Praktyka projektowania

Program wykładu

- **Metastabilność**
- **Reżimy czasowe**
- **Styl projektowania**
- **Styl kodowania**

Literatura

„A VHDL Synthesis primer” J.Bhasker,

„VHDL A Logic Synthesis Approach” D.Naylor, S.Jones,

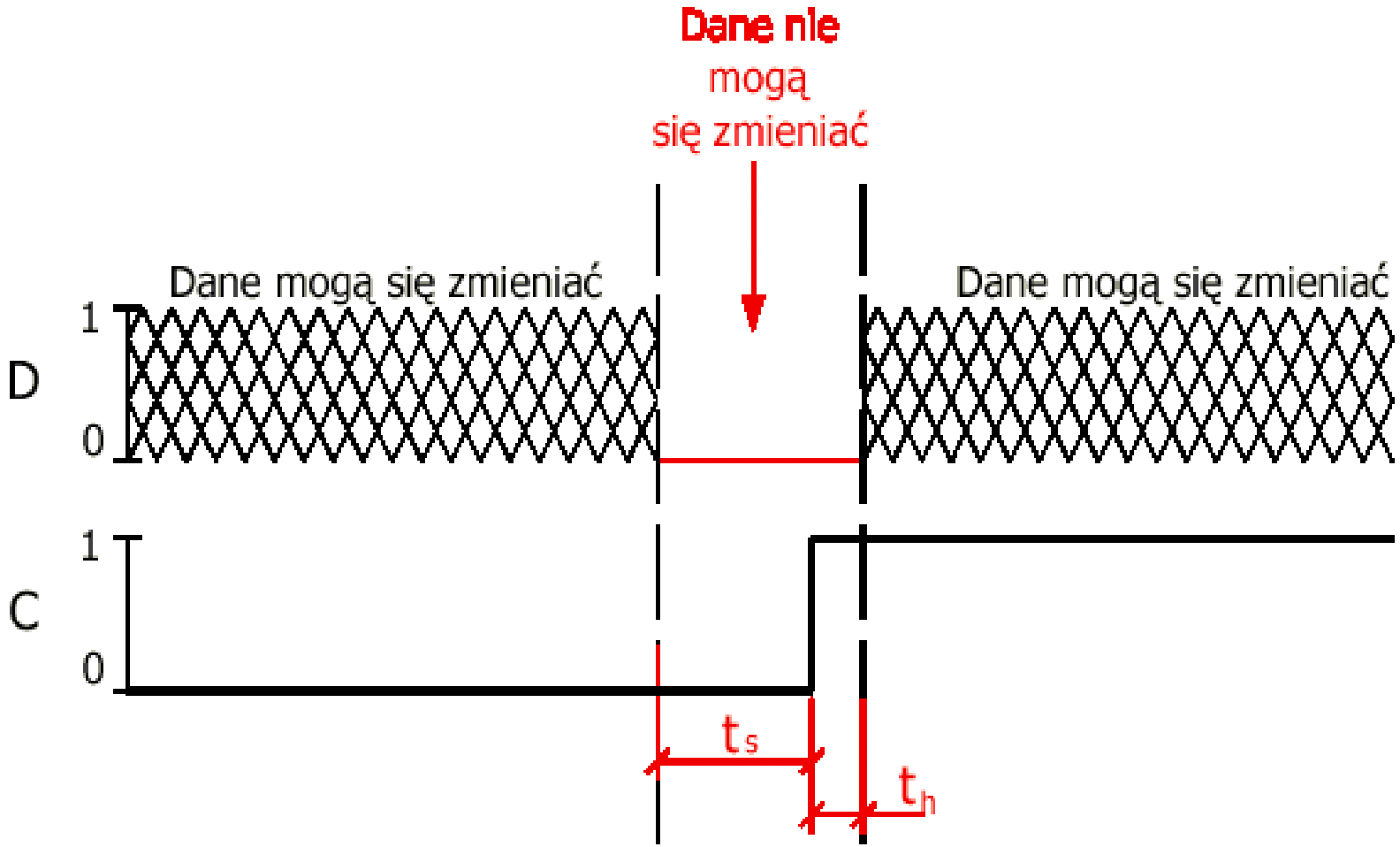
„VHDL Coding and Logic Synthesis with SYNOPSIS”
W.F.Lee,

„Reuse Methodology Manual” M.Keating, P.Bricaud,

„Synthesis and Simulation Design Guide” (Xilinx manual)

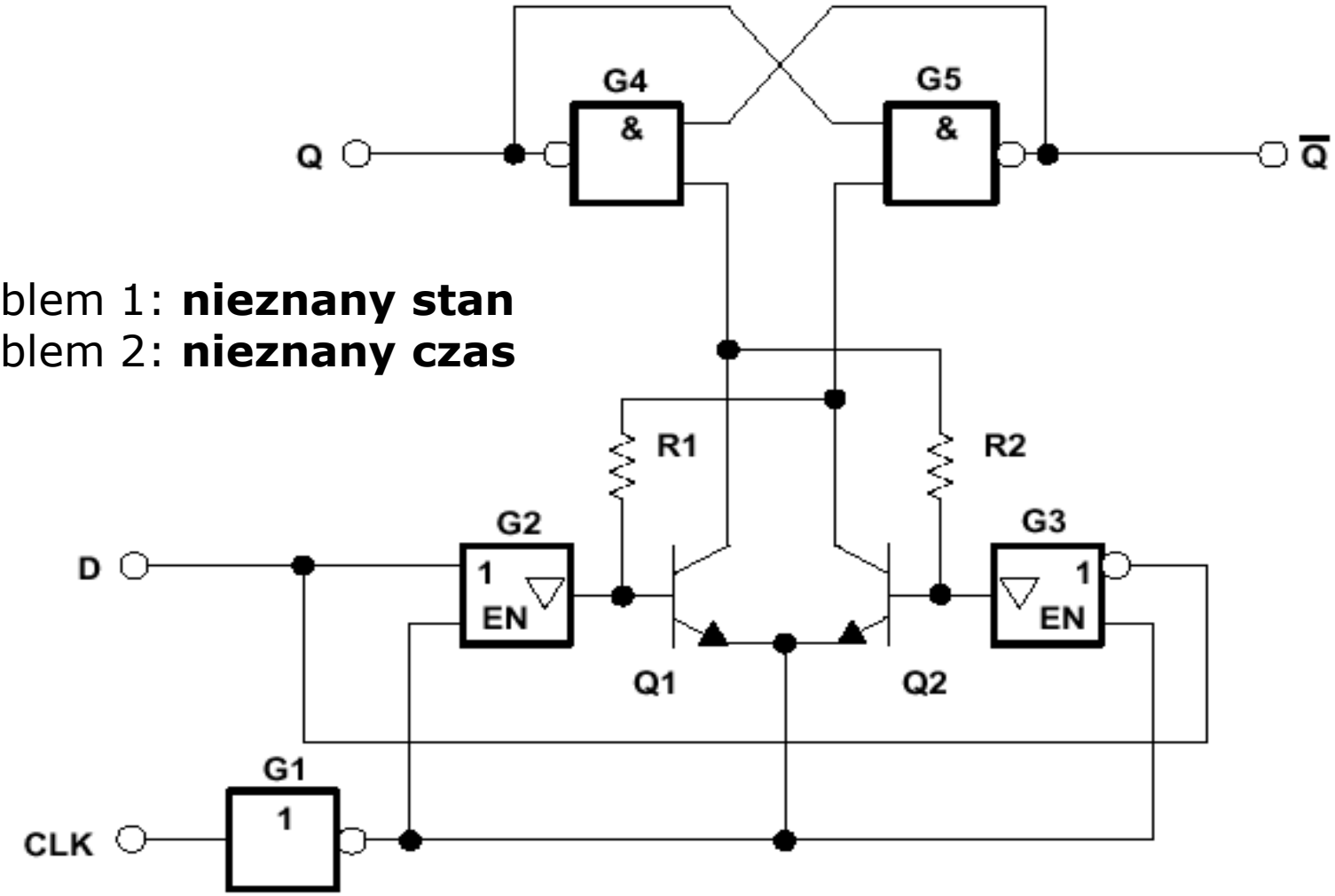
„Xilinx Synthesis Technology (XST) User Guide”
(Xilinx manual)

Warunki

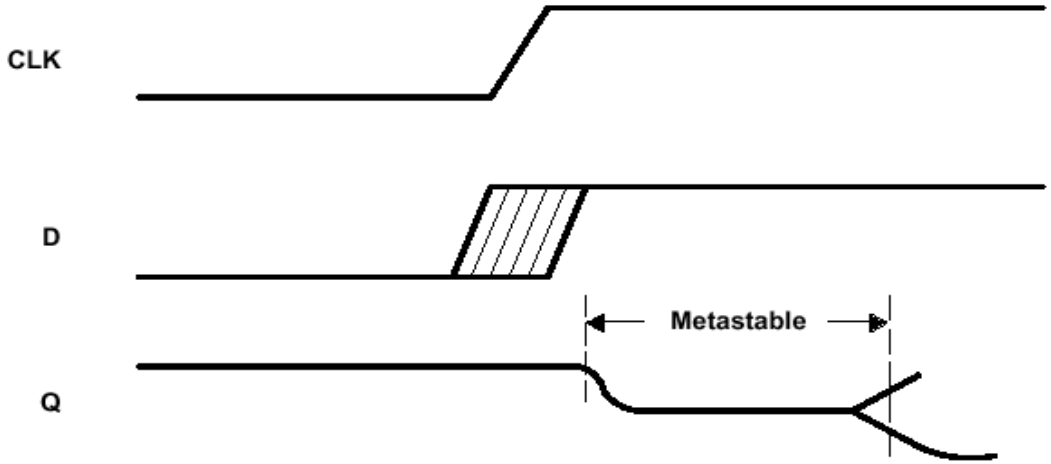

Dane nie
mogą
się zmieniać
Dane mogą się zmieniać
Dane mogą się zmieniać
D
1
0
C
1
0
t_s
t_h

Przyczyna

- Problem 1: **nieznany stan**
- Problem 2: **nieznany czas**

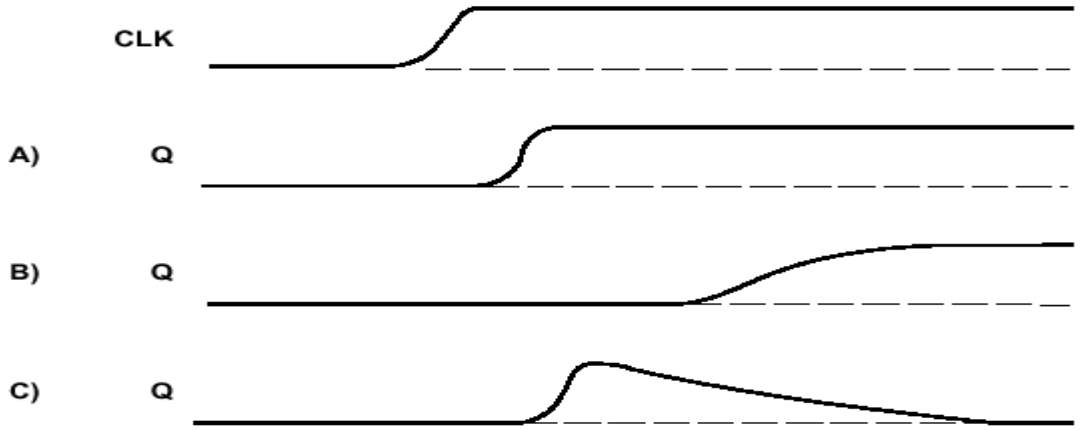


Objawy



$$MTBF = \frac{1}{f_{in} \cdot f_{clk} \cdot T_d}$$

T_d = 10...150ps
(*critical time window*)

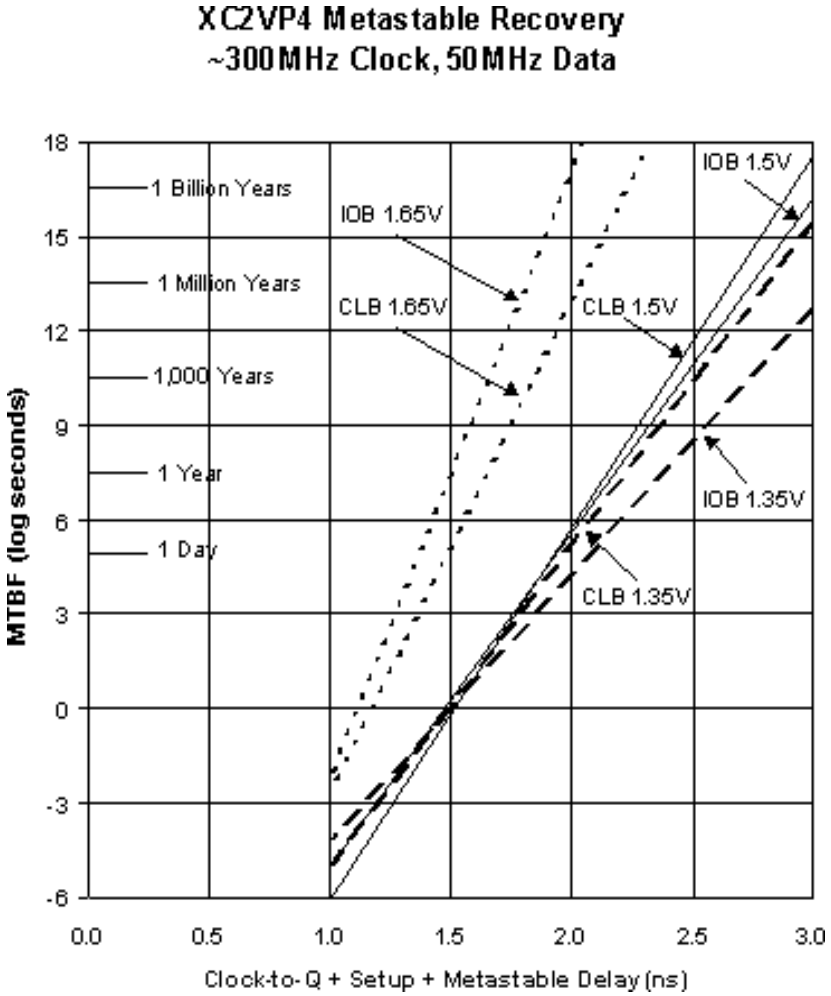


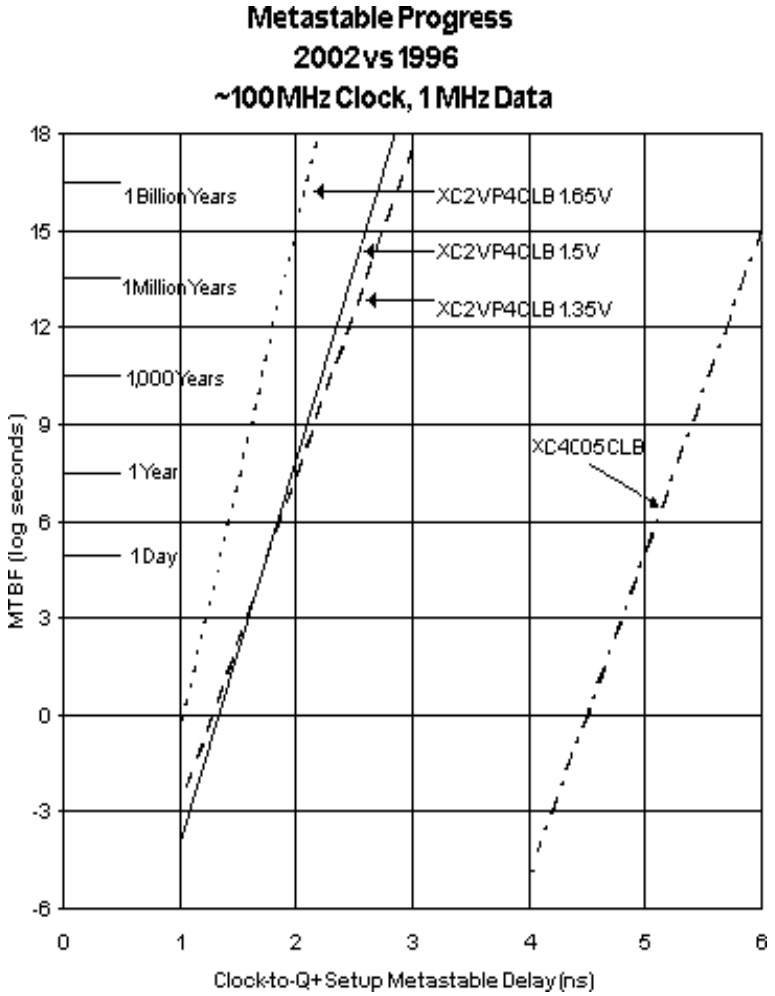
Na przykład dla:

f_{in} = 1Hz
f_{clk} = 1MHz
T_d = 30ps

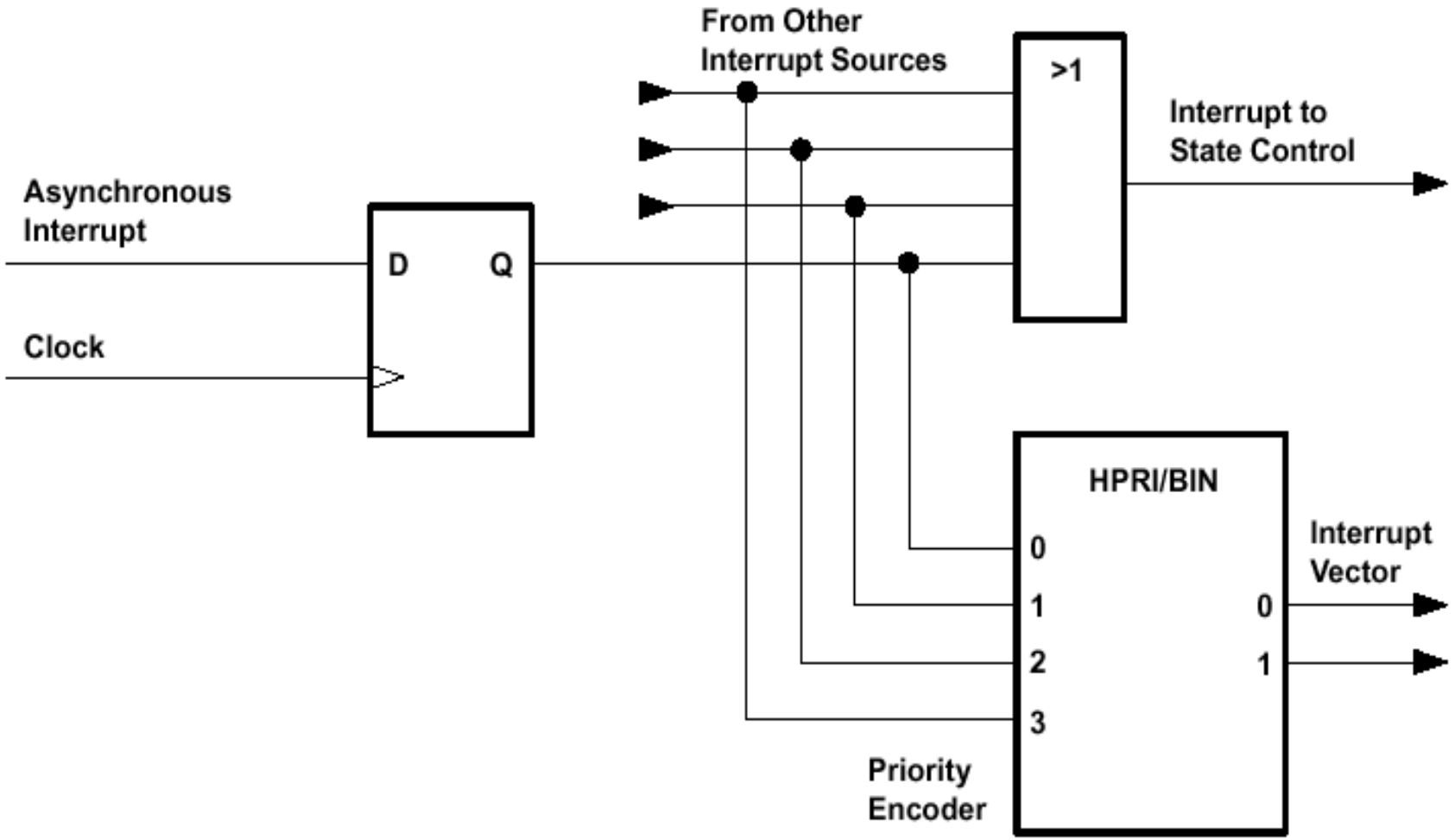
MTBF = 33,3s

Metastabilność
Wartości





Przykład

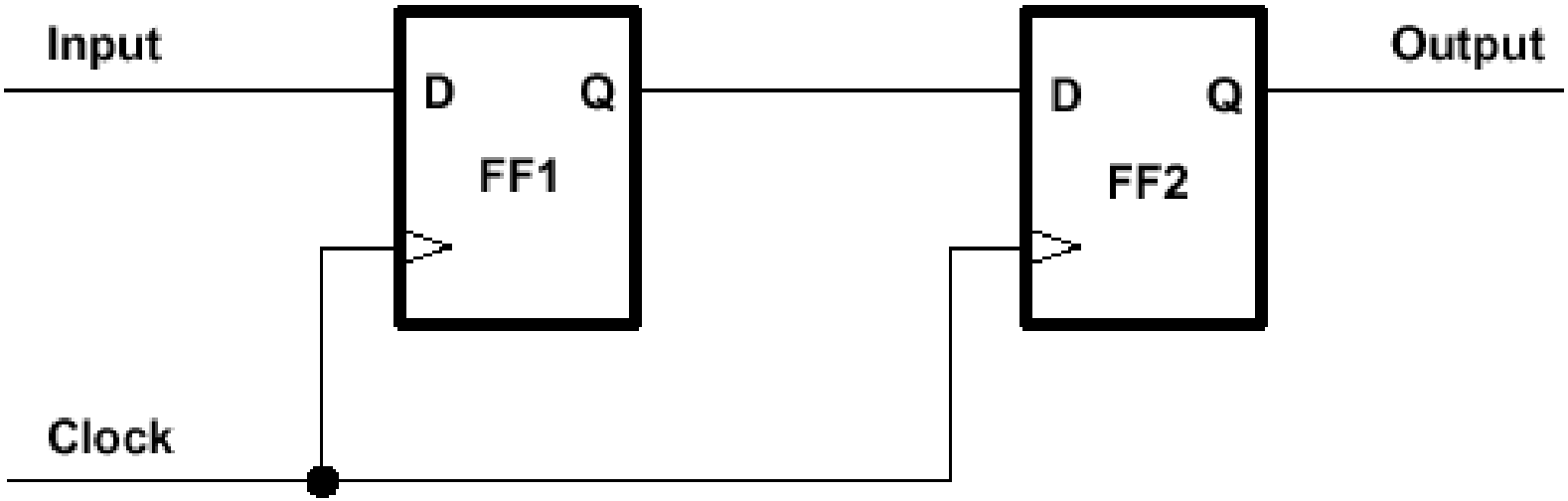

From Other
Interrupt Sources
>1
Interrupt to
State Control
Asynchronous
Interrupt
D
Q
Clock
HPRI/BIN
0
1
2
3
Interrupt
Vector
0
1
Priority
Encoder

- **Nowsze rodziny układów cyfrowych**

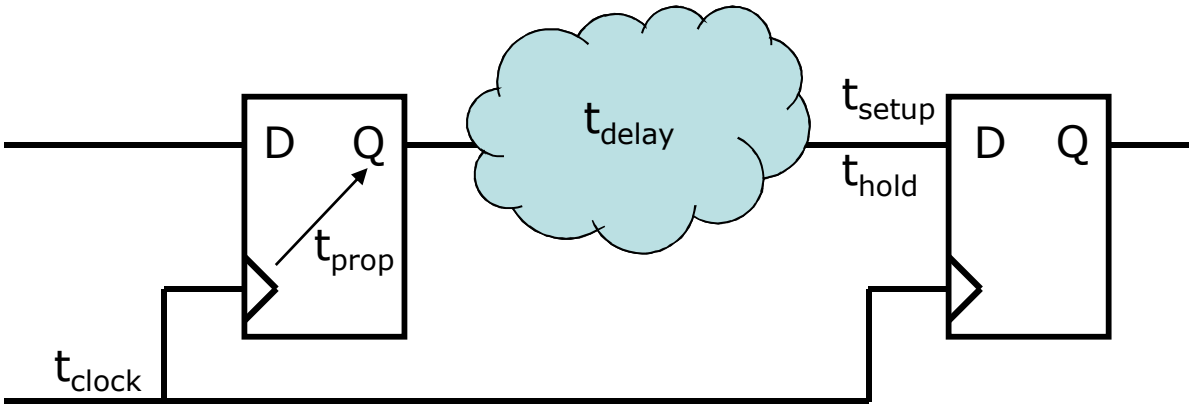
  minimalizacja czasów T_d (szybkość) oraz T_s i T_h (powtarzalność)

- **Synchronizator**

  1+ przerzutników – minimalizacja prawdopodobieństwa



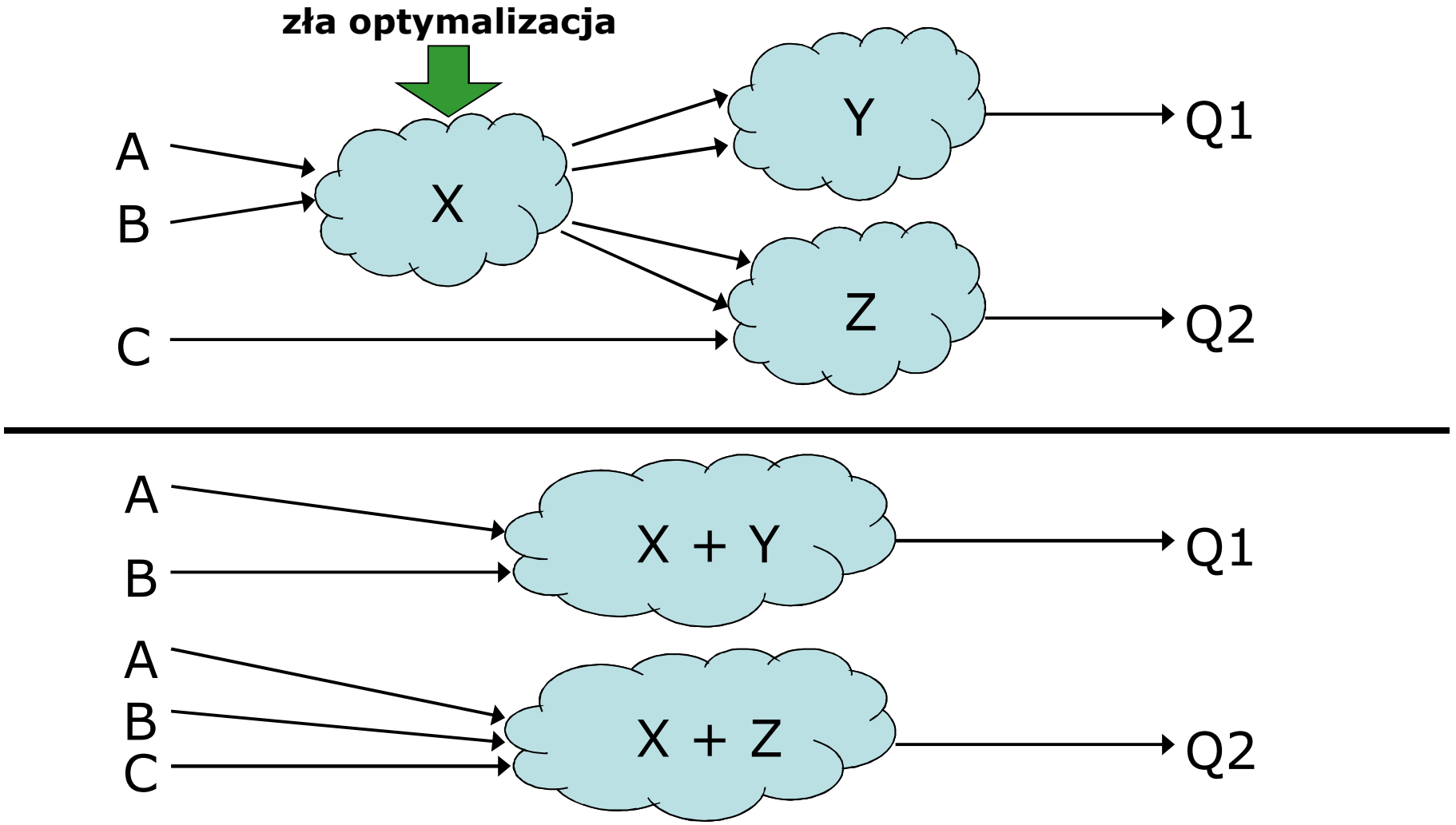
Statyczna analiza czasów



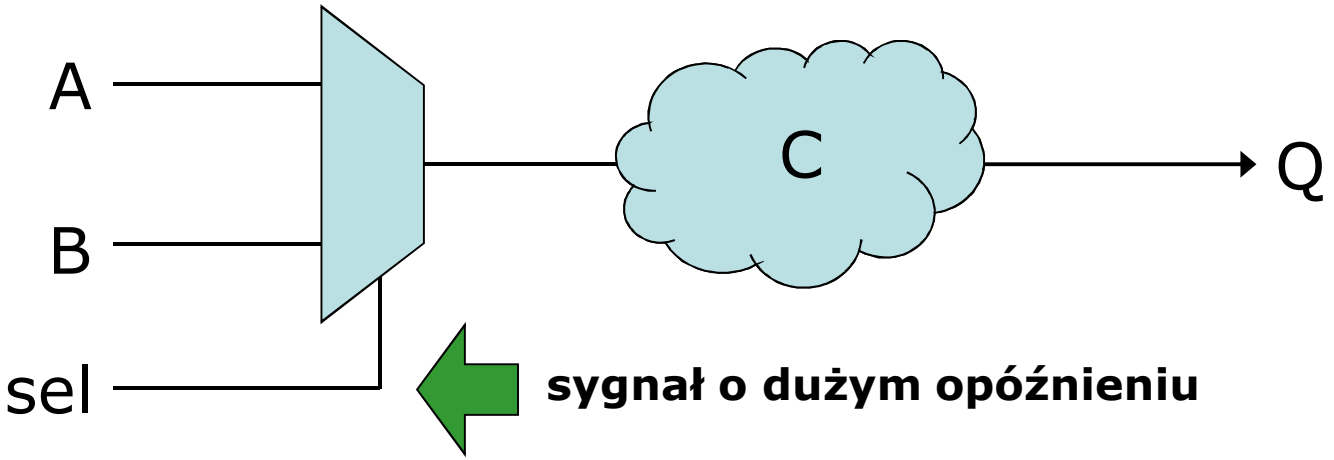
$$t_{prop} + t_{delay} < t_{clock} - t_{setup}$$

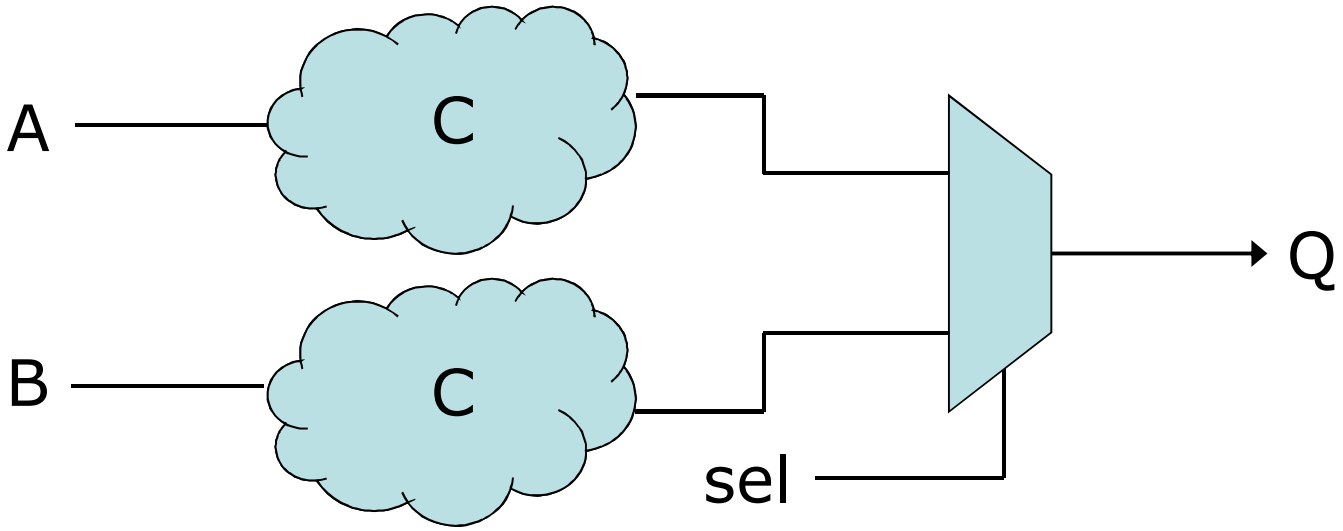
$$t_{prop} + t_{delay} > t_{hold}$$

Rozłączanie ścieżek przez duplikację logiki

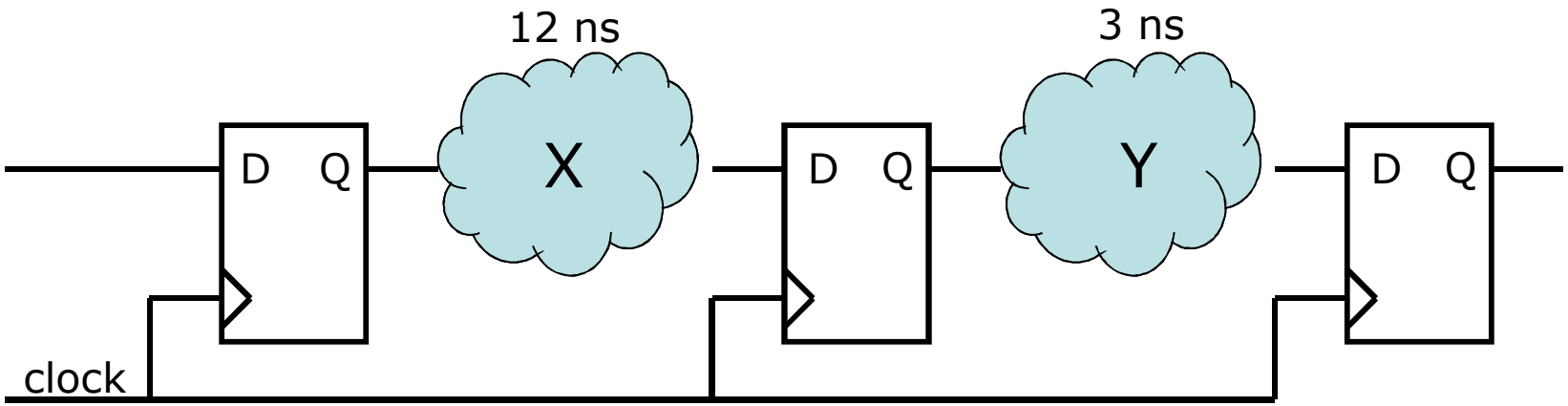


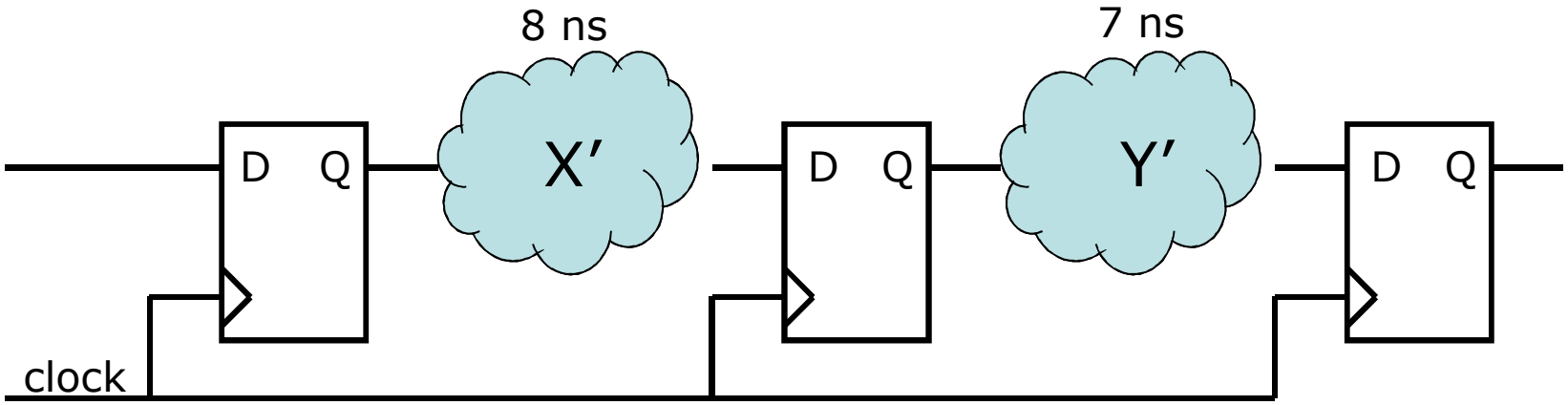
Duplikacja logiki przed selekcją





Równoważenie logiki między stopniami

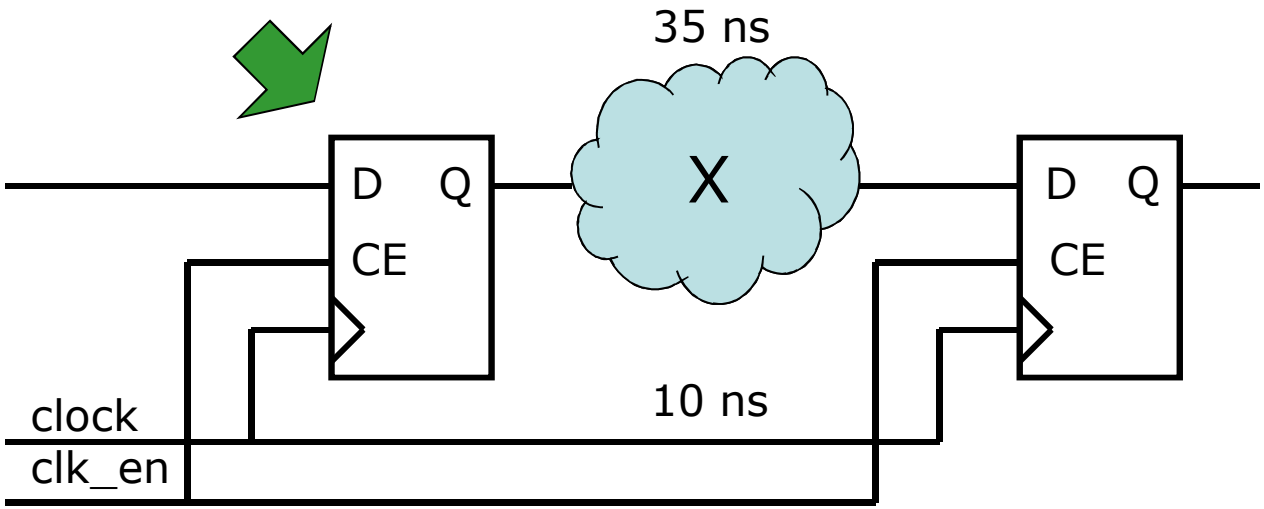




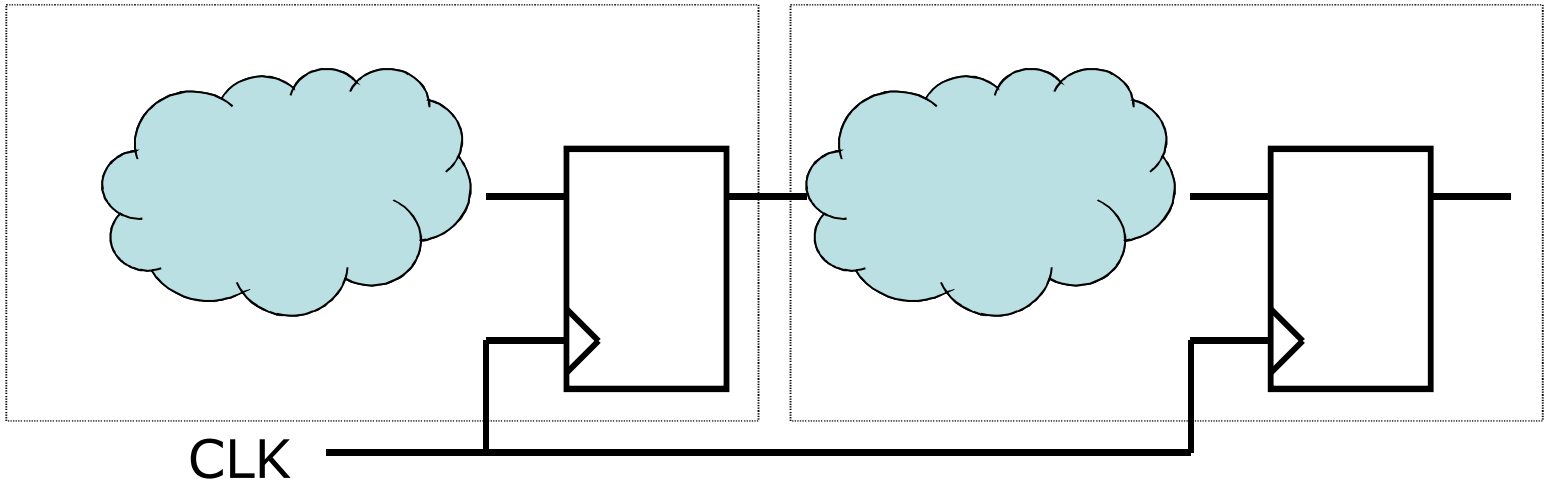
Ścieżki wielocyklowe i asynchroniczne

- ścieżki asynchroniczne
- ścieżki wielocyklowe

} informować narzędzie do syntezy !



Sygnały zegarowe



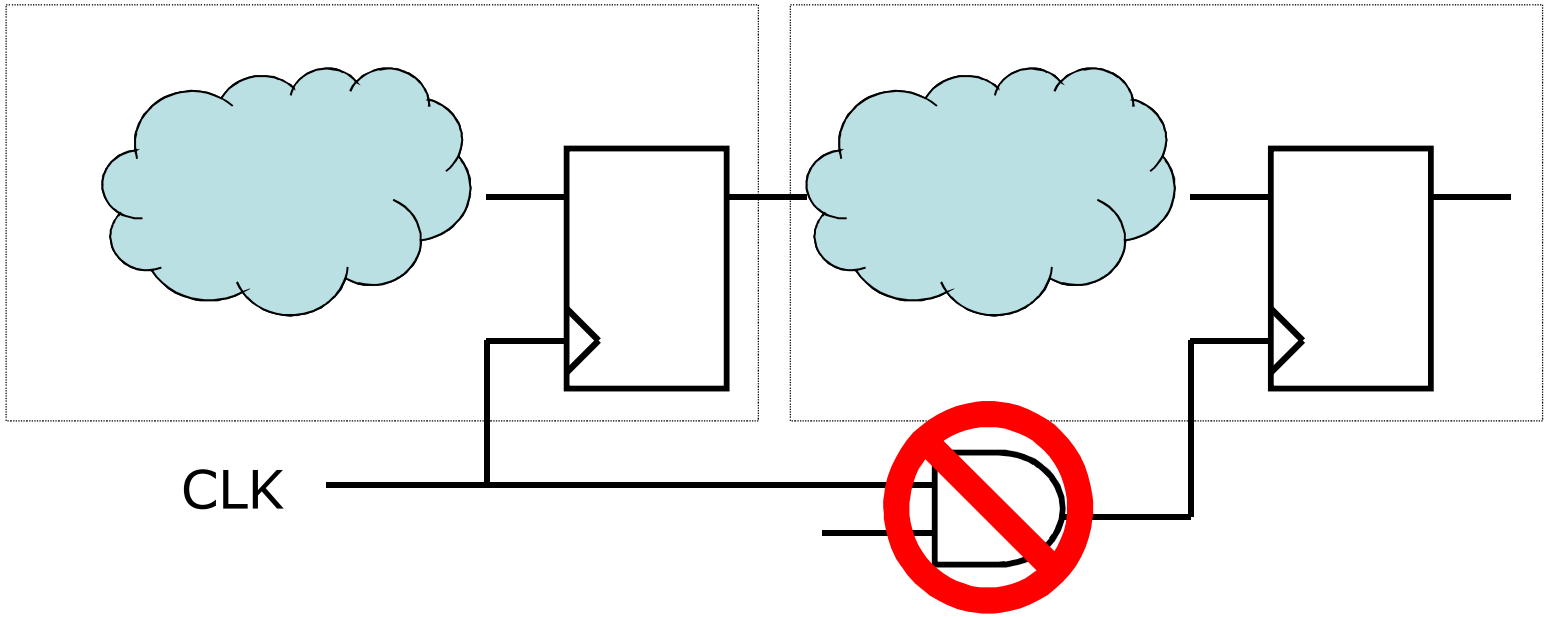
Modelowa struktura projektu:

- Jeden sygnał zegarowy
- Wszystkie przerzutniki wyzwalane tym samym zboczem

Problemy przy dwóch aktywnych zboczach:

- Zależność od współczynnika wypełnienia
  (tolerancja na zmiany współczynnika wypełnienia
  w dokumentacji projektu!)
- Problemy z testowaniem ścieżką brzegową (JTAG 1149)

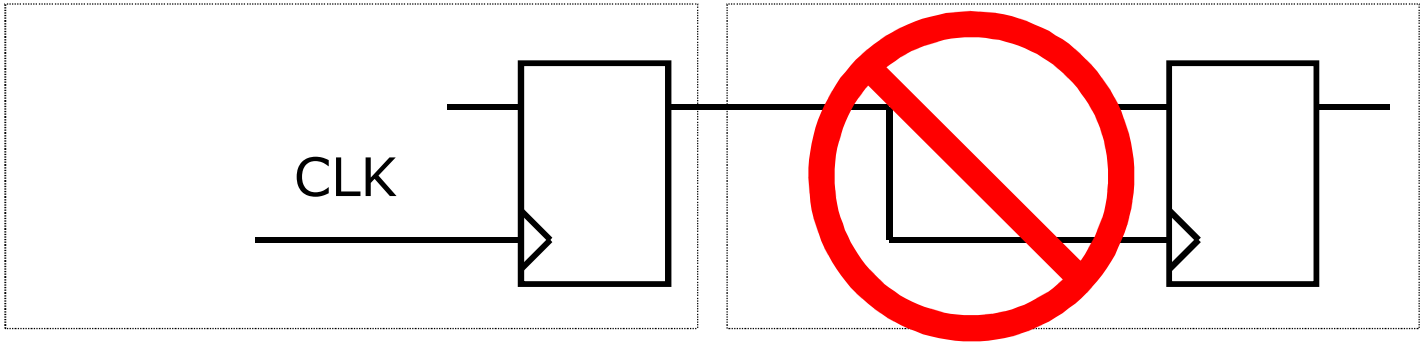
Sygnały zegarowe



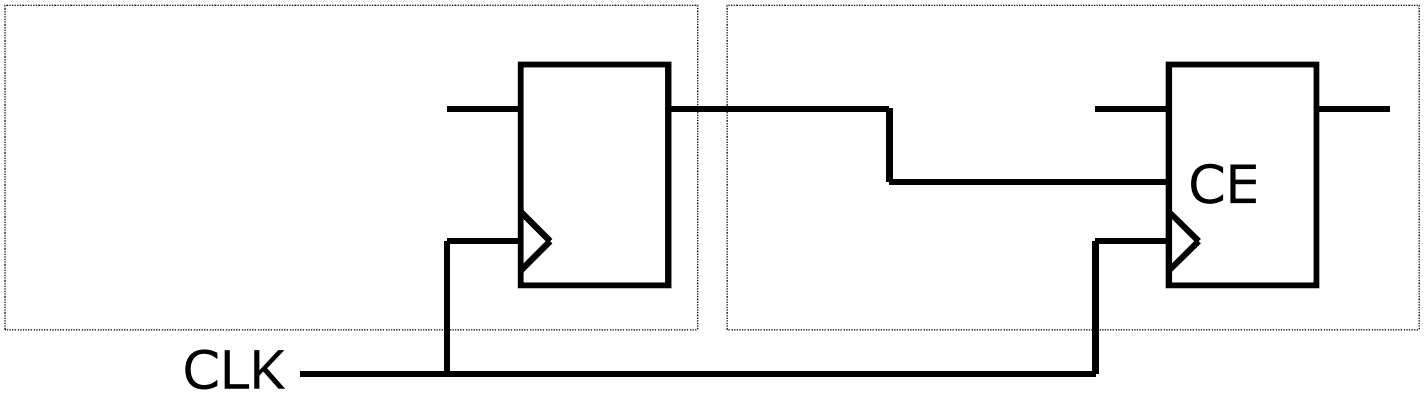
Asynchroniczne kluczowanie zegara - same problemy!
(niewykorzystanie zasobów dystrybucji sygnałów zegarowych, problemy z testowaniem, gorsze parametry czasowe, hazardy itp.)

Rozwiązanie – stosowanie wejść CE – kodowanie przez warunek:
`if (CE == 1)` po klauzuli `always (posedge CLK)`.

Sygnały zegarowe

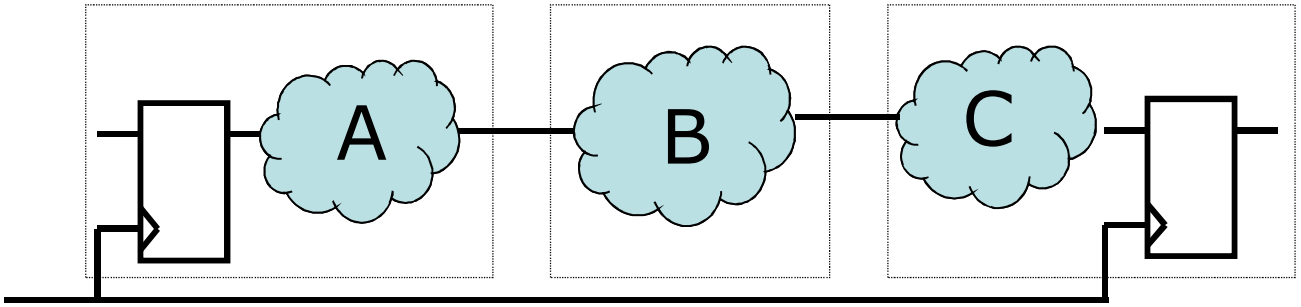


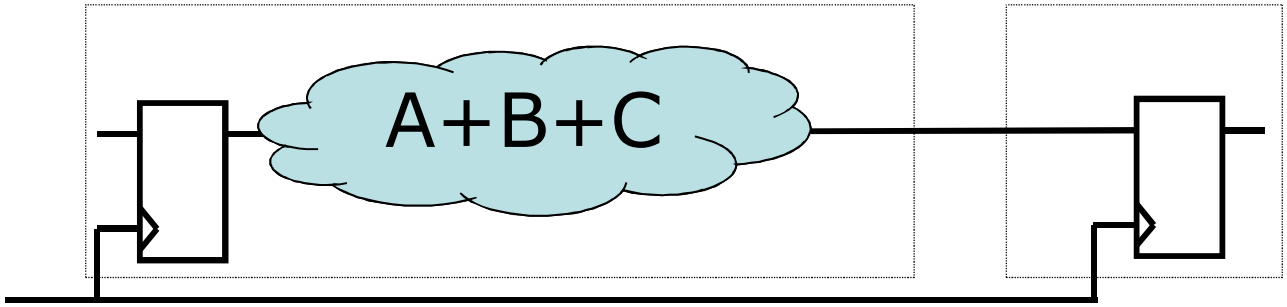
Nie należy stosować wewnętrznie (asynchronicznie!) generowanych sygnałów zegarowych.

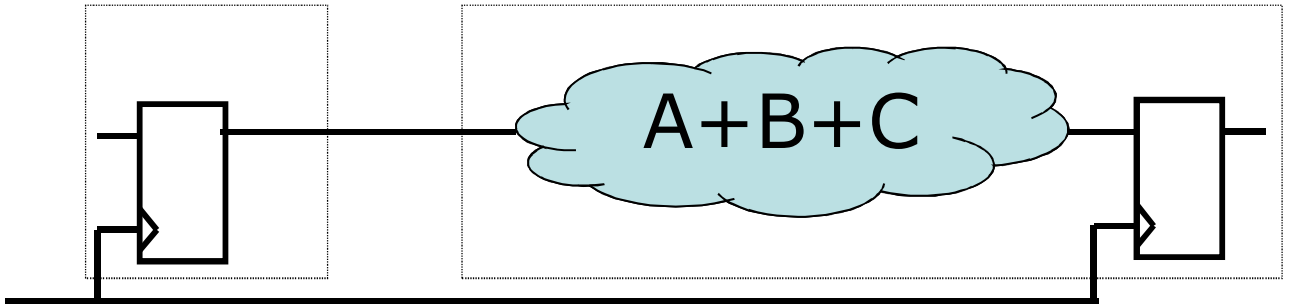


Należy natomiast projektować układy synchroniczne (w razie konieczności używać kilka sygnałów zegarowych w fazie – patrz: PLL/DLL).

Moduły







Styl projektowania
Uwzględnianie procesu syntezy

1. Rejestry wyjściowe dla sygnałów
2. Odpowiedni podział modułów logiki kombinacyjnej
3. Podział na bloki z uwagi na wymaganą szybkość działania *(speed versus area optimisation)*
4. Rozdzielenie bloków logiki asynchronicznej (jeśli już muszą być) od synchronicznej
5. Odpowiednie rozwiązania dla modułów realizujących funkcje arytmetyczne
6. Unikanie *„multicycle paths”* – czyli ścieżek sygnałów, które odchodzą od reguły „jednego okresu”
7. Eliminacja *glue logic* na *top level* – nieoptymalna minimalizacja

Styl kodowania

Struktura

- Jednolita struktura projektów
- Użycie zmiennych systemowych w ścieżkach do projektów
- Użycie przyrostków w nazwach plików dla określenia ich zawartości (np.: _tb, _beh, _struct, _rtl, _synt, _impl itp.)

Komentarze

- Nagłówek każdego pliku (skryptu też) zawiera: nazwę pliku, autora, opis funkcji, datę utworzenia, historię modyfikacji
- Komentarz opisuje intencje projektanta, a nie zachowanie się kodu
- Kasowanie złego kodu (zamiast zamiany na komentarz)

Układ

- Wcięcia na min. 3 spacje
- Jedna operacja na jedną linię kodu

Styl kodowania

Układ c.d.

- Limit długości linii (przeniesienie logiczne!)
- Użycie tabulatorów np. w klauzuli *port* deklaracji *entity*

Składnia

- Grupowanie często używanych operacji w podprogramy
- Używanie pętli i tablic
- Minimalizowanie liczby poziomów zagnieżdżenia instrukcji warunkowych

Kapitalizacja

- Nie dodawać znaczenia przez *uppercase* w narzędziach *case-insensitive*
- Zamiast tego używać przedrostków i przyrostków dla nazw
- Używać *underscore* zamiast *uppercase* dla oddzielania słów
- Używać *uppercase* tylko dla stałych (ew. typów)

Styl kodowania

Identyfikatory

- Używać znaczących nazw (min. 5 znaków)
- Nazywać obiekty stosownie do ich funkcji, a nie typu
- Nie wykorzystywać ograniczenia zakresu działania obiektów
- Wykorzystywać stałe symboliczne (nie „magiczne”)

Język VHDL

- Parametryzacja kodu (szerokość magistral, atrybuty np.: ‘left)
- W każdym pliku źródłowym tylko jedna jednostka projektowa: (*entity + architecture, configuration, package*)
- Deklaracja portów w porządku interfejsów, a nie ich kierunku
- Etykiety dla konstrukcji współbieżnych (**`begin`** / **`end`**)
- Nie tworzyć zbyt wielu podtypów (*range* zamiast *subtype*)
- Nie używać trybu *buffer* (sprzężenia zwrotne lokalne)
- Nie używać przypisań *guarded* i sygnału *guard*

Styl kodowania

- Używać nazwy **`clk`** dla sygnałów zegarowych i **`rst`** dla sygnałów kasowania
- Nie zmieniać nazwy tego samego sygnału w poszczególnych blokach projektowanego systemu
- Zaznaczać poziom aktywności sygnału, np. sygnały aktywne stanem niskim kończyć: **„`_n`”**
- Używać porządku *„downto”* w obiektach wektorowych
- Standard VITAL IEEE1076.4: Nie używać *„underscore”* w nazwach entity. Po syntezie jest on używany w pliku SDF.
- Konwencje nazw deklaracji *architecture:*
  `architecture rtl of my_syn_model is`
  dla symulacji po syntezie lub:
  `architecture sim of my_behave_model is`
  dla symulacji behawioranej.
- Przy podstawianiu modułów używać notacji specyfikacyjnej
- Stosować tylko typ *std_logic*

Dziękuję za uwagę!