf”

13. Kolejne okno pozwala na dołożenie portów do grafu. Aby dodać nowy port/sygnal należy kliknąć przycisk „New”. W polu „Name” wprowadza się jego nazwę. Pola „Array Indexes” umożliwiają definiowanie magistral. Trzeba też zdefiniować kierunek sygnału (in / out / inout) oraz jego typ (używamy wyłącznie STD\_LOGIC oraz STD\_LOGIC\_VECTOR!).  
Proszę dołożyć sześć sygnałów zgodnie z poniższą tabelą a następnie kliknąć „Dalej”.

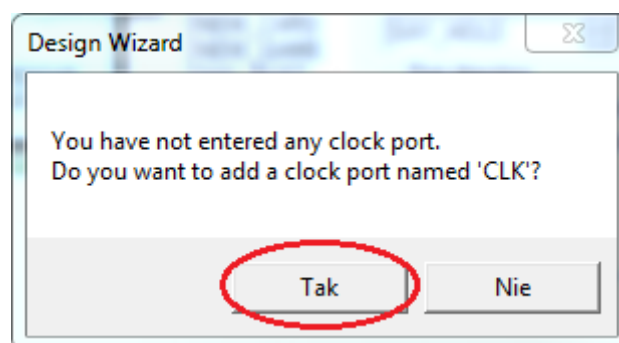
Name	Array Indexes	Port direction	Type
NEW_GAME	(puste)	In	STD_LOGIC
NEW_CARD	(puste)	In	STD_LOGIC
CARD	3 - 0	In	STD_LOGIC_VECTOR
SAY_BUST	(puste)	Out	STD_LOGIC
SAY_CARD	(puste)	Out	STD_LOGIC
SAY_HOLD	(puste)	Out	STD_LOGIC

Aby zmodyfikować wcześniej wprowadzony sygnał należy zaznaczyć jego nazwę i dokonać odpowiednich zmian. Błędne / niepotrzebne sygnały można skasować przyciskiem „Delete” na klawiaturze. W lewej części okna mamy graficzną reprezentację projektowanego elementu. Stworzone wejścia układają się na lewej krawędzi bloczka, z kolei wyjścia układu pojawiają się na prawej krawędzi. Magistrale oznaczane są grubszym „przewodem”.



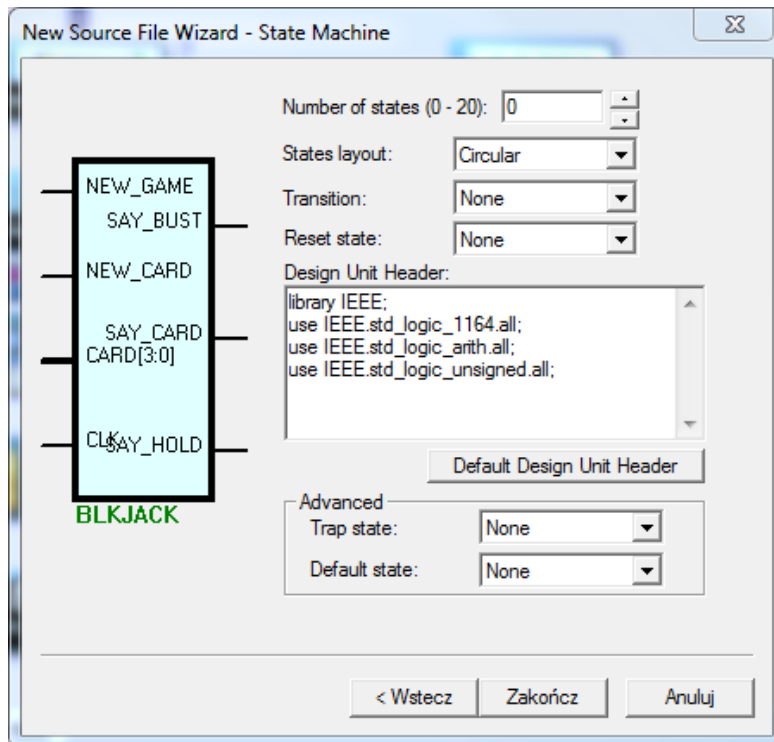
Rys. 12. Konfiguracja portów grafu.

14. Gdy program zapyta czy dodać port dla sygnału zegarowego „CLK”, wybierz „TAK”.



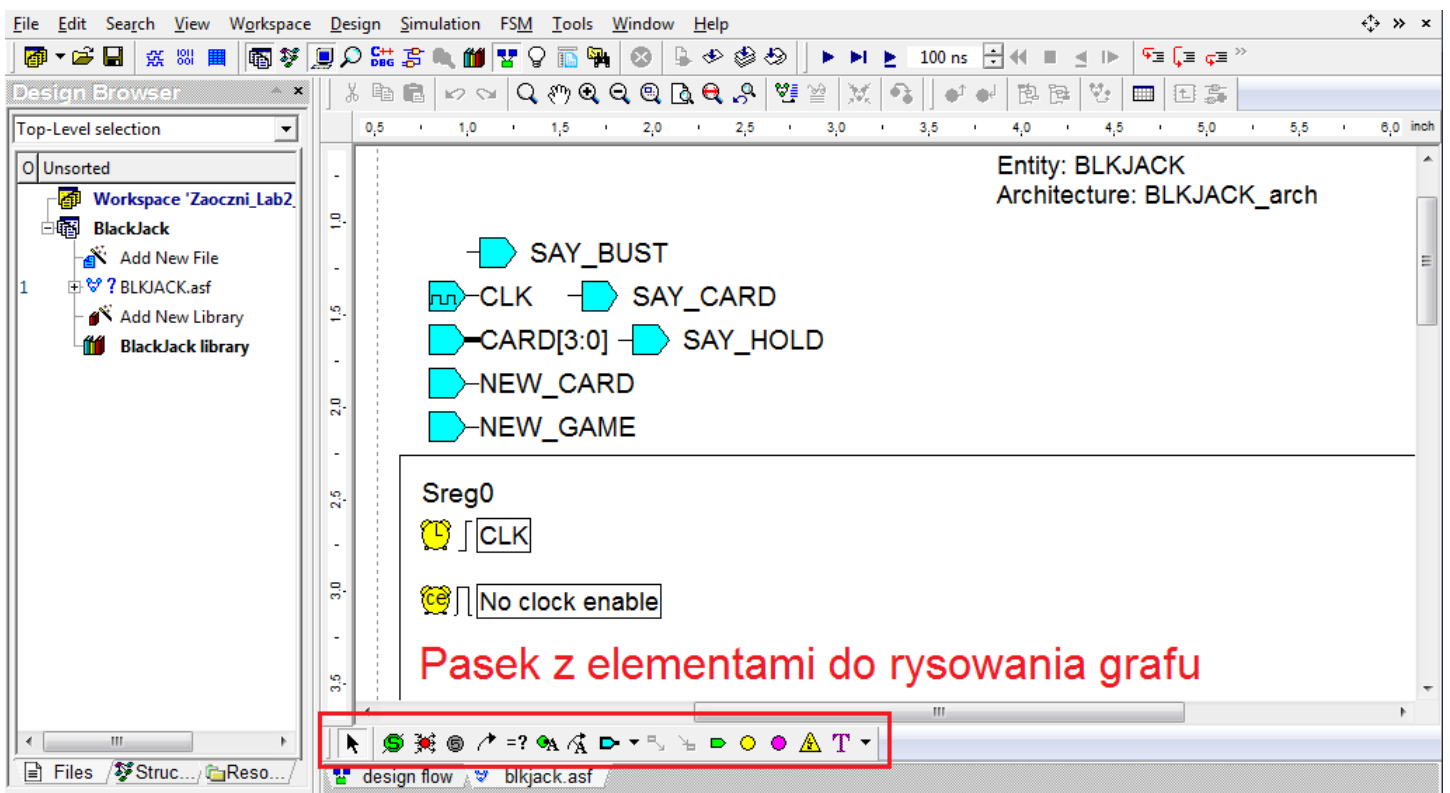
Rys. 13. Okno z pytaniem o sygnał zegarowy.

15. Kreator Nowych plików – Maszyny Stanów pozwala zdefiniować główne parametry tworzonego diagramu. Można między innymi zdefiniować liczbę stanów które zostaną wstawione przez kreatora, ułożenie symboli reprezentujących stan (okrąg, poziomo, pionowo), kierunek przejść pomiędzy stanami (do przodu, do tyłu, w obu kierunkach), numer stanu po resecie, numer stanu „trap”, domyślny stan oraz nagłówek automatycznie generowanego pliku vhdl.
- Pozostaw domyślne wartości nie zmienione i kliknij „Zakończ”.



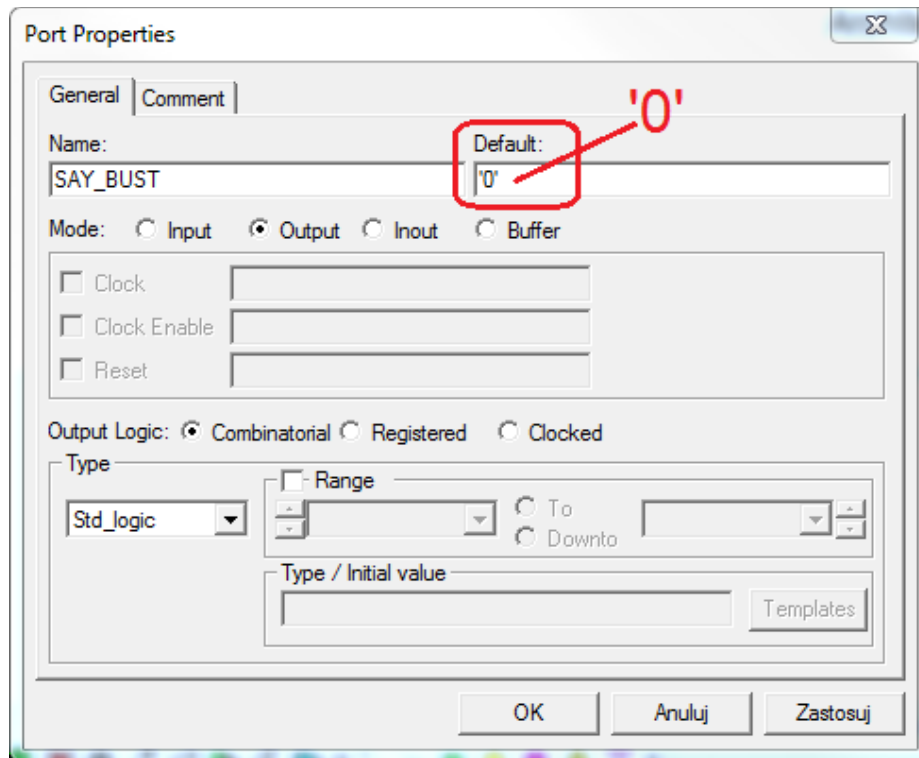
Rys. 14. Parametry tworzonego diagramu.

16. Po zakończeniu działania kreatora nowego pliku vhd, powinniśmy uzyskać rezultat podobny do poniższego. W lewej części okna jest wyświetlony Design Browser (lista plików w projekcie), prawa część to zawartość aktualnie przeglądanej pliku.



Rys. 15. Główne okno programu Active-HDL po dodaniu do projektu pliku diagramu maszyny stanów.

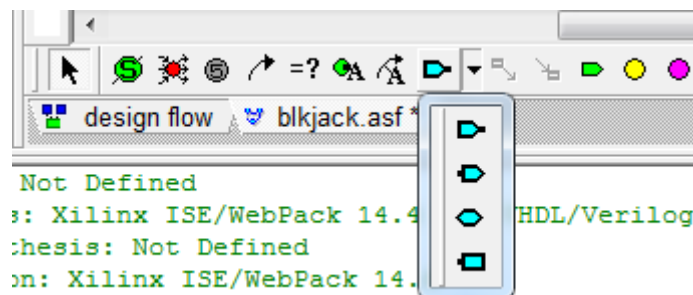
17. Ustaw domyślną wartość trzech sygnałów wyjściowych SAY\_BUST, SAY\_CARD, SAY\_HOLD, równą '0'. W tym celu dwukrotnie kliknij symbol portu i uzupełnij pole „Default” w oknie „Port Properties” (dla wszystkich trzech portów osobno).



Rys. 16. Okno właściwości portu.

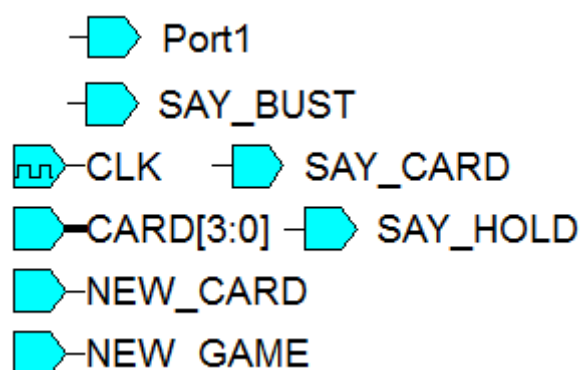
18. Dołożenie dodatkowego portu.

Kolejne porty diagramu mogą zostać dołożone na dowolnym etapie projektu. W tym celu należy rozwinąć listę portów na pasku i wybrać odpowiedni kierunek portu.



Rys. 17. Ikona skrótu dodawania kolejnych portów do diagramu.

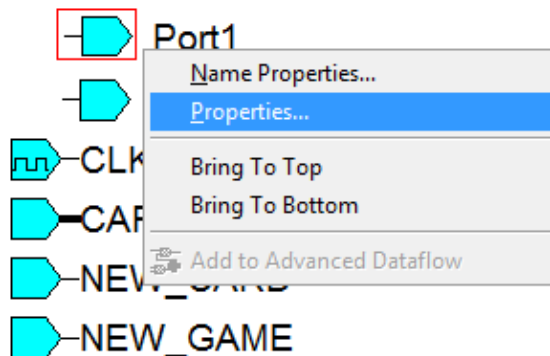
- a) Z rozwijanej listy obok symbolu portu na pasku, wybierz port wyjściowy (Output Port) i kliknij powyżej portu SAY\_BUST na diagramie. Pojawi się nowy port o nazwie „Port1”. Naciśnij klawisz „Escape” na klawiaturze by wyłączyć dodawanie kolejnych portów.



Rys. 18. Dołożenie nowego portu.

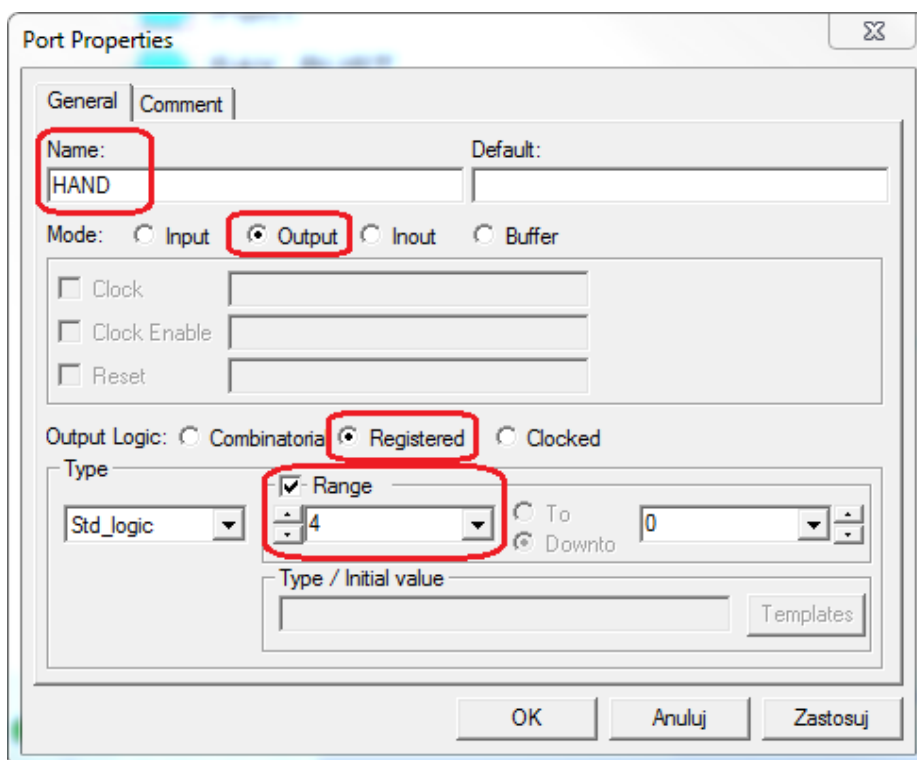


b) Kliknij prawym przyciskiem myszy na wstawionym porcie i z menu kontekstowego wybierz „Properties...”.



Rys. 19. Otwarcie okna „właściwości” portu.

c) W oknie właściwości, podaj nazwę portu „HAND” oraz ustaw zakres magistrali „4 downto 0”. Ponadto, ustaw kierunek „Out” oraz typ wyprowadzenia „Registered”. Kliknij „OK” na koniec. Maksymalna wartość kart może wynieść 26 (możemy żądać nowej karty gdy mamy wartość 16; nowa karta może mieć wagę 10). Dlatego port HAND musi umożliwiać przekazywanie liczb z zakresu od 0 do 31, co odpowiada 5-bitowemu portowi / magistrali.

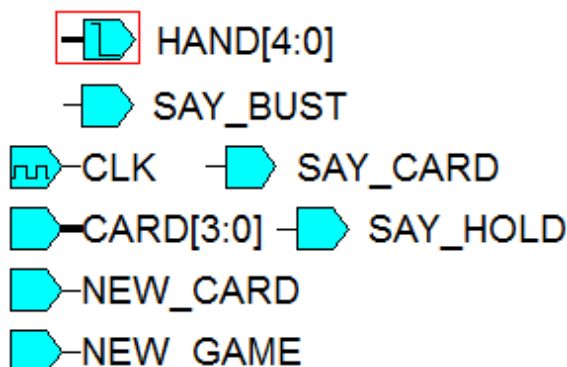


Rys. 20. Właściwości nowego portu.

W oknie właściwości portu można ustawić dany port jako sygnał zegarowy, a dla portów wyjściowych można wybrać typ kombinacyjny albo rejestrowany.

Wyjścia rejestrowane podtrzymują swoją wartość w dodatkowym rejestrze PIPO (zmiana wyjścia wymaga aktywnego sygnału zegarowego).

Wyjścia kombinacyjne zmieniają się jak tylko zmianie ulegnie stan wejść.



Rys. 21. Rejestrowany port wyjściowy HAND.

## 19. Definiowanie dodatkowych zmiennych.

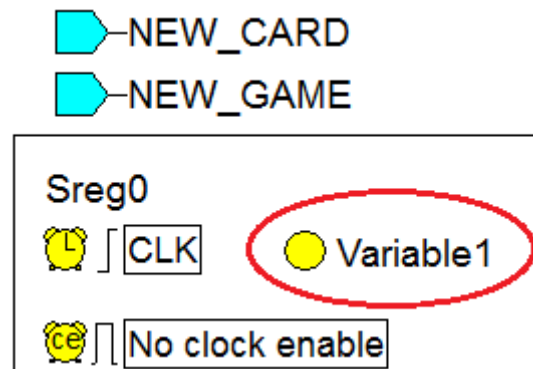
Maszyna stanów potrzebuje dodatkową zmienną (variable) do przechowywania informacji czy natrafiono na kartę ACE (jej waga może być liczona jako 1 lub 11). W przypadku posiadania karty ACE i wyniku przekraczającego 21, karta ACE może zostać policzona z wagą 1 w celu obniżenia wyniku.

- a) Na pasku kliknij ikonę skrótów „Signal/Variable” (żółte kółko):



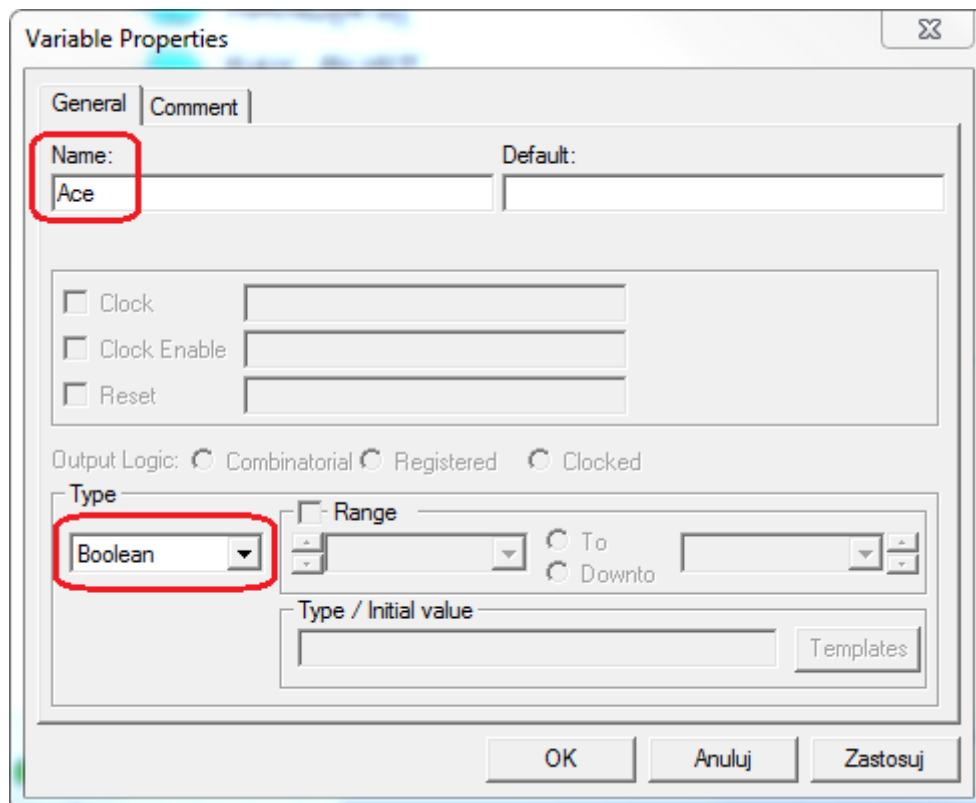
Rys. 22. Ikona „Signal/Variable”.

- b) Umieść zmienną poniżej pola tekstowego „Sreg0” na diagramie (w przeciwnym wypadku dodasz sygnał wewnętrzny, co będzie pokazane w jednym z następujących podpunktów). Wciśnij klawisz Escape na klawiaturze aby wyłączyć dodawanie kolejnych zmiennych.



Rys. 23. Dołożenie zmiennej do diagramu.

- c) Kliknij prawym przyciskiem myszy na symbolu zmiennej i wybierz „Properties...”. W oknie ustawień, zmień nazwę na „Ace”, ustaw typ zmiennej na „Boolean” i kliknij „OK” by zatwierdzić.

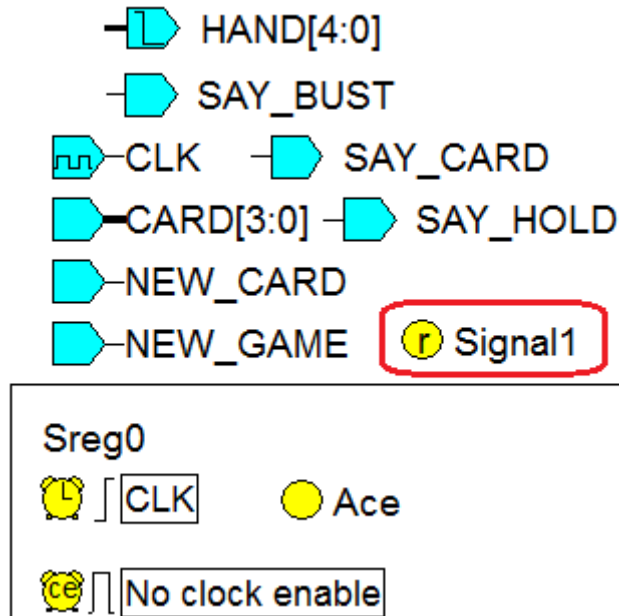


Rys. 24. Właściwości nowej zmiennej.

## 20. Definiowanie dodatkowego sygnału.

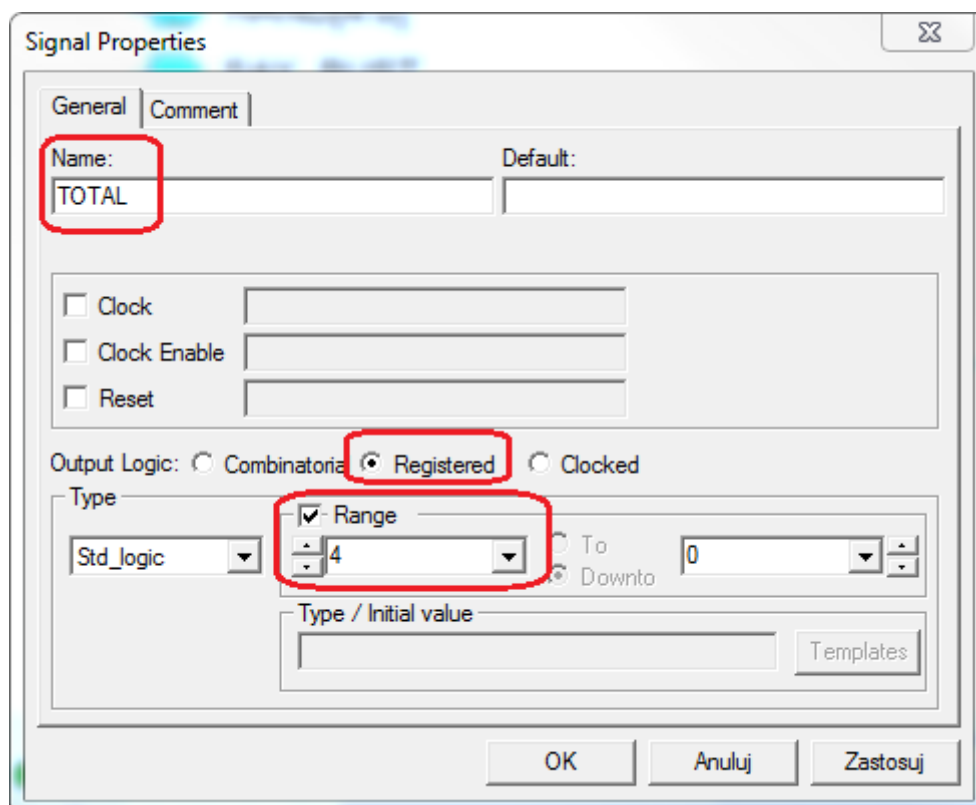
Ponieważ wyjście HAND jest portem wyjściowym, to maszyna stanów nie będzie mogła czytać jego wartości (port podaje aktualny wynik gry). Musimy wprowadzić dodatkowy sygnał przechowujący sumę wag posiadanych kart. Oczywiście będzie to wektor / magistrala o identycznej szerokości co port HAND. Sygnał ten nazwiemy TOTAL.

- Na pasku kliknij ikonę skrótów „Signal/Variable” (żółte kółko) podobnie jak w podpunkcie 19. a).
- Umieść sygnał na „marginesie” diagramu, na przykład obok portu NEW\_GAME.



Rys. 25. Dołożenie sygnału.

- Kliknij prawym przyciskiem myszy na symbolu sygnału i wybierz „Properties...”. W oknie ustawień, zmień nazwę na „TOTAL”, ustaw typ logiki „Registered”, zakres na „4 downto 0” i kliknij „OK” by zatwierdzić.

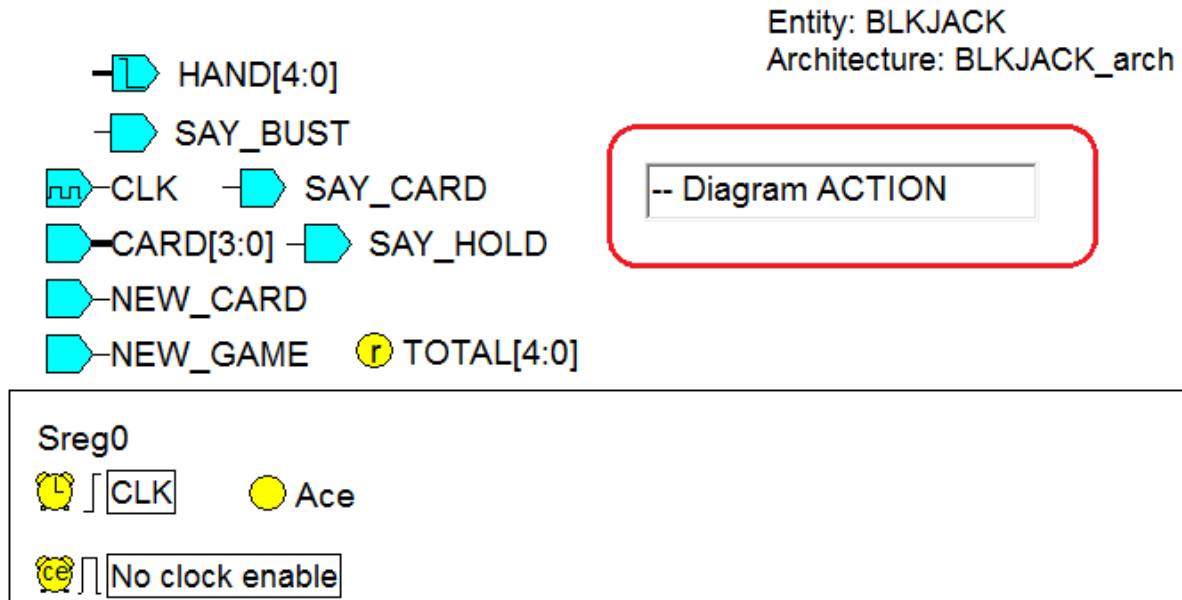


Rys. 26. Okno własności utworzonego sygnału.

## 21. Dołożenie akcji dla diagramu maszyny stanów.

Skoro mamy sygnał TOTAL, to musimy jakoś przepisać jego wartość na wyjście HAND. Takie przypisanie powinno zachodzić współbieżnie, niezależnie od stanu w którym znajduje się automat. Można to osiągnąć poprzez dodanie akcji diagramu (Diagram Actions).

- a) Wybierz z głównego menu programu FSM -> Action -> Diagram. Kursor myszy ulegnie zmianie. Umieść kursor na „marginesie” diagramu (np. na prawo od portów) i kliknij lewym przyciskiem myszy.

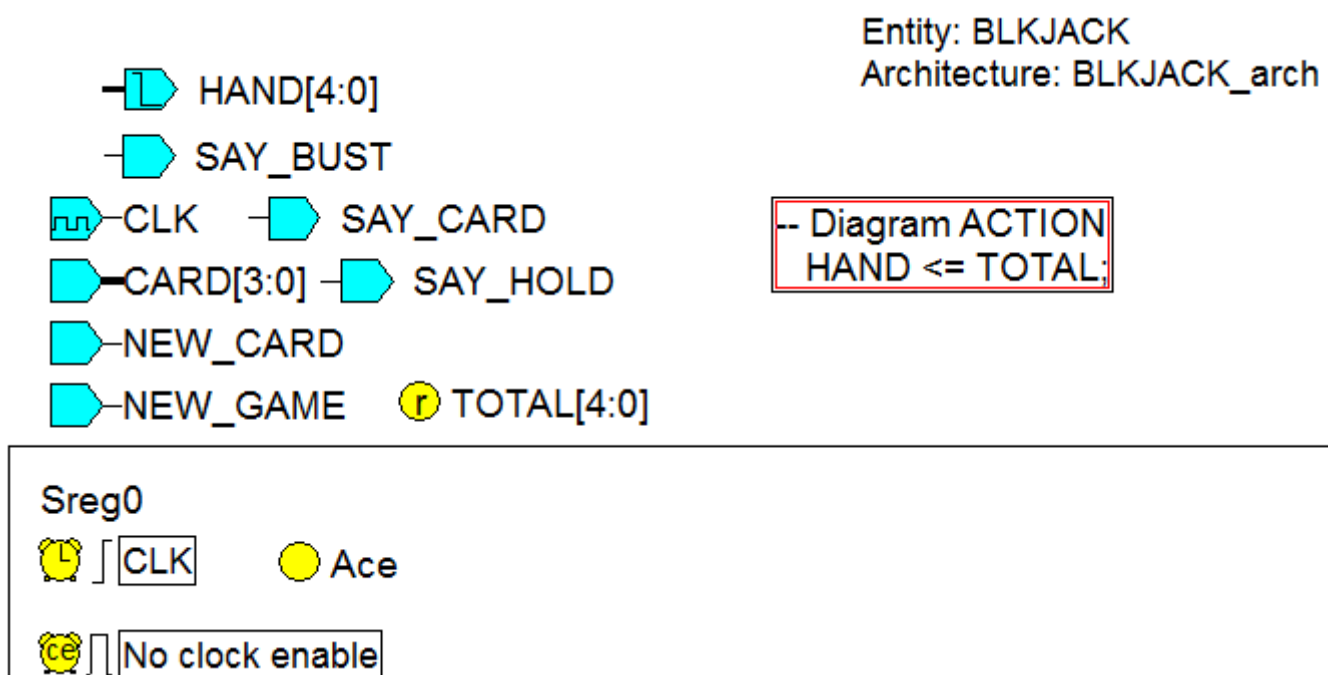


Rys. 27. Wstawienie „Diagram Action”.

- b) Wciśnij klawisz Escape na klawiaturze by zakończyć dodawanie kolejnych Akcji Diagramu.  
c) Kliknij dwukrotnie wstawione pole tekstowe i wpisz w nowym oknie:

```
--Diagram ACTION  
HAND <= TOTAL;
```


- d) Kliknij ikonę dyskietki by zapisać zmiany i zamknij okno edycji kodu.  
W tym momencie, diagram powinien być podobny do poniższego rysunku.

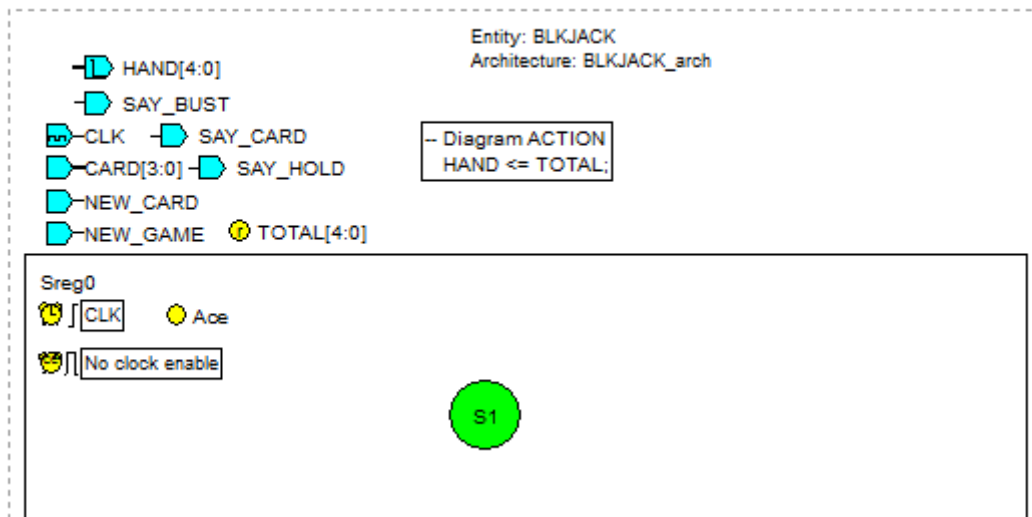


Rys. 28. Obecna postać diagramu maszyny stanów.

## 22. Ustawienie stanu po resecie.

Każda maszyna stanów musi zostać zainicjalizowana / zresetowana. W przypadku gry w Blackjack, inicjalizacja zachodzi w momencie rozpoczęcia nowej gry.

- Na pasku kliknij ikonę skrótów „State” .
- Umieść symbol stanu na środku kartki, tak jak jest pokazane poniżej (kliknięcie lewym przyciskiem myszy wstawia symbol stanu, prawy przycisk myszy wyłącza tryb wstawiania nowych stanów). Nazwa stanu zostanie automatycznie nadana jako S1.

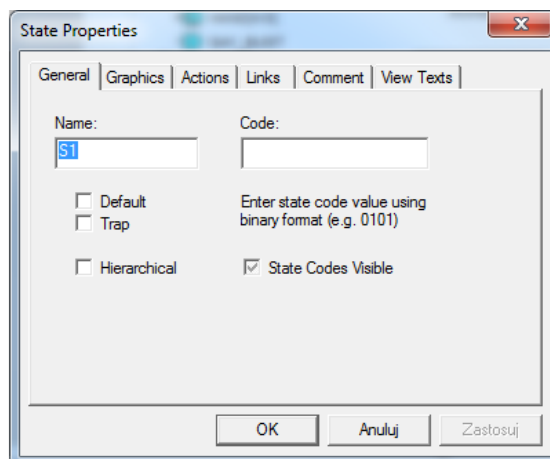


Rys. 29. Schemat grafu po wstawieniu pierwszego stanu.


- Kliknij dwukrotnie stan S1 i zmień jego nazwę na „Start”.  
Jeżeli dwukrotnie kliknięto napis, to zmieni się on w pole tekstowe (lewa część rysunku poniżej). Aby zatwierdzić zmianę należy kliknąć poza polem tekstowym.  
Jeżeli dwukrotnie kliknięto symbol stanu, to otworzy się okno z właściwościami (prawa część rysunku poniżej). W tym przypadku aby zatwierdzić zmianę należy kliknąć „OK”.

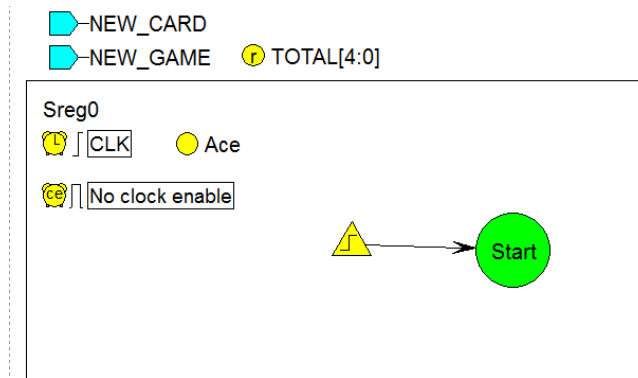


Lub



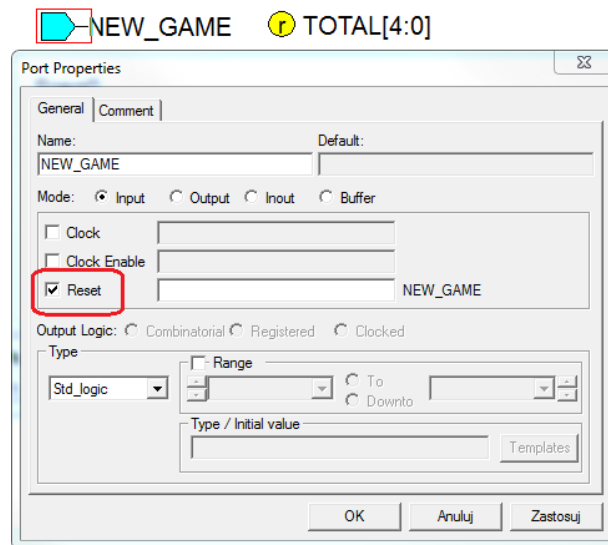
Rys. 30. Edycja nazwy stanu.

- Kliknij na pasku ikonę skrótów „Reset/Initial State Indicator” .
- Umieść symbol na lewo od stanu „Start”. Po pierwszym kliknięciu symbol wstawi się na kartkę i będzie „połączony” linią z kursorem myszki. Kliknij na stanie „Start” aby utworzyć przejście między symbolem resetu a stanem „Start” (rysunek poniżej).



Rys. 31. Schemat grafu z wstawionym symbolem resetu.

- f) Kliknij prawym przyciskiem myszki port „NEW\_GAME” i wybierz „Properties...”. W oknie właściwości zaznacz opcję „Reset” aby wskazać że ten port może być źródłem sygnału reset.

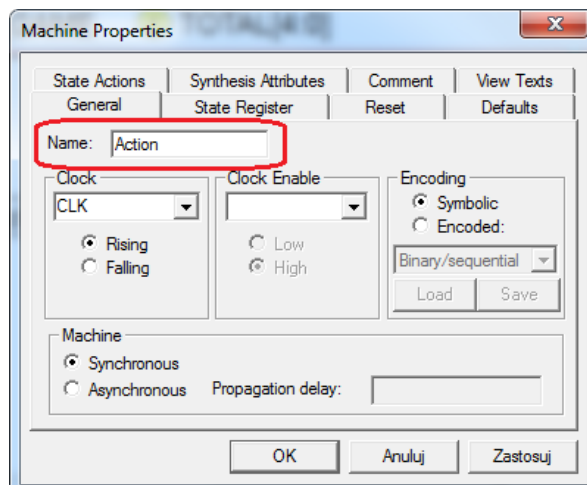


Rys. 32. Skonfigurowanie sygnału NEW\_GAME jako źródła sygnału reset.

Po zatwierdzeniu zmian, symbol portu ulegnie zmianie.

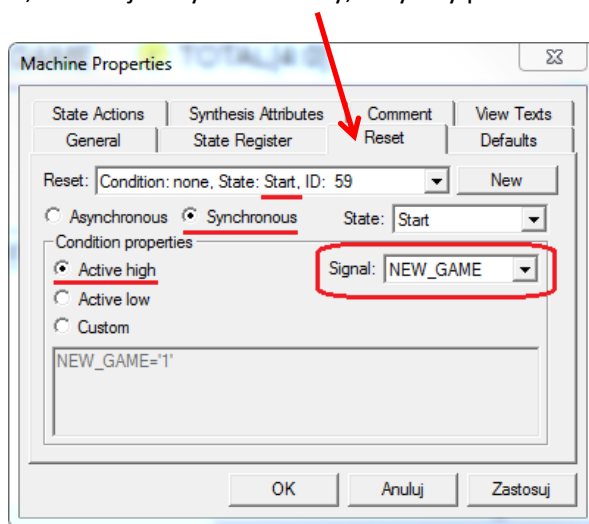


- g) Kliknij prawym przyciskiem myszy na białym polu schematu grafu (na przykład obok stanu „Start”) i wybierz „Properties...”. Otworzy się okno „Machine Properties”, które umożliwi konfigurację globalnych ustawień maszyny stanów.
- h) W oknie „Machine Properties” zmień pole Name z „Sreg0” → „Action”. W ten sposób zmienia się nazwę sygnału w generowanym kodzie VHDL, który przechowuje informację w którym stanie automat aktualnie się znajduje. (Nie zamykaj jeszcze okna ustawień).



Rys. 33. Okno globalnych ustawień maszyny stanów.

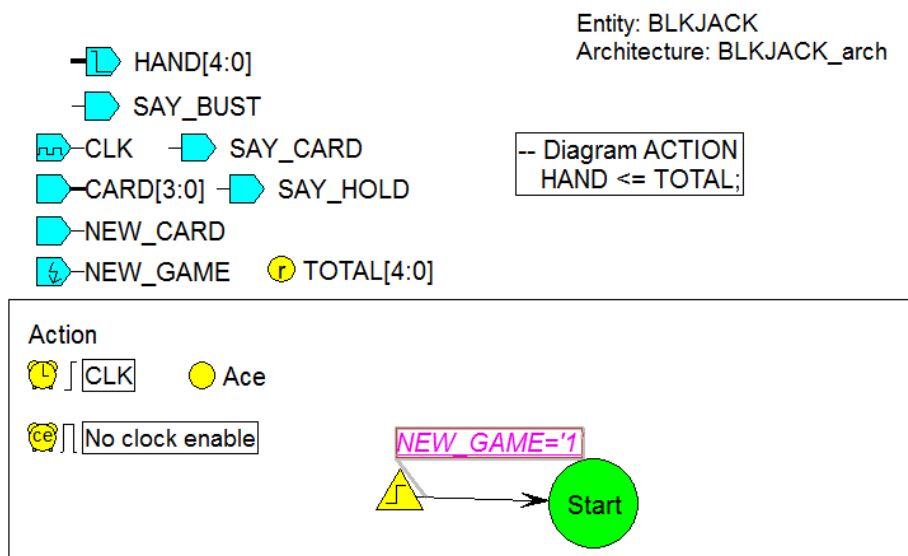
- i) W oknie „Machine Properties” zmień zakładkę na „Reset” i ustaw pole „Signal” na „NEW\_GAME”. Upewnij się że na rozwijanej liście jest stan „Start”, a reset jest synchroniczny, aktywny poziomem wysokim.



Rys. 34. Ustawienia resetu automatu.

Poprawne wykonanie tego podpunktu sprawi że automat przejdzie do stanu „Start” gdy przy narastającym zboczcu sygnału zegarowego sygnał NEW\_GAME będzie miał poziom wysoki.

- j) Zatwierdź wszelkie zmiany w oknie „Machine Properties” klikając „OK”. Graf automatu powinien wyglądać jak na rysunku poniżej.



Rys. 35. Widok grafu automatu.

Żółty trójkąt – symbol resetu – nie jest stanem. Jest to graficzna reprezentacja przejścia do pierwszego stanu (po resecie), oraz informacja jaki warunek musi zostać spełniony by zresetować automat. Po spełnieniu warunku  $NEW\_GAME=1$ , automat przejdzie do stanu „Start” niezależnie od tego w którym stanie się znajduje.

### 23. Dołożenie akcji wykonywanych tuż po resecie.

Jak tylko pojawi się aktywny sygnał resetu, automat przejdzie do stanu „Start”. Aby zdefiniować akcje wykonywane tuż po resecie (np. inicjalizacja wartości zmiennych/sygnałów), musimy zdefiniować „entry action” (akcja wejścia). Taka akcja jest automatycznie zaczepiana w górnej części symbolu stanu. Kod VHDL podany w „entry action” wykona się jednorazowo w momencie gdy automat przechodzi do tego stanu. W zależności od potrzeb projektanta, „entry action” może zostać zdefiniowane dla dowolnego stanu automatu.

- Wybierz z głównego menu programu FSM -> Action -> Entry.
- Umieść dużą czarną kropkę na symbolu stanu Start i kliknij lewym przyciskiem myszy. Otworzy się pole tekstowe do wpisania kodu VHDL dla „entry action”.


- c) W polu tekstowym umieść tekst:  
`TOTAL <= "00000";`  
 W efekcie, po wejściu do stanu „Start” zmienna TOTAL przechowująca informację o sumie wag posiadanych kart ulegnie wyzerowaniu.
- d) Kliknij poza polem tekstowym aby zakończyć edycję. Upewnij się że twój graf wygląda podobnie jak na rysunku poniżej. W szczególności sprawdź wygląd cudzysłowów – mogły ulec zmianie przy kopiowaniu tekstu z pliku pdf instrukcji. Jeżeli musisz dokonać poprawek, to dwukrotnie kliknij na kodzie VHDL.



Rys. 36. Akcja podejmowana przy wejściu do stanu „Start”.

#### 24. Dołożenie stanu zażądania nowej karty.


Po zresetowaniu, automat powinien zażądać nowej karty. Czynność tę zrealizujemy dodając nowy/kolejny stan o nazwie „Hit\_me”.

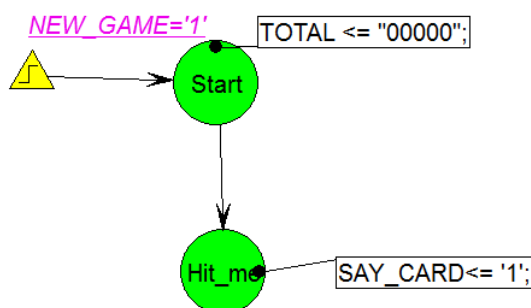
- a) Na grafie, pod stanem „Start”, dołóż symbol nowego stanu o nazwie „Hit\_me”. W tym celu postępuj podobnie jak w podpunktach 22a) – 22c).
- b) Dołóż przejście ze stanu „Start” do stanu „Hit\_me”. W tym celu kliknij na pasku ikonę skrótu Transition . Kliknij na symbolu stanu „Start” a następnie na symbolu stanu „Hit\_me”. Powinna utworzyć się strzałka skierowana od „Start” do „Hit\_me”. Kliknij prawym przyciskiem myszy by wyłączyć dodawanie kolejnych przejść.

Przejście ze stanu „Start” do stanu „Hit\_me” jest bezwarunkowe i zostanie wykonane automatycznie przy kolejnym takcie zegara (automat będzie w stanie „Start” przez jeden takt zegara po czym przy kolejnym zboczu sygnału zegarowego, automat bezwarunkowo przejdzie do stanu „Hit\_me”). Jednakże pobranie kolejnej karty powinno nastąpić wyłącznie gdy sygnał SAY\_CARD jest równy 1.

Nowa karta będzie odbierana wyłącznie gdy automat znajdzie się w stanie „Hit\_me” i sygnał SAY\_CARD (wyjściowy) będzie miał poziom logiczny wysoki. Z tego powodu sygnał SAY\_CARD musi być ustawiony przez cały czas przebywania automatu w stanie „Hit\_me”. Aby to zrealizować musimy użyć „state action” (akcja stanu) zamiast „entry action”.

„State action” jest wykonywana w każdym cyklu zegara o ile automat pozostaje w danym stanie. „Entry action” jest wykonywana jednorazowo w momencie wejścia do danego stanu. „Exit action” (akcja wyjścia) jest wykonywana w momencie gdy automat opuszcza dany stan.

- c) Wybierz z menu głównego programu FSM -> Action -> State lub użyj ikony skrótu na pasku .
- d) Ustaw wiszącą, dużą kropkę na stanie „Hit\_me” i kliknij lewym przyciskiem myszy.
- e) W powstałym polu tekstowym wpisz:  
`SAY_CARD <= '1';`
- f) Kliknij prawym przyciskiem myszy aby wyłączyć dodawanie „State action” dla innych stanów. W tym momencie graf powinien wyglądać jak na rysunku poniżej.

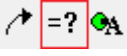


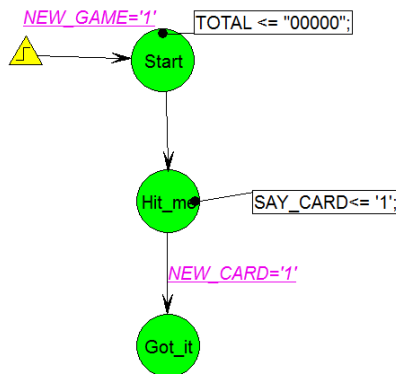
Rys. 37. Graf maszyny stanów.



## 25. Odebranie nowej karty przez automat.

Po zażądaniu nowej karty, automat powinien poczekać na jej dostarczenie. Pojawienie się nowej karty dla automatu jest sygnalizowane przez zewnętrzny układ ustawieniem poziomu wysokiego na sygnale NEW\_CARD (wejściowy).

- Na grafie, pod stanem „Hit\_me”, dołóż symbol nowego stanu o nazwie „Got\_it”. W tym celu postępuj podobnie jak w podpunktach 22a) – 22c).
- Dołóż przejście ze stanu „Hit\_me” do stanu „Got\_it”. Postępuj podobnie jak w 24b).
- Kliknij na pasku ikonę skrót Condition . Kliknij na przejściu ze stanu „Hit\_me” do stanu „Got\_it”.
- W powstałym polu tekstowym wpisz (bez średnika na końcu):  
`NEW_CARD=' 1 '`
- Kliknij prawym przyciskiem myszy poza polem tekstowym aby zakończyć edycję. Rysunek grafu poniżej.



Rys. 38. Graf maszyny stanów.

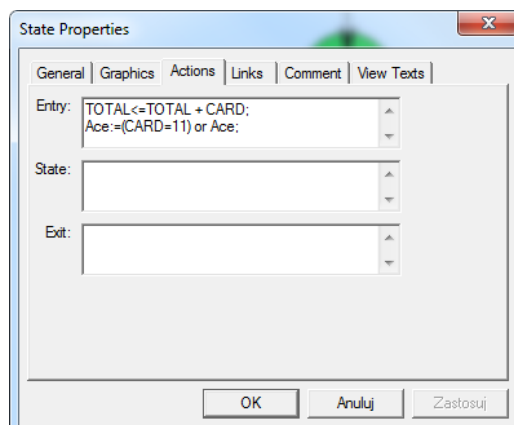
Maszyna stanów żąda nowej karty w stanie „Hit\_me” poprzez ustawienie poziomu wysokiego na wyjściu SAY\_CARD. Aby maszyna stanów przeszła do stanu „Got\_it”, na wejściu NEW\_CARD musi się pojawić poziom wysoki.

Wyjście SAY\_CARD wraca do poziomu niskiego gdy maszyna stanów opuści stan „Hit\_me”. Dzieje się tak, gdyż w opcjach portu SAY\_CARD ustawiliśmy wartość domyślną równą 0.

Wszystkie akcje są wykonywane przy zboczu narastającym sygnału zegarowego (można to zmienić w „Machine properties”).

Po otrzymaniu nowej karty należy zaktualizować wynik gry (sumę wag posiadanych kart). Ponadto należy sprawdzić czy otrzymana karta to Ace (o wadze 11 lub 1). Jeżeli otrzymana karta to Ace należy ustawić zmienną ACE (została zdefiniowana na grafie w podpunkcie 19b). Czynności te będą wykonywane jednorazowo w momencie wejścia do stanu „Got\_it”. Tym razem zostanie pokazany drugi sposób definiowania „entry action”.

- Kliknij prawym przyciskiem myszy na symbolu stanu „Got\_it” i wybierz „Properties...”.
- W otwartym oknie zmień zakładkę na „Actions”.
- W polu „Entry” wklej poniższy kod:  
`TOTAL<=TOTAL + CARD;`  
`Ace:=(CARD=11) or Ace;`

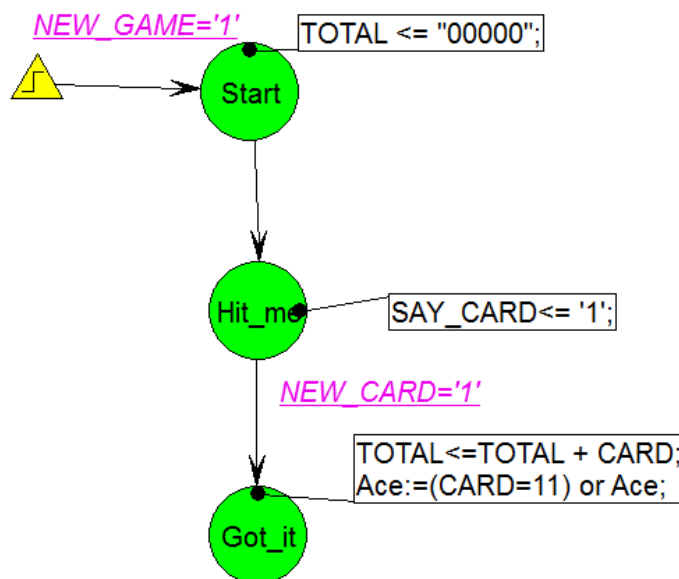


Rys. 39. Właściwości stanu „Got\_it”.

Powyższy kod zwiększa wartość sygnału wewnętrznego TOTAL o wagę odebranej karty. Do zmiennej Ace wpisana zostanie jej poprzednia wartość lub wynik testu czy nowa karta ma wagę 11 (waga karty Ace).

i) Kliknij „OK” aby zatwierdzić zmiany. Wygląd grafu poniżej.

Gdyby kod z podpunktu h) został wklejony w polu „State” (zdefiniowanie „State action”), to wynik gry zwiększałby się z każdym taktem zegara tak długo jak automat pozostawałby w stanie „Got\_it”.

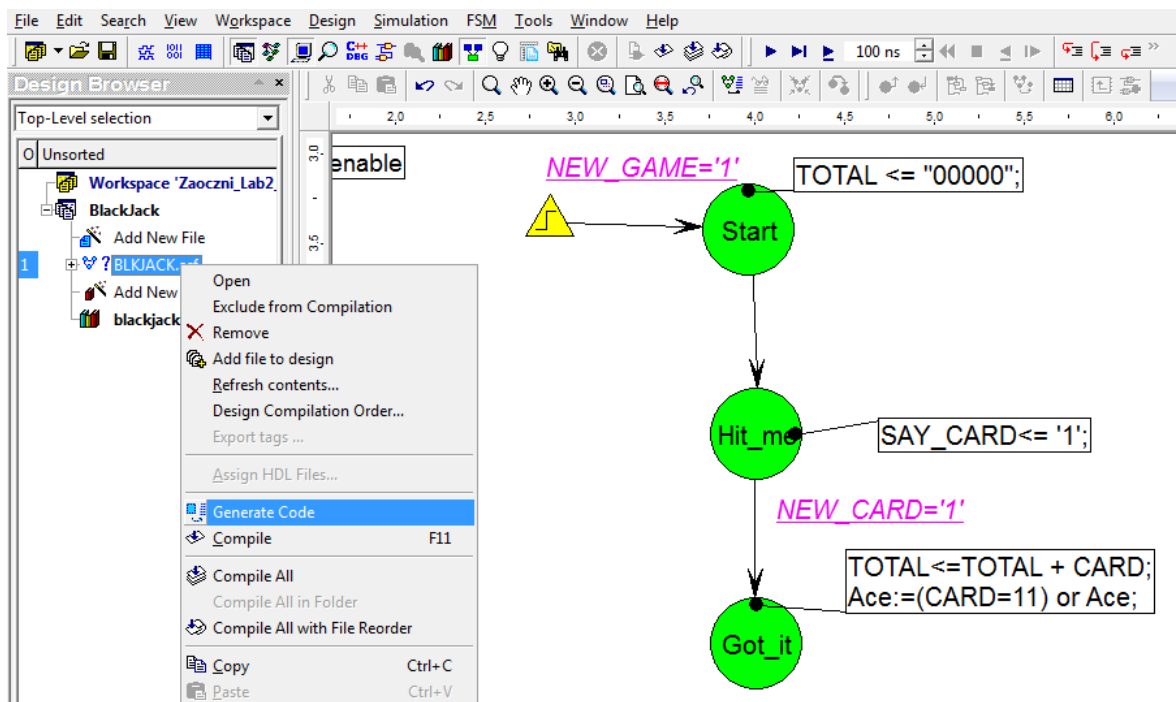


Rys. 40. Wygląd grafu maszyny stanów.

## 26. Generowanie kodu VHDL

Na tym etapie fragment budowanego automatu został już zrealizowany. Zanim rozbudujemy projekt, sprawdzimy jego poprawność (czy się kompiluje) oraz przeprowadzimy analizę wygenerowanego kodu vhd. Proces generowania kodu HDL sprawdza poprawność narysowanego grafu ale nie sprawdza instrukcji wpisanych przez użytkownika (np. „entry action”). Pełna weryfikacja poprawności kodu vhd jest wykonywana w momencie jego kompilacji.

a) W „Design Browser” kliknij prawym przyciskiem myszy na pliku „BLKJACK.asf” i wybierz opcję „Generate Code”.



Rys. 41. Generowanie kodu VHDL na podstawie grafu automatu.

b) Z menu głównego programu wybierz FSM -> View HDL Code.

Automatycznie wygenerowany kod składa się z kilku sekcji:

**- Wybór bibliotek (zawsze na początku pliku)**

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

**- Deklaracji entity wraz z deklaracją portów zdefiniowanych na diagramie automatu**

```
entity BLKJACK is
    port (
        CARD: in STD_LOGIC_VECTOR (3 downto 0);
        CLK: in STD_LOGIC;
        NEW_CARD: in STD_LOGIC;
        NEW_GAME: in STD_LOGIC;
        HAND: out STD_LOGIC_VECTOR (4 downto 0);
        SAY_BUST: out STD_LOGIC;
        SAY_CARD: out STD_LOGIC;
        SAY_HOLD: out STD_LOGIC);
end BLKJACK;
```

**- Globalnych deklaracji typów wyliczeniowych, sygnałów i innych obiektów wspólnych dla wszystkich maszyn stanu**

```
architecture BLKJACK_arch of BLKJACK is
-- diagram signals declarations
signal TOTAL: STD_LOGIC_VECTOR (4 downto 0);
-- SYMBOLIC ENCODED state machine: Action
type Action_type is (
    Start, Hit_me, Got_it
);
signal Action, NextState_Action: Action_type;
-- attribute enum_encoding of Action_type: type is ... -- enum_encoding attribute
-- is not supported for symbolic encoding
attribute fsm_extract: string;
attribute fsm_extract of Action: signal is "yes";
attribute fsm_fftype: string;
attribute fsm_fftype of Action: signal is "d";
attribute fsm_style: string;
attribute fsm_style of Action: signal is "bram";
attribute safe_implementation: string;
attribute safe_implementation of Action: signal is "no";
-- Declarations of pre-registered internal signals
signal int_HAND, next_HAND: STD_LOGIC_VECTOR (4 downto 0);
signal next_TOTAL: STD_LOGIC_VECTOR (4 downto 0);
begin
-- concurrent signals assignments
-- Diagram ACTION
    HAND <= TOTAL;
```

**- Sekcji deklaracji dla maszyny (obiekty lokalne dla procesu)**

```
-----
-- Machine: Action
-----
-- Next State Logic (combinatorial)
-----
Action_NextState: process (CARD, NEW_CARD, TOTAL, Action)
-- machine variables declarations
variable Ace: BOOLEAN;
```

- Opis przejść pomiędzy stanami oraz akcji podejmowanych w danym stanie

```
begin
    NextState_Action <= Action;
    -- Set default values for outputs and signals
    SAY_CARD <= '0';
    next_TOTAL <= TOTAL;
    SAY_BUST <= '0';
    SAY_HOLD <= '0';
    case Action is
        when Start =>
            NextState_Action <= Hit_me;
        when Hit_me =>
            SAY_CARD<= '1';
            if NEW_CARD='1' then
                NextState_Action <= Got_it;
                next_TOTAL <= TOTAL + CARD;
                Ace := (CARD=11) or Ace;
            end if;
    --vhdl_cover_off
        when others =>
            null;
    --vhdl_cover_on
    end case;
end process;
```

- sekcja rejestru stanu – uaktualnia wartość zmiennej przechowującej stan automatu oraz definiuje domyślny stan po resecie

```
-----
-- Current State Logic (sequential)
-----
```

```
Action_CurrentState: process (CLK)
begin
    if CLK'event and CLK = '1' then
        if NEW_GAME='1' then
            Action <= Start;
        else
            Action <= NextState_Action;
        end if;
    end if;
end process;
```

- sekcja wyjść rejestrowanych (wyjściowy rejestr PIPO)

```
-----
-- Registered Outputs Logic
-----
```

```
Action_RegOutput: process (CLK)
begin
    if CLK'event and CLK = '1' then
        if NEW_GAME='1' then
            TOTAL <= "00000";
        else
            TOTAL <= next_TOTAL;
        end if;
    end if;
end process;

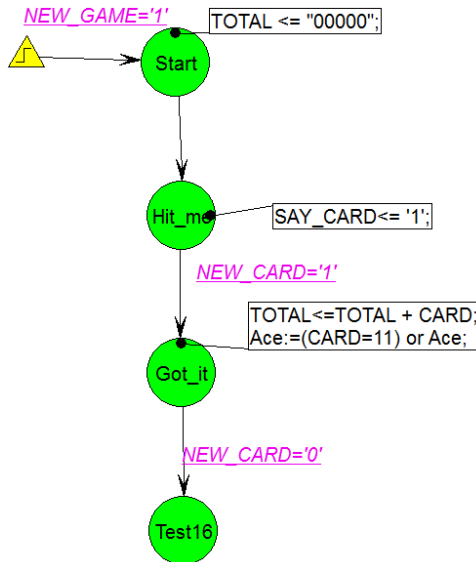
end BLKJACK_arch;
```

## 27. Analiza wyniku po odebraniu nowej karty

Po otrzymaniu nowej karty i zaktualizowaniu wyniku gry należy przeprowadzić analizę wyniku. Jeżeli wynik gry jest mniejszy od 17, to automat powinien zażądać kolejnej karty, czyli wrócić do stanu „Hit\_me”.

- Na rysunku grafu, pod stanem „Got\_it” dołóż kolejny stan i nazwij go „Test16”.
- Dołóż przejście ze stanu „Got\_it” do stanu „Test16”.
- Utwórz warunek (condition) dla tego przejścia (bez średnika na końcu):  
`NEW_CARD = '0'`

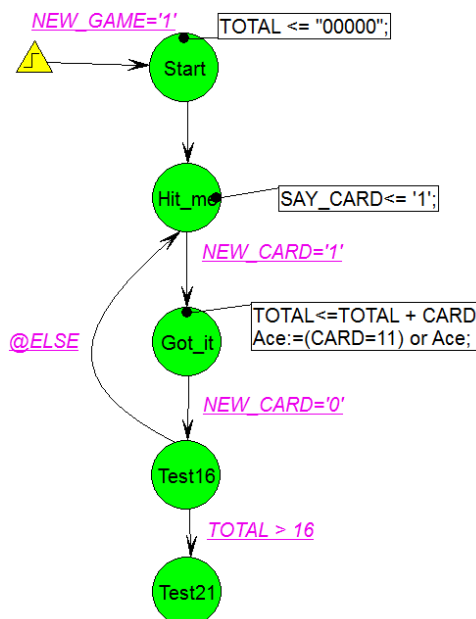
Dołożenie tego warunku zabezpiecza automat przed pobraniem tej samej karty dwa razy.



Rys. 42. Diagram maszyny stanów.

- Na rysunku grafu, pod stanem „Test16” dołóż kolejny stan i nazwij go „Test21”.
- Dołóż przejście ze stanu „Test16” do stanu „Test21”. Nadaj mu warunek (bez średnika):  
`TOTAL > 16`
- Dołóż przejście ze stanu „Test16” do stanu „Hit\_me”. Nadaj mu warunek:  
`@ELSE`

Przejście @ELSE jest wykonywane gdy żaden z pozostałych warunków nie jest spełniony. Inaczej mówiąc, gdy wynik gry jest mniejszy lub równy 16, to automat przejdzie do stanu „Hit\_me” i zażąda kolejnej karty.



Rys. 43. Diagram maszyny stanów.

## 28. Analiza całkowitego wyniku gry

Gdy automat dotrze do stanu „Test21”, całkowity wynik gry wynosi 17 lub więcej. Teraz należy sprawdzić czy wynik przekroczył 21. Jeżeli nie, to automat powinien ustawić flagę SAY\_HOLD (sygnał wyjściowy). Jeżeli wynik przekracza 21, to należy ustawić flagę SAY\_BUST (sygnał wyjściowy), która informuje o przegraniu gry. Zanim jednak SAY\_BUST zostanie ustawiony należy sprawdzić czy któraś z pobranych kart nie jest kartą Ace. Jeżeli automat posiada kartę Ace, to może obniżyć całkowity wynik o 10 (Ace może być liczone jako 11 lub 1) i powrócić do gry.

a) Pod stanem „Test21” dodaj dwa nowe stany. Stanowi po lewej nadaj nazwę „Bust”, stanowi po prawej nadaj nazwę „Hold”.

b) Wstaw przejście ze stanu „Test21” do stanu „Hold” i nadaj mu warunek:

```
TOTAL<22
```


c) Dla stanu „Hold” zdefiniuj „state action”:

```
SAY_HOLD <= '1';
```

d) Wstaw przejście ze stanu „Test21” do stanu „Test16” i nadaj mu warunek:

```
Ace
```

Ace jest zmienną typu boolean i nie trzeba wykonywać testu logicznego aby sprawdzić jej wartość.

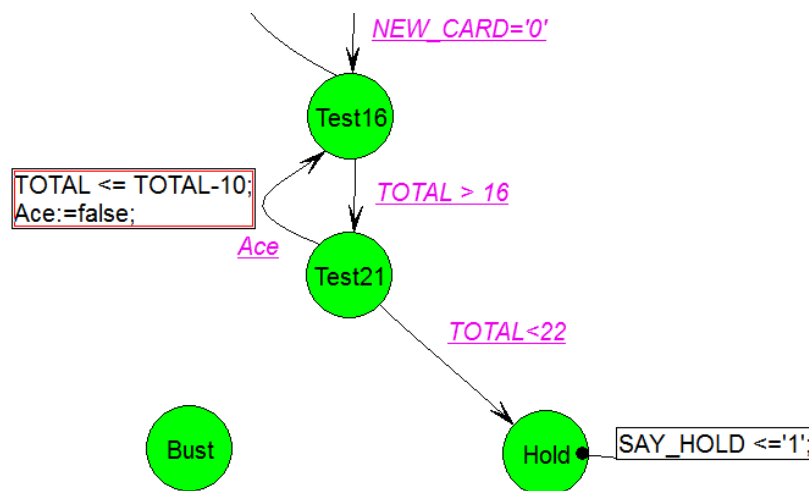
e) Wybierz z paska ikonę skrótu „Transition Action” .

f) Kliknij na przejściu z „Test21” do „Test16” (obok Ace) i wpisz poniższy kod:

```
TOTAL <= TOTAL-10;
```

```
Ace:=false;
```

„Transition Action” (akcja przejścia) jest wykonywana w momencie przechodzenia z jednego stanu do drugiego i pozwala zmniejszyć liczbę stanów w automacie. Gdybyśmy z niej nie skorzystali, należałoby wprowadzić pomocniczy stan, w którym zmniejszylibyśmy wynik TOTAL o 10 punktów.



Rys. 44. Fragment rysowanego grafu.

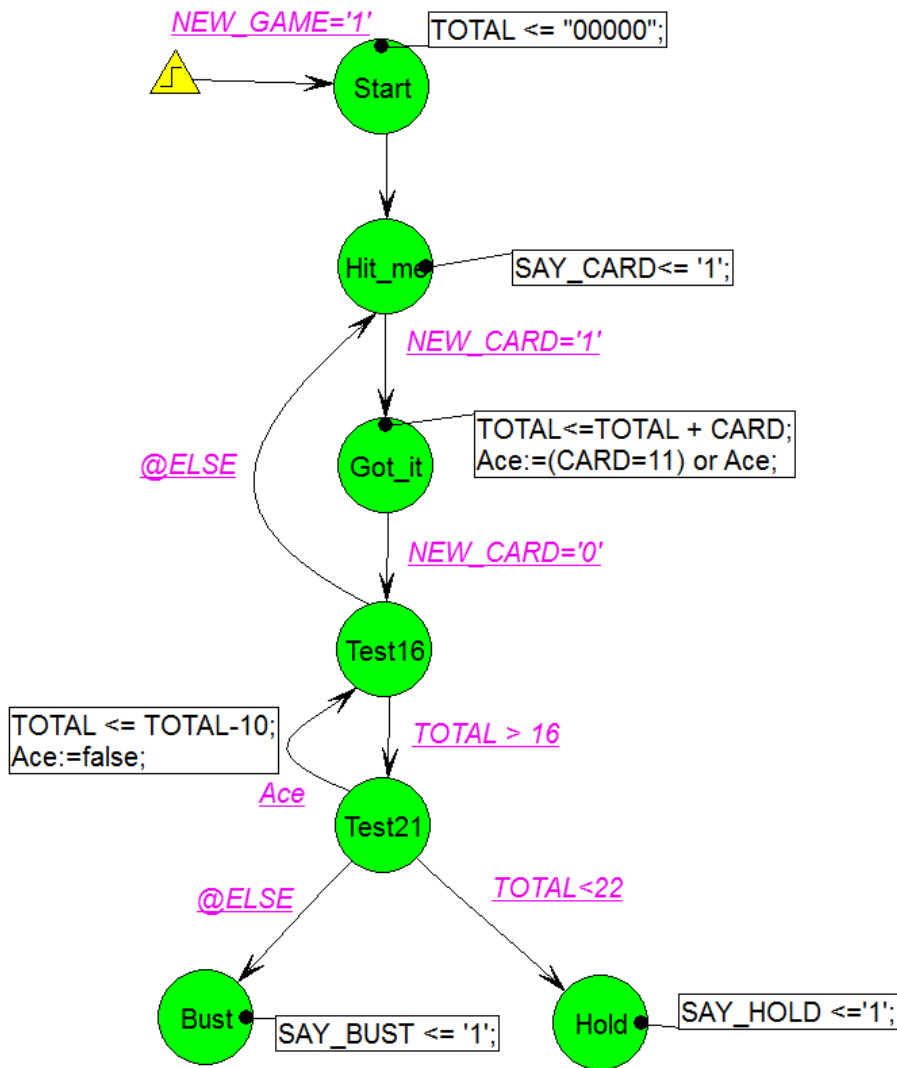
g) Wstaw przejście ze stanu „Test21” do stanu „Bust” i nadaj mu warunek:

```
@ELSE
```

h) Dla stanu „Bust” zdefiniuj „state action”:

```
SAY_BUST <= '1';
```

Końcowy wygląd rysowanego diagramu maszyny stanów przedstawiono na rysunku poniżej.

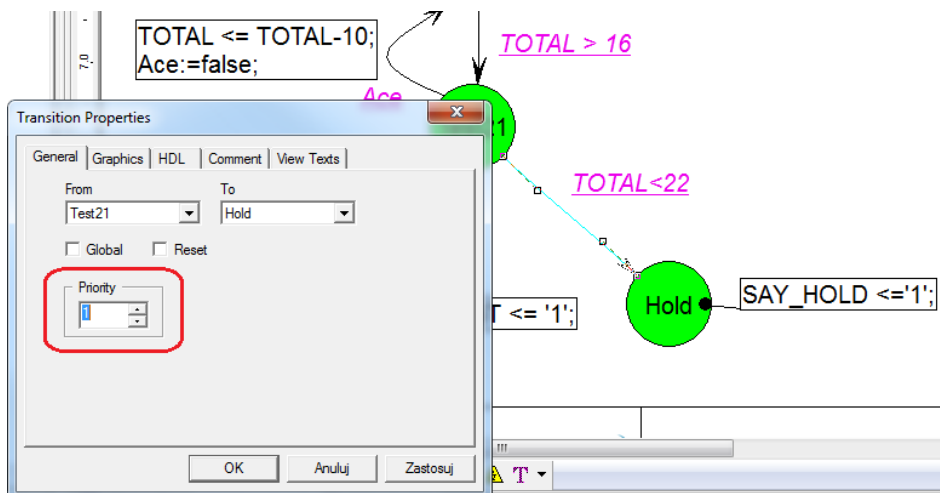


Rys. 45. Graf zaprojektowanej maszyny stanów.

## 29. Nadanie priorytetu warunkom

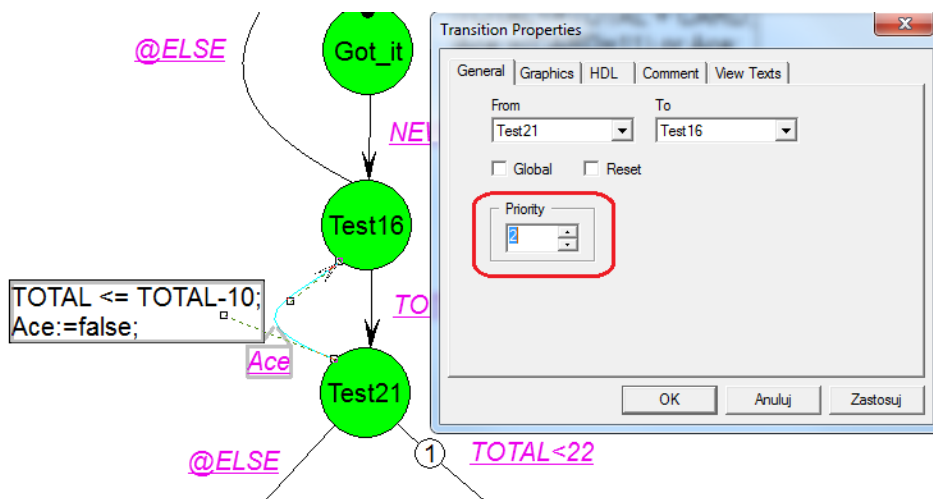
Gdy automat znajdzie się w stanie „Test21” zarówno warunek  $TOTAL < 22$  oraz Ace mogą być spełnione. W takim przypadku działanie automatu będzie zależało od kolejności w jakiej warunki będą sprawdzane. Aby automat zachowywał się deterministycznie, należy przypisać różne priorytety tym warunkom. Ponieważ nasz automat powinien w pierwszej kolejności sprawdzić czy wynik gry jest mniejszy od 22, to przejściu z „Test21” do „Hold” przypiszemy priorytet 1. Natomiast przejście z „Test21” do „Test16” otrzyma priorytet 2.

- Kliknij prawym przyciskiem myszy przejście z „Test21” do „Hold” i wybierz „Properties...”.
- W otwartym oknie ustaw priorytet równy 1. Kliknij „OK” aby zatwierdzić zmiany.



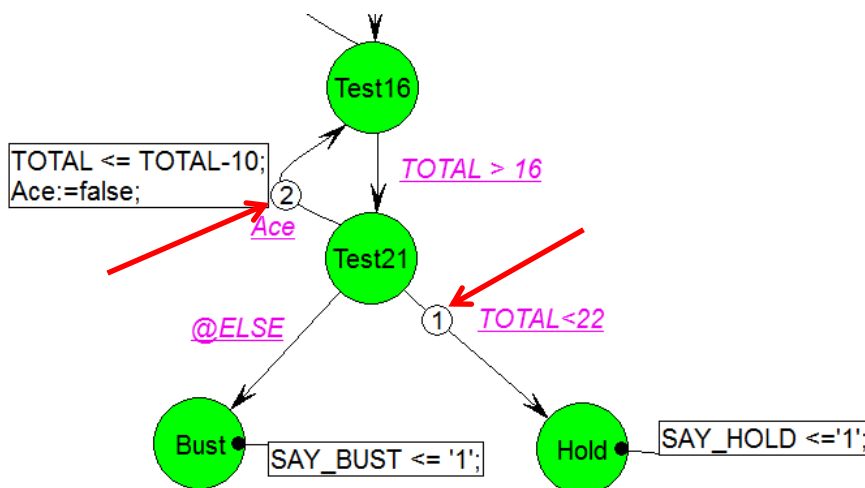
Rys. 46. Nadanie priorytetu przejściu pomiędzy stanami automatu „Test21” → „Hold”.

- c) Kliknij prawym przyciskiem myszy przejście z „Test21” do „Test16” i wybierz „Properties...”.
- d) W otwartym oknie ustaw priorytet równy 2. Kliknij „OK” aby zatwierdzić zmiany.



Rys. 47. Nadanie priorytetu przejściu pomiędzy stanami automatu „Test21” → „Test16”.

Przejścia bez nadanego priorytetu będą sprawdzane/wykonywane jako ostatnie.



Rys.48. Fragment diagramu automatu z przypisanymi priorytetami do przejść.

### 30. Wybór sposobu kodowania stanów automatu

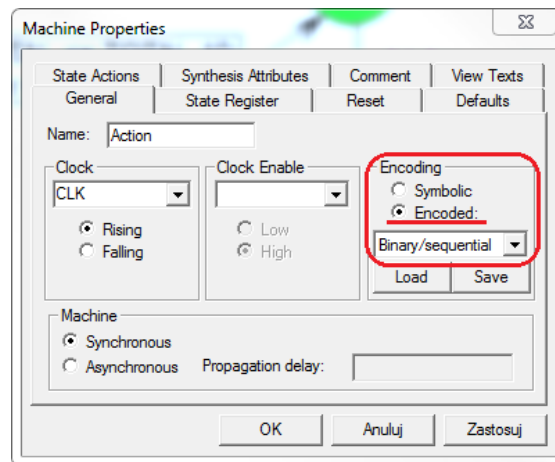
W momencie generowania kodu HDL na podstawie narysowanego grafu poszczególne stany automatu są kodowane jako wartość w rejestrze stanu (kilka przerzutników, których wyjścia stanowią sygnał wewnętrzny o nazwie Action). Każdemu stanowi przypisana jest inna wartość w rejestrze. Jednakże, gdy automat zmienia stan to w przypisaniach współbieżnych mogą wystąpić hazardy. Może się zdarzyć, że z powodu hazardów i opóźnień automat przejdzie do błędnego stanu. Z tego powodu, kodowanie z „gorącą jedynką” (one hot) jest rekomendowane dla wszystkich układów FPGA. W kodowaniu tym jest tyle przerzutników ile stanów automatu. W danej chwili tylko jeden przerzutnik ma wpisaną logiczną jedynkę a pozostałe są wyzerowane. Przy takim kodowaniu minimalizuje się liczbę przerzutników jednocześnie zmieniających swoją wartość. Jednakże kodowanie to ma wadę – zużywa więcej zasobów układu programowalnego (jeden przerzutnik na każdy stan automatu). Dlatego kodowanie one-hot nie jest polecane dla automatów realizowanych w CPLD, gdyż mają one mało przerzutników. Edytor diagramów stanu w Active-HDL pozwala wybrać sposób kodowania stanów automatu. Można wybrać pomiędzy kodowaniem one-hot a kodowaniem binarnym. Jak również można zdefiniować dowolny sposób kodowania stanów ręcznie dopisując kody do każdego stanu automatu.

W tym projekcie kodowanie binarne zostanie wybrane by ułatwić analizę układu w symulatorze.

- a) Kliknij prawym przyciskiem myszy na pustym polu diagramu stanów (np. obok stanu „Bust”) i wybierz „Properties...”. Otworzy się okno „Machine Properties”.



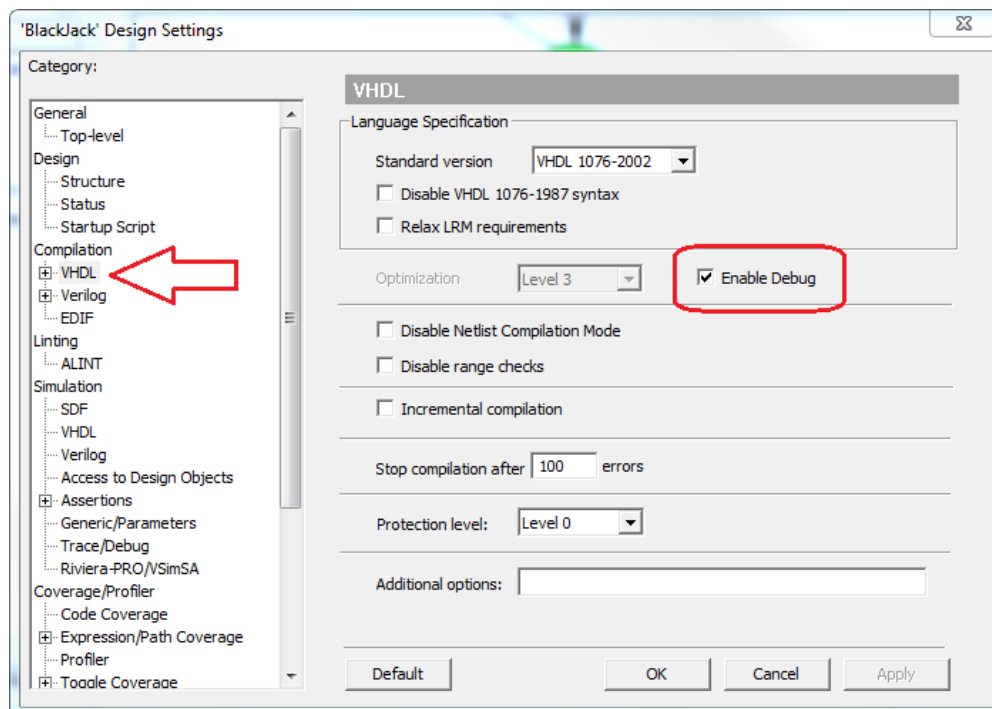
- b) W zakładce „General” zaznacz opcję „Encoded” i z rozwijanej listy wybierz „Binary/sequential”.  
(W przypadku gdy kodowanie „Symbolic” jest wybrane, to sposób kodowania wybiera narzędzie do syntezy.)



Rys. 49. Okno globalnych ustawień automatu. Wybór sposobu kodowania stanów.

### 31. Kompilacja projektu.

- a) Wybierz z głównego menu programu Design → Settings. Wybierz w lewej części okna Compilation / VHDL. Upewnij się że opcja „Enable Debug” jest zaznaczona.



Rys. 50. Ustawienie by kompilator wygenerował dane do debugowania.

- b) Zapisz plik a następnie skompiluj go. Poprawną kompilację pliku symbolizuje ikona  obok jego nazwy.

Active-HDL automatycznie wygeneruje kod vhdl na podstawie narysowanego diagramu i dokona jego kompilacji. Cały proces **może zająć dłuższą chwilę**.

Metoda I:

Kliknij na pasku przycisk skrótu



Metoda II:

Wybierz z menu Design -> Compile.

Metoda III:

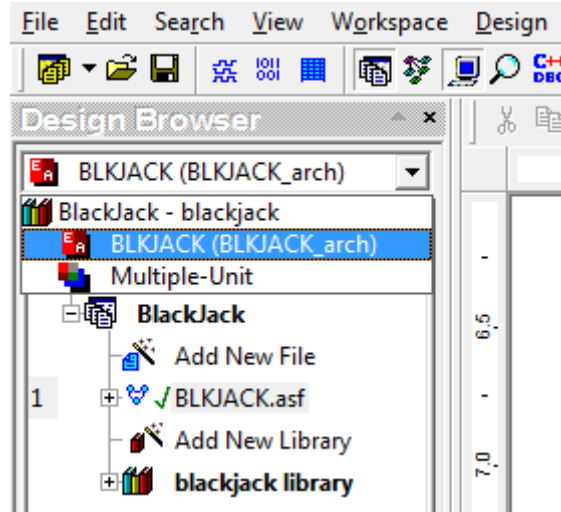
Kliknij prawym przyciskiem myszy na BLKJACK.asf w Design Browser i wybierz Compile.

## 32. Symulacja projektu

- a) Wybieramy co chcemy symulować poprzez wskazanie pary entity-architecture jako „Top-Level”. Chcemy przesymulować działanie komponentu „BLKJACK”.

Metoda I:

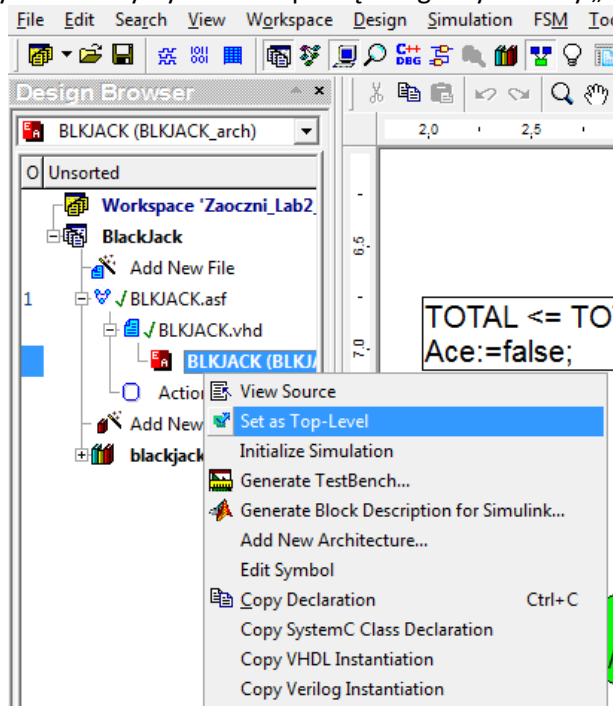
Od samej góry okna Design Browser jest rozwijana lista. Wybieramy w niej „BLKJACK (BLKJACK\_arch)”.



Rys. 51a. Wskazanie komponentu jako „Top-Level” w Design Browser.

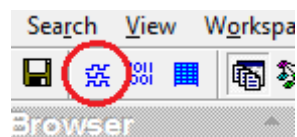
Metoda II:

W Design Browser rozwiń plusik obok BLKJACK.asf a następnie obok BLKJACK.vhd. Pojawi się w drzewie projektu kolejna pozycja którą klikamy prawym przyciskiem myszy i z menu podręcznego wybieramy „Set as Top-Level”.



Rys. 51b. Wskazanie komponentu jako „Top-Level” w Design Browser.

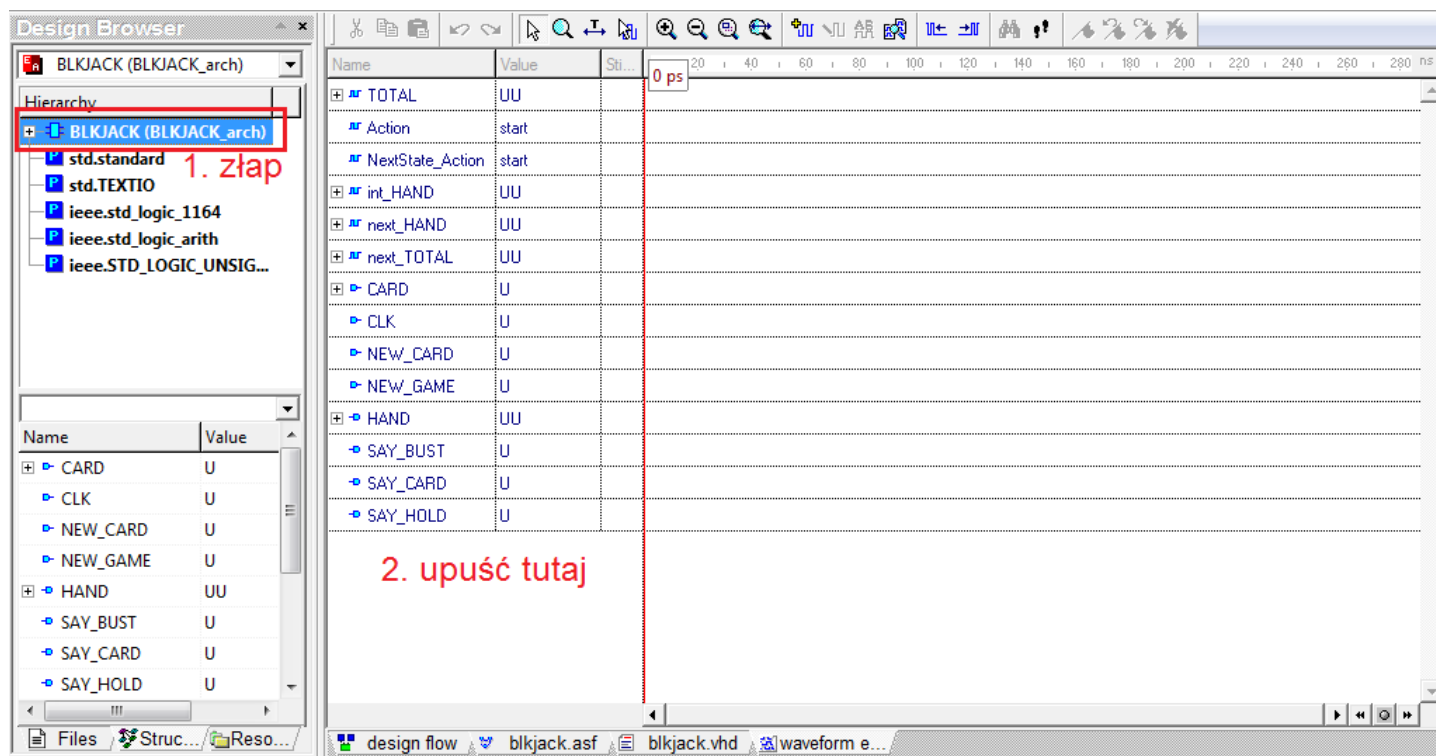
- b) Z menu głównego wybierz Simulation -> Initialize Simulation.  
c) Korzystając z ikony skrótu otwórz New Waveform.



Rys. 52. Ikona skrótu New Waveform.

W prawej części okna programu otworzy się nowa zakładka z możliwością oglądania przebiegów.

- d) W Design Browser, zakładka Structure, zaznacz „BLKJACK (BLKJACK\_arch)” i trzymając wciśnięty lewy przycisk myszy przenieś go do okna New Waveform (tak jak się przesuwa ikony na pulpicie).

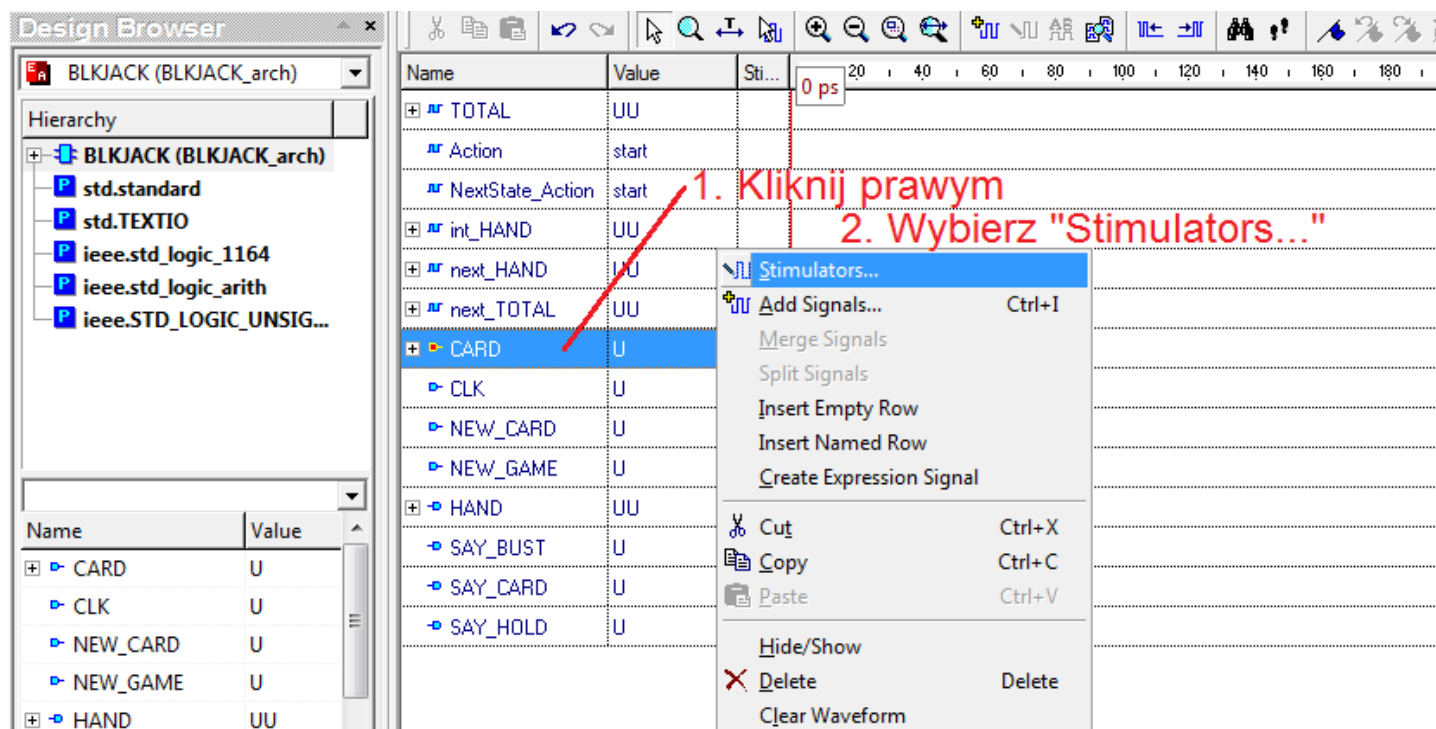


Rys. 53. Okno symulatora z wybranymi sygnałami.

Niepotrzebny sygnał można usunąć zaznaczając go i wciskając klawisz „Delete” na klawiaturze.

- e) Otwórz okno pobudzeni / wymuszeń / stymulatorów.

Kliknij prawym przyciskiem myszy sygnał o nazwie CARD. Z menu kontekstowego wybierz „Stimulators...”.



Rys. 54. Otwieranie okna wymuszeń (Stimulators).

- f) Przypisz stymulator w postaci „Formuła” do sygnałów CARD, NEW\_CARD, NEW\_GAME oraz CLK.

W tym celu zaznacz odpowiedni sygnał w „waveform editor”, wybierz typ wymuszenia „Formuła”, w polu „Enter formula” wklej kod z tabeli poniżej, kliknij przycisk „Apply”.

Nazwa sygnału	Kod wklejany w polu „Enter formula”
CARD	16#A 0, 16#A 100 ns, 16#3 170 ns, 16#B 300 ns, 16#4 400 ns
NEW_CARD	0 0, 1 80 ns -r 100 ns
NEW_GAME	1 0, 0 20 ns -r 600 ns
CLK	0 0, 1 5 ns -r 10 ns

Składnia formuły: wartość [czas {, wartość czas} [ -r okres ] ]

Nawiasy [] oznaczają element opcjonalny;

Nawiasy {} oznaczają element opcjonalny, który może być powtarzany.

Name	Value	Sti...
+	TOTAL	UU
+	Action	start
+	NextState_Action	start
+	int_HAND	UU
+	next_HAND	UU
+	next_TOTAL	UU
+	CARD	A
+	CLK	0
+	NEW_CARD	0
+	NEW_GAME	1
+	HAND	UU
+	SAY_BUST	U
+	SAY_CARD	U
+	SAY_HOLD	U

Stimulators

Signals: Hotkeys | Predefined

Set:  New Remove

Signals: 

Name	Type
<input checked="" type="checkbox"/> CARD	Formula
<input checked="" type="checkbox"/> NEW_CARD	Formula
<input checked="" type="checkbox"/> NEW_GAME	Formula
<input checked="" type="checkbox"/> CLK	Formula

Type:  Clock  Formula

Forces a waveform defined by a textual formula.

value:  time offset:  0 ns

repeat above sequence every:

Enter formula:   Accept Active-CAD format.

Display paths Save  Strength:

1. Zaznacz sygnał

2. Wybierz "Formuła"

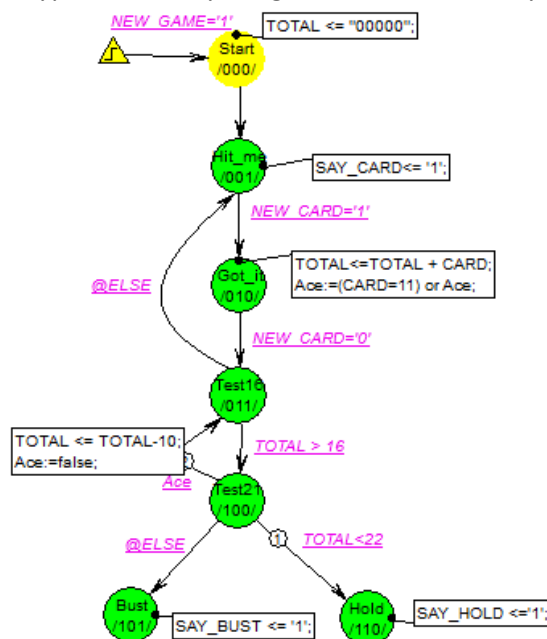
3. Wklej kod z tabeli

4. Kliknij "Apply"

Rys. 55. Nadawanie wymuszeń sygnałom.

g) Zmień zakładkę na edytor diagramu tak by oglądać graf automatu.

Edytor maszyny stanów zmienia kolor wypełnienia aktywnego stanu na kolor żółty.



Rys. 56. Graf automatu z aktywnym stanem „Start”.

h) Kliknij jeden raz ikonę skrótu „Trace Over Transition” znajdującą się na górnym pasku.

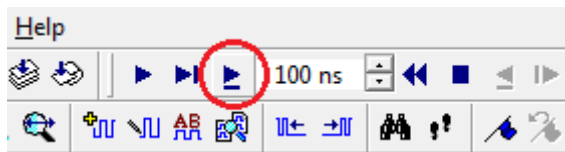


Rys. 57. Ikona „Trace Over Transition”.

„Trace Over Transition” symuluje działanie układu aż do napotkania kolejnego przejścia pomiędzy stanami. W tym momencie automat powinien znaleźć się w stanie „Hit\_me” i przy kolejnym zboczu zegara powinien ustawić sygnał SAY\_CARD.

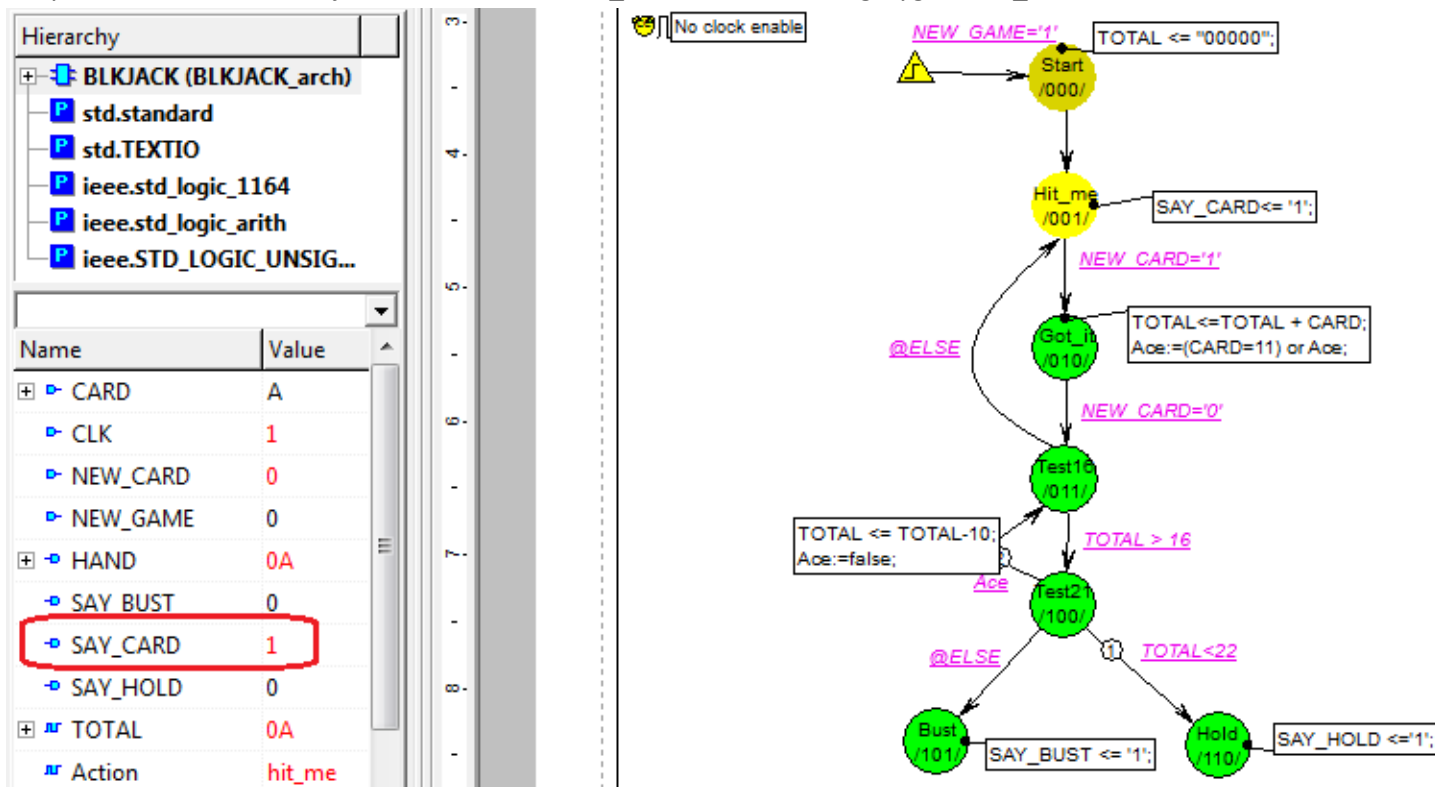
i) Kliknij jeden raz ikonę skrótu „Run For”.

Nigdy nie uruchamiaj symulacji klikając skrót „Run” (nie mamy zdefiniowanego momentu zakończenia symulacji; symulacja będzie się wykonywać „w nieskończoność” i zawiesi komputer).



Rys. 58. Przycisk skrótu „Run For”.

W tym momencie automat jest nadal w stanie „Hit\_me” ale zmianie uległ sygnał SAY\_CARD.

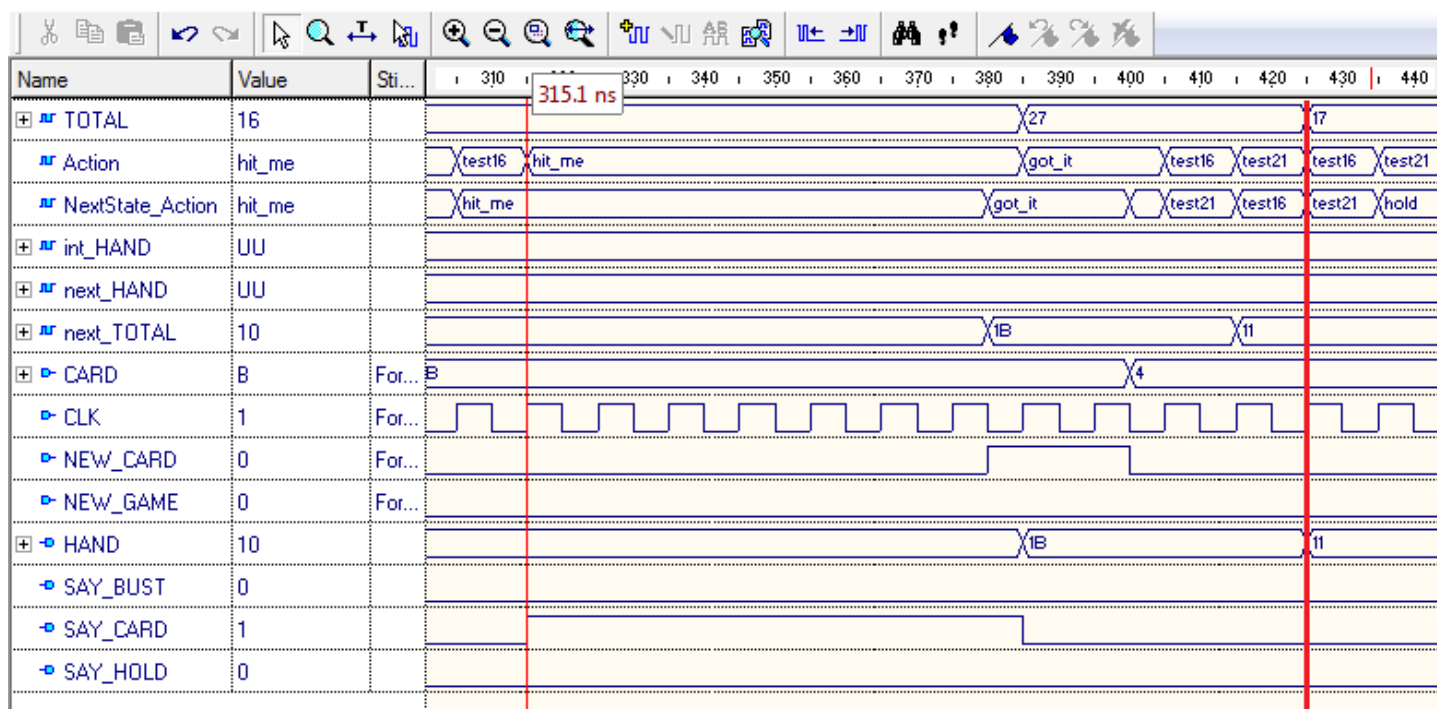


Rys. 59. Symulacja potwierdza wykonanie „state action”.

j) Kliknij jeden raz ikonę skrótu „Trace Over Transition”. Automat powinien znaleźć się teraz w stanie „Got\_it”, czyli automat odebrał nową kartę o wadze 10 (wartość A szesnastkowo).

k) Kliknij parę razy ikonę „Trace Over Transition” obserwując działanie automatu. Graf na bieżąco pokazuje przejścia pomiędzy stanami a w lewym oknie można podglądać chwilowe wartości sygnałów.

- l) Oczywiście przebieg symulacji można też oglądać na waveform. W szczególności widać tam, że w 425 ns symulacji całkowity wynik TOTAL został zmniejszony o 10 (użycie karty Ace).

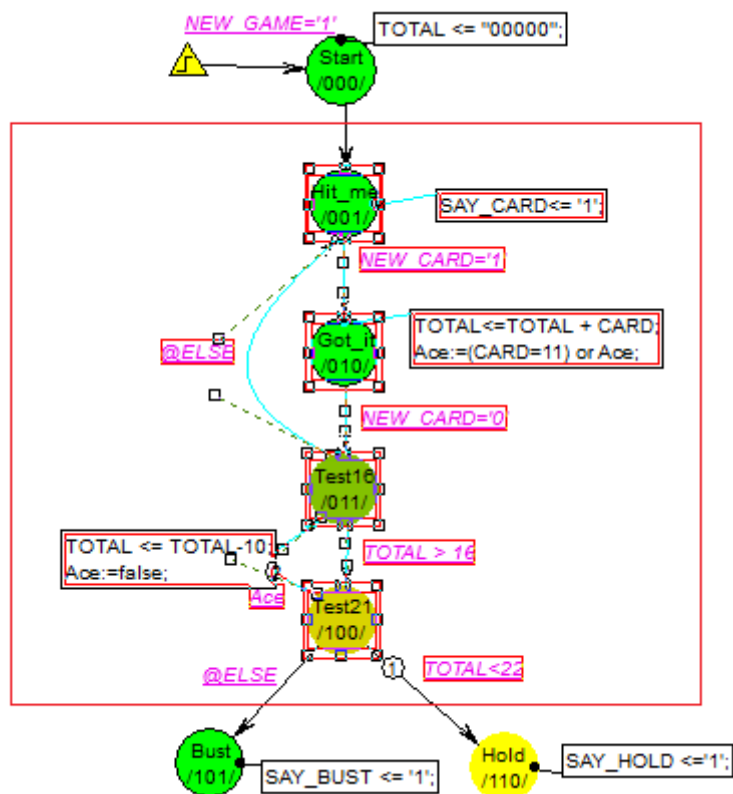


Rys. 60. Wyniki symulacji w oknie new waveform.

### 33. Tworzenie hierarchii

Edytor diagramów pozwala na rysowanie automatów z wykorzystaniem hierarchii. Zmodyfikujemy istniejący diagram aby pokazać jak stworzyć hierarchię w projekcie poprzez zamianę fragmentu grafu w stan hierarchiczny.

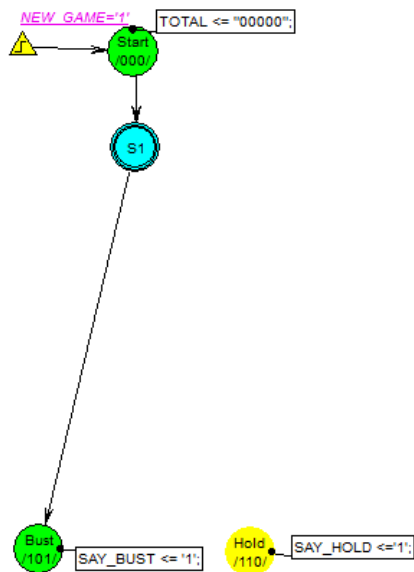
- a) Zaznacz fragment grafu, tak jak jest pokazane poniżej.



Rys. 61. Zaznaczenie fragmentu diagramu automatu w celu jego konwersji do stanu hierarchicznego.

b) Wybierz z menu głównego programu FSM → Hierarchy → Convert to Hierarchical State.

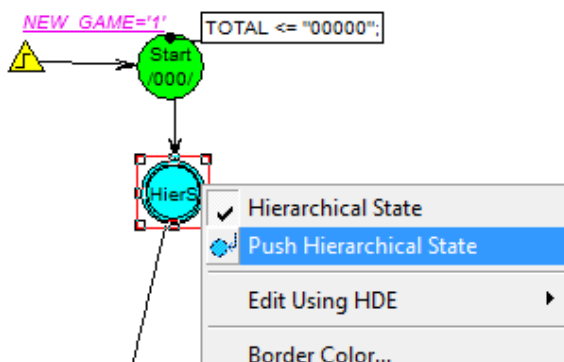
Powstanie nowy stan, zaznaczony kolorem niebieskim (informacja że jest to stan hierarchiczny).



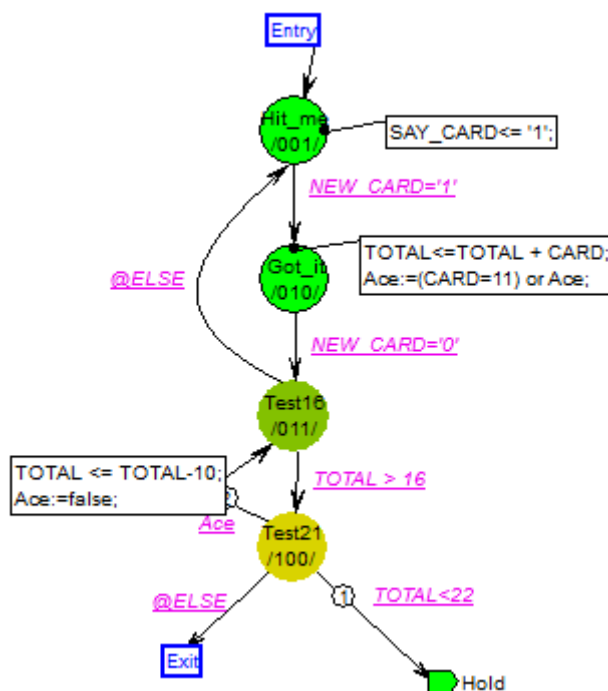
Rys. 62. Graf automatu po konwersji.

c) Kliknij prawym przyciskiem myszy na stanie „S1” i wybierz „Properties...”. Zmień nazwę stanu na „HierS”. Kliknij „OK” aby zatwierdzić zmiany.

d) Kliknij prawym przyciskiem myszy na stanie „HierS” i wybierz „Push Hierarchical State”. Otworzy się nowa karta z fragmentem grafu automatu przekonwertowanym do stanu hierarchicznego.



Rys. 63. Otwarcie poddiagramu reprezentowanego w postaci stanu hierarchicznego.



Rys. 64. Zawartość nowej karty po wykonaniu operacji „Push Hierarchical State”.

Stan „Hit\_me” jest połączony z symbolem Hierarchy Entry, natomiast stan „Test21” jest połączony z symbolem Hierarchy Exit. Hierarchy Entry oraz Hierarchy Exit są linkami do głównego grafu na wyższym poziomie hierarchii. Przejście ze stanu „Test21” do stanu „Hold” jest zrealizowane z wykorzystaniem symbolu „link”. Symbol „link” pozwala utworzyć przejście do dowolnego stanu w projekcie automatu, niezależnie czy stan ten znajduje się na tej samej karcie, czy na innej. Oczywiście stan do którego chcemy się linkować musi należeć do tego samego automatu.

- e) W głównym menu programu wybierz FSM → Hierarchy →Pop Hierarchy aby powrócić z powrotem na główną kartę.
- f) Zapisz wprowadzone zmiany i skompiluj projekt.
- g) Sprawdź w symulacji czy automat działa tak jak poprzednio.