# Cyfrowe systemy asynchroniczne (1)

*Większość współcześnie projektowanych systemów cyfrowych opiera się na dwóch zasadniczych: wszystkie sygnały są binarne oraz czas ma naturę dyskretną określoną przez kolejne takty sygnału zegarowego. Odrzucenie drugiego z założeń prowadzi systemów określanych jako asynchroniczne.*

Przyjęcie założenia o dyskretnej naturze czasu znakomicie upraszcza proces projektowania w przypadku systemów synchronicznych,. Między innymi ignorowane mogą być takie zjawiska jak hazard w układach kombinacyjnych oraz wyścigi w układach sekwencyjnych. Zjawisk tych nie można już pominąć podczas projektowania systemów asynchronicznych. Zasadniczo systemy asynchroniczne, to takie w których nie występuje globalny sygnał zegarowy, synchronizujący ich działanie.

## Zalety systemów asynchronicznych

Systemy asynchroniczne charakteryzują się pewnymi zaletami w stosunku do systemów synchronicznych. Do zalet tych można zaliczyć [1, 2]:

1. Nie występują problemy z dystrybucją globalnego sygnału zegarowego.
   Coraz większa złożoność systemów cyfrowych wymusza zwrócenie większej uwagi projektantów na zagadnienia dystrybucji sygnału zegarowego. Aktywne zbocze sygnału zegarowego powinno występować w tym samym czasie w znacznie oddalonych geometrycznie fragmentach układu scalonego (*clock skew*). W układach asynchronicznych, wobec braku globalnego sygnału zegarowego, problemy te nie występują.
2. Większa szybkość działania.
   Szybkość działania systemów synchronicznych jest ograniczona szybkością działania najwolniejszego elementu w systemie (*worst case performance*). W układach asynchronicznych wiele podsystemów może pracować współbieżnie i wówczas wydajność jest ograniczona zależnościami czasowymi pomiędzy przepływem danych i sygnałów sterujących. Szybkość działania systemów asynchronicznych może zmieniać się dynamicznie (poszczególne podsystemy mogą wykryć moment czasu w którym wykonywanie operacji zostanie zakończone) i dlatego determinowana jest poprzez element o przeciętnym czasie wykonania operacji (*average case performance*).
3. Możliwość adaptacji do zmiennych warunków pracy.
   Opóźnienia wnoszone przez poszczególne elementy logiczne układu scalonego mogą się zmieniać wraz ze zmianą parametrów procesu produkcyjnego, temperatury czy też napięcia zasilania układu. Projektant układów synchronicznych musi brać pod uwagę kombinację najgorszego przypadku spośród wymienionych czynników, dobierając odpowiednio częstotliwość taktowania całego systemu. Ze względu na adaptacyjną naturę układów asynchronicznych (wykrywanie momentu zakończenia wykonywania operacji) układy asynchroniczne będą pracować poprawnie odpowiednio zwiększając lub zmniejszając szybkość przetwarzania, niezależnie od zmian wnoszonych przez czynniki zewnętrzne.
4. Mniejszy pobór mocy.
   W przypadku układów asynchronicznych nie ma potrzeby stosowania dodatkowych drajwerów i buforów dla sygnału zegarowego, redukujących zjawisko różnicy faz (*clock skew*). W układach synchronicznych nawet w danej chwili nie używane komponenty są cały czas taktowane sygnałem zegarowym powodując niepotrzebne straty mocy. W przypadku współczesnych wysokowydajnych mikroprocesorów synchronicznych szacuje się, że moc tracona na dystrybucję sygnału zegarowego stanowi nawet połowę mocy pobieranej przez cały układ (typowa wartość to około 30%). W przypadku układów asynchronicznych moc tracona jest tylko w komponentach, które są w danej chwili aktywne (realizują określone operacje).
5. Mniejsza emisja zakłóceń elektromagnetycznych.
   Aktywność systemów synchronicznych skoncentrowana jest wokół pewnej ściśle określonej częstotliwości. Niemal cała energia wypromieniowanych zakłóceń elektromagnetycznych skupiona jest w wąskim paśmie wokół częstotliwości sygnału zegarowego oraz jej harmonicznych. Działanie poszczególnych podsystemów w układach asynchronicznych jest nieskorelowane, dlatego widmo zakłóceń ma charakter znacznie bardziej rozproszony. Mniejsza jest również maksymalna amplituda tych zakłóceń.
6. Większe możliwości modularyzacji systemu.
   Poszczególne komponenty układów asynchronicznych, a także całe systemy mogą być ze sobą bezpośrednio łączone bez konieczności dodatkowej synchronizacji, tak jak ma to miejsce w przypadku systemów synchronicznych taktowanych sygnałami zegarowymi o różnymi częstotliwościach.
7. Skuteczne działanie elementów realizujących wzajemne wykluczenie oraz praca z zewnętrznymi sygnałami wejściowymi.
   Elementy, które gwarantują poprawną realizację wzajemnego wykluczenia (*mutual exclusion*) niezależnych sygnałów oraz synchronizacja z sygnałem zegarowym zewnętrznych sygnałów wejściowych są podatne na występowanie zjawiska metastabilności. Stan metastabilny charakteryzuje się m. in. wydłużeniem, w ogólnym przypadku nieokreślonym, czasu odpowiedzi układu. W systemach synchronicznych wszystkie elementy systemu powinny charakteryzować się ograniczonymi opóźnieniami. Dlatego też istnieje duże prawdopodobieństwo, że synchroniczna realizacja elementów wzajemnego wykluczenia będzie prowadziła do błędnego działania układu. Z kolei systemy asynchroniczne są w stanie oczekiwać arbitralnie długi czas na ukończenie operacji swoich podsystemów, zapewniając tym samym poprawną realizację elementów wzajemnego wykluczenia. Dodatkowo, ponieważ w systemach asynchronicznych nie występuje sygnał zegarowy z którym muszą być synchronizowane inne sygnały, systemy te charakteryzują się znacznie lepszą akomodacją zewnętrznych sygnałów wejściowych, które z natury są asynchroniczne.

## Słabe strony systemów asynchronicznych

Mimo wymienionych wyżej zalet systemów asynchronicznych, jak również faktu, iż zainteresowanie nimi datuje się już od połowy lat 50-tych ubiegłego wieku, to jednak wciąż systemy te nadzwyczaj rzadko są realizowane w praktycznych aplikacjach. Istotnym problemem utrudniającym stosowanie układów asynchronicznych na szeroką skalę, jest niedostatek efektywnych metodologii projektowania i mała dostępność narzędzi wspomagających proces projektowania układów asynchronicznych [2]. Chociaż ostatnio pojawiły się dość skuteczne rozwiązania firmy Handshake Solution [3], to jednak w odniesieniu do systemów asynchronicznych podnoszone są też inne zarzuty, takie jak wymienione niżej [4]:

1. Mniejsze zużycie energii nie jest ewidentne.
   Co prawda w systemach asynchronicznych nie istnieje globalny sygnał zegarowy, którego dystrybucja prowadzi do znacznych strat mocy, jednak w zamian tworzone są lokalne sygnały żądania/potwierdzenia wykonania operacji, które są propagowane pomiędzy poszczególnymi stopniami/komponentami systemu. Ścieżki doprowadzeniowe tych sygnałów stają się krytyczne z punktu widzenia wydajności systemu, stanowią większe obciążenie o charakterze pojemnościowym oraz ich aktywność jest taka sama jak pozostałej logiki. W systemach asynchronicznych występuje więc przesunięcie zużycia energii na układy logiczne związane z obsługą tych sygnałów. Ten argument nie wydaje się jednak być trafnym w każdym przypadku. Na przykład największą zaletą zaprojektowanych przez Handshake Solution asynchronicznych mikroprocesorów jest właśnie niskie zużycie energii (także mała emisja zakłóceń elektromagnetycznych). Gdy mikroprocesor nie wykonuje żadnych operacji, moc pobierana jest pomijalnie mała. Należy jednak zauważyć, że w systemach synchronicznych również są stosowane pewne mechanizmy pozwalające np. na zaprzestanie taktowania niewykorzystywanych w danej chwili komponentów, co również prowadzi do redukcji pobieranej energii.
2. Wydajność systemów asynchronicznych jest niedeterministyczna.
   Szybkość działania układów asynchronicznych może zmieniać się dynamicznie zależnie od warunków środowiska pracy a nawet samych danych. Oznacza to, że wydajność systemu, rozumiana jako szybkość pracy, nie jest znana z góry – jest niedeterministyczna. W przypadku niektórych zastosowań nie ma to znaczenia. Ogólnie jednak właściwość taka nie jest pożądana. Na przykład układy asynchroniczne nie nadawałyby się do implementacji systemów czasu rzeczywistego o twardych wymaganiach czasowych (*hard real time systems*), dla których niezwykle istotna jest deterministyczna znajomość czasu odpowiedzi systemu.
3. Uruchamianie i testowanie systemów asynchronicznych jest bardzo trudne.
   Uruchamiając system synchroniczny można dowolnie zmniejszać (zmieniać) częstotliwość sygnału taktującego sprawdzając jaki to ma wpływ na działanie układu i w ten sposób stosunkowo łatwo identyfikować element systemu, który zawodzi. Układy asynchroniczne muszą być uruchamiane przy pełnej prędkości działania. Wykrycie elementu, który działa nieprawidłowo wymaga niemal detektywistycznych umiejętności, wnioskowania na podstawie zachowanie systemu itp. Również testowanie układów asynchronicznych za pomocą testerów, które z natury przeznaczone są do testowania systemów synchronicznych, nie jest łatwe i wymaga pewnych specjalnych zabiegów.

Do wymienionych wyżej słabych stron systemów asynchronicznych można jeszcze dodać to, że ze względu na konieczność eliminacji zjawisk takich jak hazard oraz implementacji logiki odpowiedzialnej za generowanie sygnałów żądania/potwierdzenia wykonania operacji - systemy te często są znacznie bardziej złożone (zajmują większą powierzchnię krzemu) od analogicznych systemów realizowanych w sposób synchroniczny. Również wiedza na temat zagadnień projektowania układów asynchronicznych nie jest zbyt rozpowszechniona.

## Klasyfikacja układów asynchronicznych

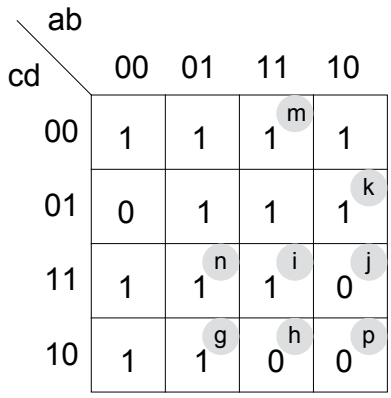
Układy asynchroniczne można podzielić na trzy kategorie:

- klasyczne układy typu Huffmana,
- układy z samotaktowaniem (*self-clocked*),
- układy ze współpracą lokalną (*self-timed*).

Inny podział układów asynchronicznych uwzględnia założenia związane z przyjętym modelem czasowym. Według tego podziału możemy wyróżnić [2]:

- układy bazujące na modelu ograniczonych opóźnień (*bounded delay*),
- układy mikropotokowe (*micropipeline*),
- układy niewrażliwe na opóźnienia (*delay-insensitive*),
- układy niezależne od szybkości (*speed-independed*).

Układy asynchroniczne, a ściślej asynchroniczne automaty sekwencyjne, można też sklasyfikować ze względu na pewne ograniczenia związane ze zmianą stanu wejść. Wyróżnia się układy SIC o zmianie stanu jednego wejścia (*single-input change*), układy MIC o jednoczesnej zmianie stanu wielu wejść (*multiple-input change*) oraz UIC – układy o zmianie stanów wejścia bez ograniczeń (*unrestricted-input change*). W układach SIC tylko jedno wejście układu może zmieniać się w danej chwili czasu. Następne zmiany stanu wejścia muszą być przedzielone pewnym minimalnym interwałem czasowym δ. W układach MIC wiele sygnałów wejściowych może zmieniać się w określonym przedziale czasowym o długość




**Rysunek 1. Przykładowa funkcja logiczna z hazardem funkcyjnym**

$\delta_1$. Te zmiany traktowane są jako symultaniczne. Następne zmiany stanu wejść możliwe są po upływie czasu $\delta_2$. W przypadku układów UIC dowolne wejście układu może zmieniać się w dowolnym czasie, pod warunkiem, że żaden sygnał wejściowy nie zmieni się dwa razy w ciągu jakiegokolwiek okresu o długości δ.

## Hazard w układach kombinacyjnych

Spora część problemów związanych z projektowaniem systemów asynchronicznych koncentruje się wokół zjawiska hazardu lub pewnych niepożądanych impulsów mogących pojawiać się na wyjściu układów logicznych. W systemach synchronicznych wystąpienie krótkotrwałych, niepożądanych impulsów jest dopuszczalne z wyjątkiem chwili czasu w pobliżu aktywnego zbocza zegara. Wpływ tych impulsów na działanie systemu synchronicznego można wyeliminować poprzez odpowiedni dobór okresu sygnału zegarowego. W systemach asynchronicznych nie występuje globalny sygnał zegarowy. System taki powinien odpowiedzieć na zmianę stanu wejść w dowolnej chwili czasu. Dlatego też wszelkie niepożądane impulsy, będące wynikiem np. zjawiska hazardu, prowadzą do nieprawidłowej pracy systemu.

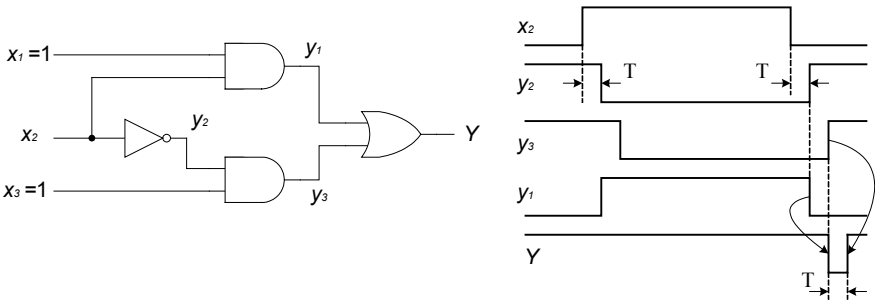
Przypomnijmy krótko podstawowe aspekty związane z występowaniem zjawiska hazardu w układach kombinacyjnych oraz sekwencyjnych.

Dla układów kombinacyjnych, oprócz hazardu logicznego, definiuje się również hazard funkcyjny (*function hazard*) [5]. Dowolna funkcja boole'owska $f$ więcej niż jednej zmiennej, której wartość nie zmienia się monotonicznie (tzn. tylko raz z 0 na 1 lub z 1 na 0) podczas zmiany stanu zmiennych wejściowych, zawiera hazard funkcyjny dla tej zmiany stanu wejść.

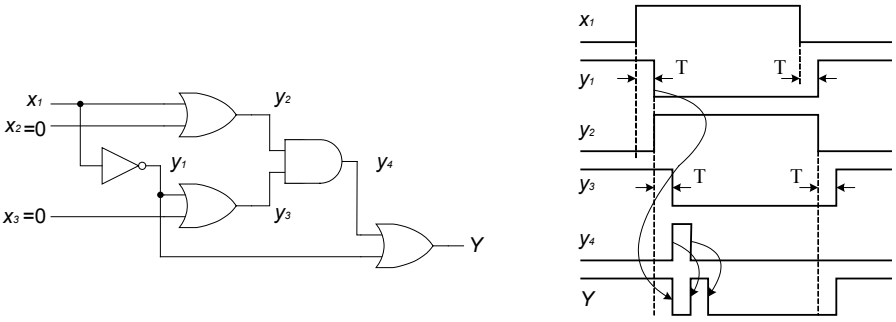
Funkcja $f$ zawiera statyczny hazard funkcyjny przy zmianie stanu zmiennych wejściowych ze stanu A do stanu C, gdy $f(A)=f(C)$ i istnieje przynajmniej jeden stan wejścia $B\epsilon[A,C]$, czyli taki stan, który muszą przyjąć zmienne wejściowe przechodząc ze stanu A do stanu C, przy założeniu zmiany tylko jednej zmiennej wejściowej (bitu) w danej chwili, dla którego stanu $f(B)\neq f(A)$. Funkcja $f$ zawiera dynamiczny hazard funkcyjny przy zmianie stanu zmiennych wejściowych ze stanu A do stanu D, gdy $f(A)\neq f(D)$ i istnieje przynajmniej jedna para stanów B i C zmiennych wejściowych, takich że $B\epsilon[A,D]$ i $C\epsilon[B,D]$, dla których $f(B)=f(D)$ i $f(A)=f(C)$.

Rozważmy przykładową funkcję logiczną $f$ zdefiniowaną za pomocą tablicy Karnaugha na **rysunku 1**. Funkcja ta zawiera statyczny hazard funkcyjny dla zmiany wejść ze stanu i do stanu k: $f(i)=f(k)$, $f(j)=0$ i $j\epsilon[i,k]$. Funkcja $f$ zawiera z kolei dynamiczny hazard funkcyjny dla zmiany wejść ze stanu g do stanu j: $f(g)=1$, $f(j)=0$, $h\epsilon[g,j]$, $i\epsilon[h,j]$, $f(g)=f(i)=1$, $f(h)=f(j)=0$. Zmiana wejść funkcji $f$ ze stanu k do m jest wolna od statycznego hazardu funkcyjnego, a zmiana wejść ze stanu n do stanu p jest wolna od dynamicznego hazardu funkcyjnego.

Hazard funkcyjny jest właściwością funkcji logicznej i nie może być wyeliminowany poprzez odpowiedni dobór opóźnień lub modyfikację sieci logicznej. Jeżeli funkcja $f$ nie zawiera hazardu funkcyjnego przy zmianie stanów wejściowych, to realizacja




**Rysunek 2. Układ kombinacyjny z hazardem statycznym w 1 [6]**




**Rysunek 3. Układ kombinacyjny z hazardem dynamicznym [6]**

tej funkcji może wciąż zawierać hazard logiczny ze względu na opóźnienia powodowane przez rzeczywiste elementy wykorzystane do implementacji funkcji. Hazardem logicznym w układach kombinacyjnych określa się krótkie zakłócenia szpilkowe, pojawiające się na wyjściu układu podczas procesów przejściowych [6].

Układ kombinacyjny zawiera statyczny hazard logiczny dla funkcji *f*, przy zmianie stanu zmiennych wejściowych ze stanu A do stanu B, gdy $f(A)=f(B)$ i następuje chwilowa zmiana stanu wyjścia układu podczas wspomnianej zmiany stanów wejściowych. Układ kombinacyjny zawiera dynamiczny hazard logiczny dla funkcji *f*, przy zmianie stanu zmiennych wejściowych ze stanu A do stanu B, gdy $f(A)\neq f(B)$ i wyjście układu nie zmienia się monotonicznie podczas trwania opisanej zmiany stanów wejściowych.

Hazard na wyjściu układu logicznego może pojawić, gdy [6]:

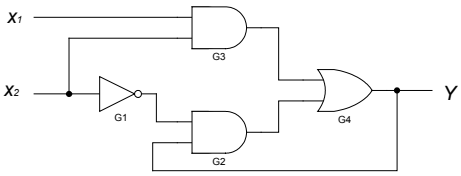
- choćby jeden sygnał wejściowy dochodzi do wyjścia drogami o różnych opóźnieniach,
- przy jednoczesnej zmianie dwóch lub więcej sygnałów wejściowych przychodzą one do wyjścia drogami o różnych opóźnieniach,
- układ zapewnia dla wszystkich sygnałów wejściowych drogi o jednakowych opóźnieniach, lecz sygnały te zmieniają swe stany logiczne niejednocześnie (hazard funkcyjny).

Jeżeli sieć logiczna zawiera hazard funkcyjny dla danej zmiany stanu wejść, to nie może jednocześnie zawierać hazardu logicznego dla tej zmiany wejść.

Na **rysunku 2** przedstawiono przykładowy układ z hazardem statycznym w 1, a na **rysunku 3** przykład układu wielopoziomowego z hazardem dynamicznym. Dla podanych wykresów czasowych przyjęto założenie, że każda z bramek logicznych wnosi opóźnienie jednostkowe T.

Istnieje wiele opracowanych metod eliminacji zjawiska hazardu. Dla układu z rysunku 2 aby wyeliminować hazard statyczny, pierwotną funkcję $Y = x_1 \cdot x_2 + \overline{x_2} \cdot x_3$ należy zmodyfikować dodając człon konsensusu $x_1 \cdot x_3$, czyli: $Y = x_1 \cdot x_2 + \overline{x_2} \cdot x_3 + x_1 \cdot x_3$. W tablicy Karnaugha odpowiada to dodatkowemu połączeniu sąsiadujących grup implikantów. Metoda ta jest jednak skuteczna tylko w przypadku, gdy w danej chwili zmienia się sygnał tylko jednej zmiennej wejściowej. W przypadku przeciwnym układ może nadal zawierać hazard. Układ z rysunku 3 można z kolei zminimalizować do postaci $Y = \overline{x_1} + x_3$ eliminując tym samym hazard dynamiczny. Wówczas jednak, przy jednoczesnej zmianie $x_1$ i $x_3$ z 1 na 0, wystąpi hazard statyczny w 1. Wtedy pozostaje jedynie ograniczenie liczby zmiennych wejściowych, które mogą zmieniać się w danej chwili czasu, tylko do jednej (układy typu SIC).

Jednakże wymienione wyżej metody eliminacji hazardu odnoszą się do realizacji układów cyfrowych na poziomie bramek logicznych i nie nadają się bezpośrednio do zastosowania w układach FPGA. Jeżeli dana funkcja logiczna może być zaimplementowana za pomocą pojedynczej tablicy LUT (w układach FPGA opartych na tablicach LUT), wówczas implementacja tej funkcji jest wolna od hazardu. Jednak w przypadku realizacji funkcji logicznej zajmującej więcej niż jedną tablicę LUT hazard nadal może wystąpić. Celem uniknięcia tego hazardu na ogół konieczna jest ingerencja projektanta w sposób działania narzędzi rozmieszczania i planowania połączeń, np. poprzez wykorzysta-




**Rysunek 4. Prosty układ sekwencyjny z hazardem**

nie makr sprzętowych, umieszczenie odpowiednich dyrektyw rozmieszczania względnego w pliku wymuszeń projektanta itp.

## Hazard w układach sekwencyjnych

Hazard w układach sekwencyjnych, spowodowany propagacją poprzez pętlę sprzężenia zwrotnego nowej wartości sygnału jeszcze przed zakończeniem propagacji sygnału wejściowego do wszystkich punktów wewnętrznych układu, określa się jako hazard podstawowy (*essential hazard*) [5]. Jeżeli w wyniku wystąpienia hazardu podstawowego na wyjściu układu pojawi się niepożądany impuls (*glitch*) wówczas taki hazard nazywa się przejściowym hazardem podstawowym (*transient essential hazard*). Jeżeli zaś w rezultacie pojawienia się hazardu podstawowego układ przejdzie do niewłaściwego stanu stabilnego, wówczas taki hazard określa się jako podstawowy hazard stanu stabilnego (*steady-state essential hazard*).

Hazard podstawowy stanowi istotną cechę konkretnej implementacji układu. Potencjalnie hazard ten może być wykryty poprzez analizę tablicy przejść, jeśli pojedyncza zmiana sygnału wejściowego doprowadza opisywany układ do innego stanu końcowego, niż w przypadku, w którym wartość rozpatrywanego sygnału zmieniłaby się trzy razy. Jedynym sposobem uniknięcia hazardu zasadniczego jest zapewnienie, że sygnały zmiennych stanu nie mogą być podawane zwrotnie do układu przed rozpatrywaną zmianą sygnału wejściowego. Można to osiągnąć poprzez staranne rozmieszczenie topograficzne elementów cyfrowych, jak również poprzez celowe wprowadzenie dodatkowych opóźnień sygnałów w ścieżkach zmiennych stanu.

Przykład prostego asynchronicznego układu sekwencyjnego (typu Huffmana) zawierającego hazard podstawowy przedstawiono na **rysunku 4**. Układ realizuje funkcję $y' = x_1 \cdot x_2 + \overline{x_2} \cdot y$ ($y'$ oznacza stan „następny” wyjścia $y$). Rozważmy przejście układu ze stanu stabilnego $(y, x_1, x_2) = (1, 1, 1)$ do stanu $(1, 1, 0)$ – zmiana stanu wejścia $x_2$ z 1 na 0. Na skutek dodatkowego opóźnienia wnoszonego przez inwerter G1 sterujący bramkę G2 i propagacji nowej wartości sygnału przez pętlę sprzężenia zwrotnego, układ znajdzie się w stanie stabilnym (0,1,0), zamiast – zgodnie z równaniem, w stanie (1,1,0).

Hazard w układach sekwencyjnych może być przyczyną wyścigów, w tym niekorzystnych wyścigów krytycznych. Wyścig nazywamy krytycznym, jeżeli zmiana stanu stabilnego automatu przebiegająca poprzez różne stany przejściowe, może doprowadzić do innego stanu stabilnego, niż pożądany. Wyścig nazywamy niekrytycznym, jeżeli zmiana stanu stabilnego automatu, przebiegająca poprzez różne stany przejściowe, prowadzi do pożądanego końcowego stanu stabilnego [6]. Zjawiska wyścigów mogą być wyeliminowane przez odpowiednie zakodowanie stanów wewnętrznych automatu.

**Zbigniew Hajduk**
**zhajduk@prz-rzeszow.pl**

***Literatura***

*1 Myers Ch. J.: Asynchronous Circuit Design, John Wiley & Sons Inc., 2001.*
*2 Hauck S.: Asynchronous Design Methodologies: An Overview,*
*3 Handshake Solutions: http://www.handshakesolutions.com/.*
*4 Borkar S.: Does asynchronous logic design really have a future?, EE Times, 2003.*
*5 Unger S. H., Hazards, Critical Races and Metastability, IEEE Trans. on Comp., Vol. 44, No. 6, June 1995.*
*6 Kalisz J., Podstawy elektroniki cyfrowej, WKŁ, Warszawa 1998.*