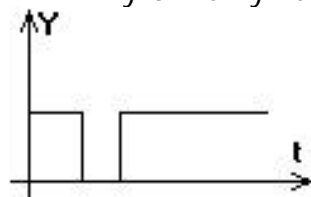


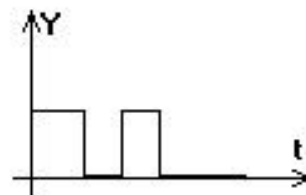
Hazardy.

Zjawisko powstawania błędnych stanów na wyjściu układu w stanach przejściowych nosi nazwę hazardu. Błędne stany na wyjściu (hazardy) powstają w stanach przejściowych, gdyż wtedy dają znać o sobie rzeczywiste właściwości przełączające i transmisyjne elementów cyfrowych. Podstawowym elementem budowy układów logicznych są bramki. To za pomocą bramek możemy zbudować większość urządzeń i elementów cyfrowych takich jak np. przerzutniki rejestry czy pamięci. Charakterystyki bramek rzeczywistych różnią się od charakterystyk elementu idealnego. Każdy rzeczywisty element wprowadza pewne opóźnienie którego miarą jest czas propagacji t_p . Właściwości elementów rzeczywistych powodują, że zbudowane z tych elementów układy pracują w sposób odbiegający od ich matematycznego opisu, którym to posługuje się projektant układów cyfrowych. Przy symulacjach opóźnienie bramki jest idealizowane i z reguły przyjmuje się, że jest ono maksymalne. W rzeczywistości jest ono w zakresie pomiędzy wartością minimalną a maksymalną. W regularnej sieci dwupoziomowej hazardy występują tylko na skutek różnych czasów propagacji bramek, dlatego w celu wykrycia hazardów należy rozważać równocześnie czasy propagacji maksymalne i minimalne, co nie jest możliwe dla typowej symulacji. Dlatego wykrycie hazardów podczas symulacji jest bardzo trudne – nie występowanie hazardu podczas symulacji nie świadczy, że nie występują one w praktyce.

Wyróżniamy hazardy statyczne i hazardy dynamiczne.



Hazard statyczny



Hazard dynamiczny

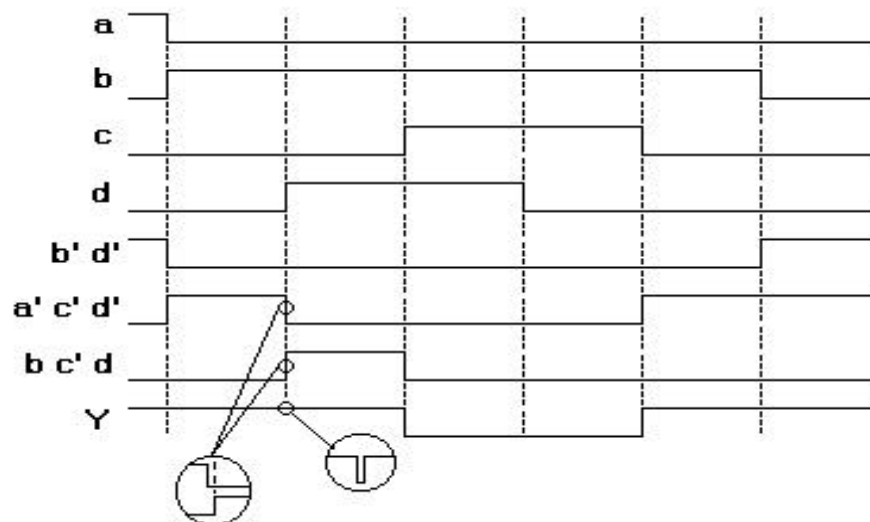
Jeżeli źródłem niepożądanego stanu wyjściowego na wyjściu układu są nieidealne właściwości przełączające, to takie przekłamanie nazywamy hazardem statycznym. Hazardami dynamicznymi nazywamy hazardy wywołane przez nieidealne właściwości transmisyjne elementów cyfrowych. Hazardy statyczne charakteryzują się pojedynczymi, chwilowymi zmianami stanu na wyjściu układu, hazardy dynamiczne zaś wielokrotnymi zmianami powtarzającymi co pewien czas. Wyróżniamy hazardy statyczne w zerze i w jedynce. Hazard statyczny w zerze charakteryzuje się pojawieniem na wyjściu układu krótkotrwałego impulsu (szpilki) o wartości „1”, gdy wyjście układu powinno pozostać w stanie „0”, analogicznie hazard w jedynce charakteryzuje pojawienie na wyjściu szpilki do „0”, gdy wyjście to powinno pozostać w stanie „1”.

Hazardy mogą powodować nieprawidłowe działanie układów jeżeli wyjścia na których się pojawiają są interpretowane asynchronicznie. W przypadku sygnałów synchronicznych, hazardy (a właściwie przejściowe stany nieustalone) nie powodują zakłócenia działania.

Sposoby usuwania hazardów przesledzimy na przykladzie funkcji logicznej zilustrowanej nastepujaca tablica Karnaugha

cd \ ab	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	0	1	0	0
10	1	0	0	1

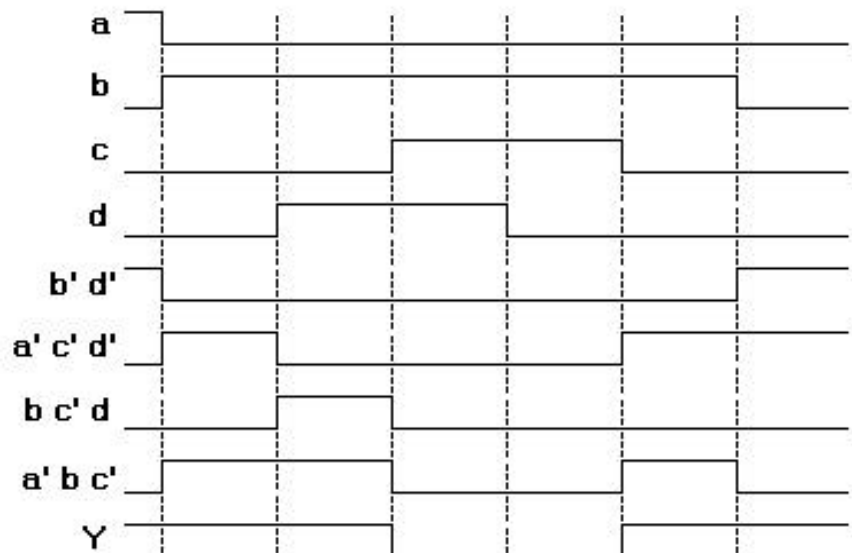
Na podstawie tablicy Karnaugha otrzymujemy zminimalizowana funkcje logiczna: $y = b' d' + a' c' d' + b c' d$



Otrzymana funkcja jest minimalna lecz nie jest wolna od hazardu dzieje sie tak dlatego, ze dwie sasiadujace jedyнки nie sa polaczone wspólna grupa. Sposobem wyeliminowania tego hazardu jest polaczenie dodatkowa grupa sasiadujacych, lecz nie polaczonych ze soba elementów.

cd \ ab	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	0	1	0	0
10	1	0	0	1

Otrzymamy tym samym nastepujaca funkcje logiczna:
 $y = b' d' + a' c' d' + b c' d + a' b c'$



Zaimplementowany układ będzie wolny od hazardów ponieważ przy zmianie stanu wejścia d z „0” na „1” wyjście będzie podtrzymywane w stanie „1” przez dodatkową bramkę AND ($a' b c'$), która nie zmienia stanu w trakcie tej operacji.

Podobnie postępujemy przy minimalizacji funkcji do postaci koniunkcyjnej tyle, że łączymy teraz w grupy „0” w taki sposób, aby każde dwa sąsiadujące były w tej samej grupie.

cd \ ab	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	0	1	0	0
10	1	0	1	1

Na podstawie tablicy Karnaugh otrzymujemy zminimalizowaną funkcję logiczną: $y = (a+b+d')(c'+d')(a'+b+d)(b'+d)$

Podobnie jak poprzednio otrzymana funkcja jest minimalna lecz nie jest wolna od hazardu gdyż sąsiadujące „0” nie należą do wspólnej grupy.

Łącząc wszystkie sąsiadujące zera we wspólne grupy otrzymujemy:

cd \ ab	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	0	1	0	0
10	1	0	1	1

Otrzymamy tym samym następującą funkcję:

$$y = (a+b+d') (c'+d') (a'+b+d') (b'+d) (b'+c')$$

Zaimplementowany układ będzie wolny od hazardów ponieważ przy zmianie stanu wejścia d z „1” na „0” wyjście będzie podtrzymywane w stanie „1” przez dodatkową bramkę OR ($b'+c'$), która nie zmienia stanu w trakcie tej operacji.

W sieci dwupoziomowej wyeliminowanie hazardu statycznego gwarantuje wykonanie sieci wolnej również od hazardów dynamicznych. W sieciach wielopoziomowych eliminacja hazardów statycznych nie wystarcza do eliminacji hazardów dynamicznych. Istnieją techniki eliminujące hazardy dynamiczne w sieciach wielopoziomowych, ale są one raczej dość skomplikowane, więc najlepiej jest budować sieci dwupoziomowe. Eliminacja hazardów zakłada, że zmianie ulega równocześnie nie więcej niż jeden sygnał wejściowy. Praktycznie jest bardzo mało prawdopodobne, że zmieniają się w tym samym czasie (z dokładnością co do propagacji pojedynczej bramki – około 1ns) dwa lub więcej sygnałów, dlatego założenie to jest prawie zawsze spełnione (chyba, że układ asynchroniczny jest sterowany układem synchronicznym). Przy symulacji możemy łatwo w tym samym czasie zmieniać wszystkie sygnały wejściowe i dlatego ciągle mogą być obserwowane hazardy, mimo tego, że w teorii one nie występują.

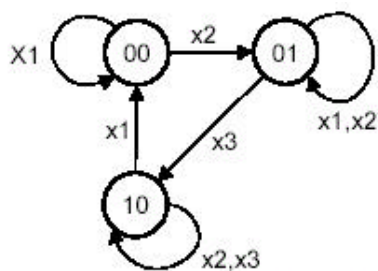
Równie niekorzystnym zjawiskiem jest zjawisko gonitwy. Ze zjawiskiem tym mamy do czynienia w układach bardziej rozbudowanych takich jak automaty jeżeli więcej niż jedna zmienna ulega zmianie podczas stanu wejściowego. W takim przypadku nie jesteśmy w stanie przewidzieć która zmieni się pierwsza (wygra wyścig).

W automatach asynchronicznych aby zapobiegać zjawisku gonitwy stosujemy przy projektowaniu automatów tzw. kodowanie antygonitwowe. Kodowanie to polega na tym, że każdą parę stanów sąsiednich kodujemy za pomocą kodów różniących się jednym bitem. Przykładem takiego kodowania jest zapisanie stanów automatu za pomocą kodu Graya.

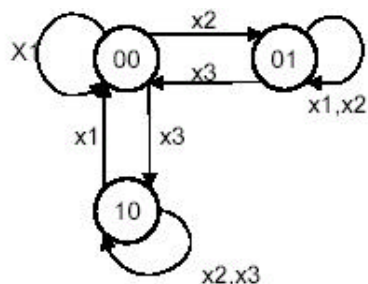
S_n	Q_1	Q_0
S1	0	0
S2	0	1
S3	1	1
S4	1	0

Niekiedy aby zrealizować takie kodowanie konieczna jest modyfikacja automatu przy pomocy stanów pośrednich lub stosowanie przejść cyklicznych.

Przejścia cykliczne:



Możliwość wystąpienia gonitwy



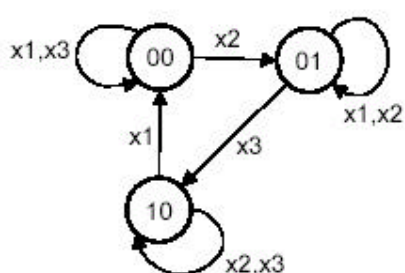
Brak zjawiska gonitwy

Dla x_3 przy przejściu ze stanu „01” do „10” możliwe jest wystąpienie gonitwy (gdyż w jednym czasie zmieniają się stany na dwóch wyjściach przerzutników automatu).

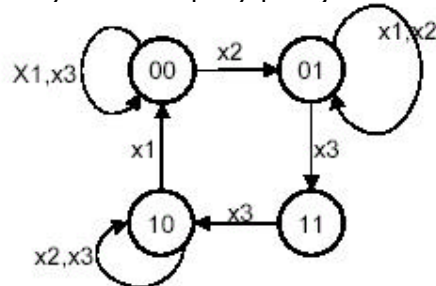
Dlatego też stosuje się tak zwane przejście cykliczne polegające na tym iż dla x_3 przechodzimy najpierw do stanu „00” a z tego z kolei do stanu „10”. Przejście takie charakteryzuje się tym, że w danym momencie na wyjściu przerzutników automatu zmienia się tylko jeden ze stanów.

Dodanie stanu pośredniego:

Zjawisko gonitwy możemy niekiedy także eliminować poprzez dodanie stanu pośredniego. Jest to nowy stan automatu, który różni się tylko jednym bitem zarówno dla stanu „początkowego” jak i „koncowego” dla danego przejścia. W naszym przykładzie jest to stan „11” różniący się zarówno od stanu „01” – początkowego jak i „10” – końcowego jednym bitem przy przejściu dla x_3 .



Możliwość wystąpienia gonitwy



Brak zjawiska gonitwy

Takie przejście zapewnia nam to iż w danej chwili na wyjściu przerzutników automatu zmienia się tylko jeden ze stanów.