# Języki Opisu Sprzętu

*Prowadzący: dr inż. Andrzej Skoczeń*

*Dodatek 1*

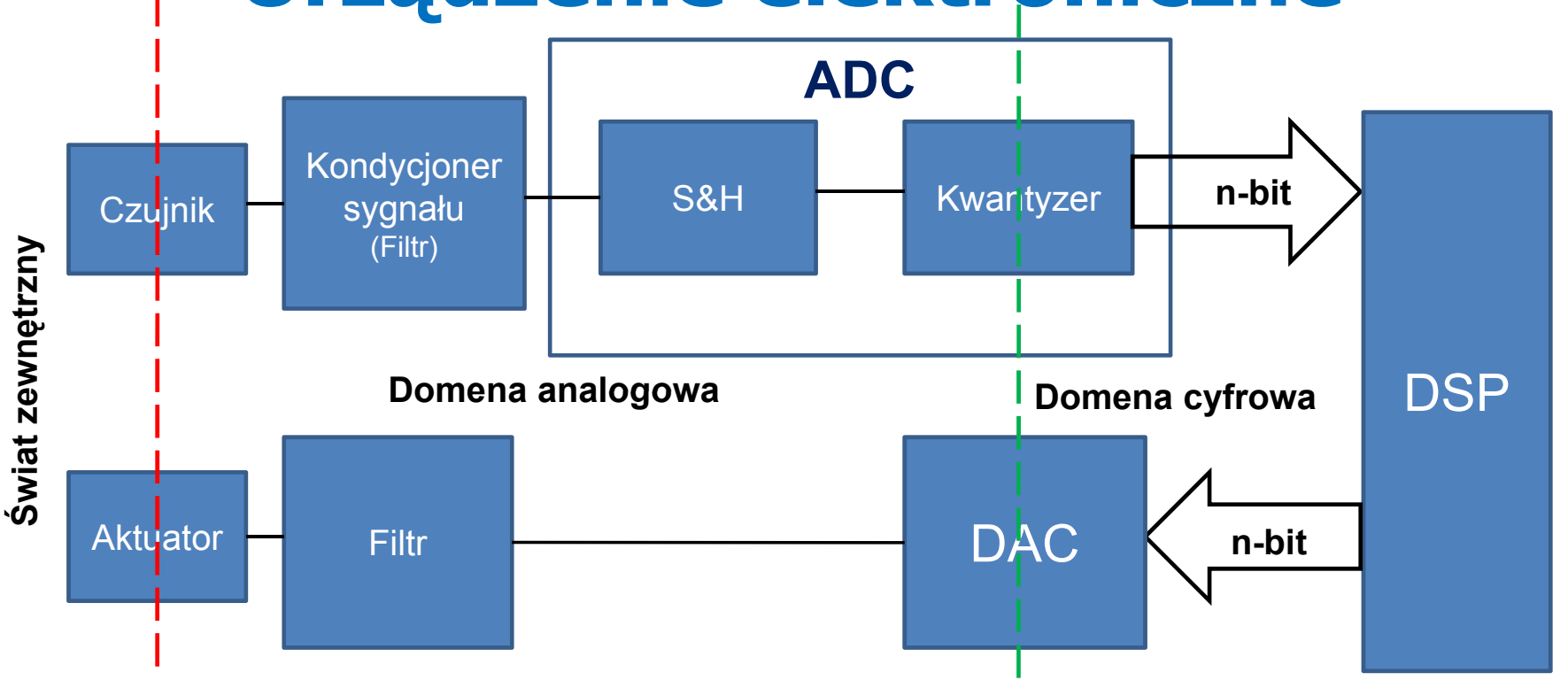
***2014***

*Październik 2014*

- **Próbkowanie i kwantowanie**
- **Podstawy techniki cyfrowej: algebra Boole'a, postać SOP i POS funkcji logicznej**
- **Minimalizacja funkcji logicznych**
- **Rozwinięcie Shannona**
- **System funkcjonalnie pełny**
- **Sekwencjonowanie układów**
- **Układ synchroniczny**

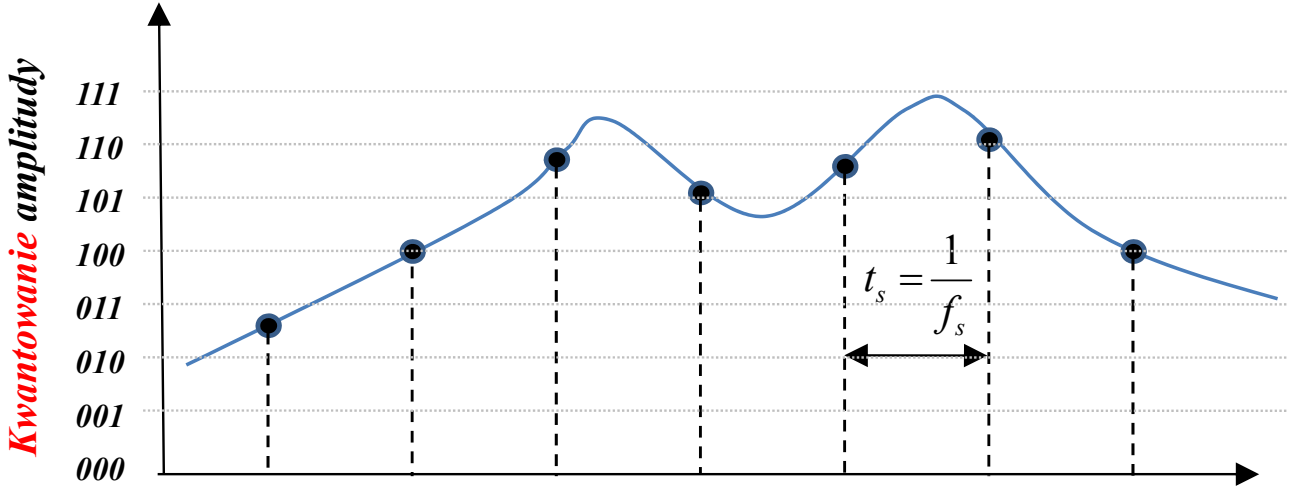
http://fatcat.ftj.agh.edu.pl/~skoczen/hdl

# Urządzenie elektroniczne



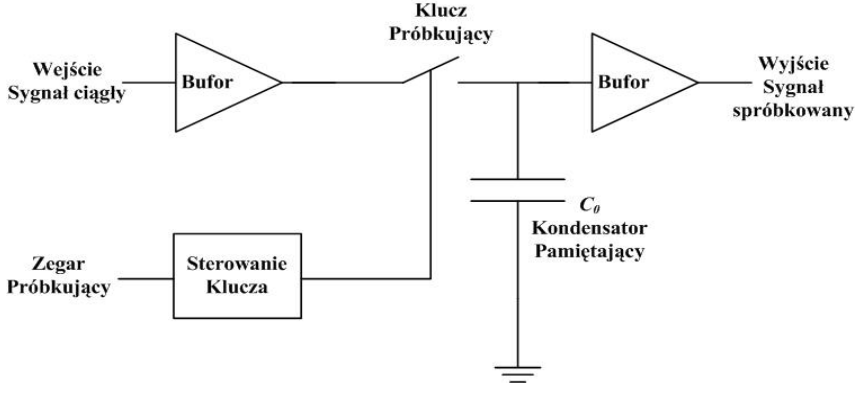

| | | |
|---|---|---|
| **S&H** | Sample and Hold | Układ próbkujaco-pamietajacy |
| **ADC** | Analog Digital Converter | Przetwornik analogowo-cyfrowy |
| **DAC** | Digital Analog Converter | Przetwornik cyfrowo-analogowy |
| **DSP** | Digital Signal Procesing | Cyfrowe przetwarzanie sygnałów |

# Próbkowanie i Kwantowanie




*Próbkowanie sygnału w czasie*

## Układ próbkująco-pamiętający



# Przetworniki analogowo-cyfrowe ADC

Trzy metody przetwarzania sygnału analogowego na cyfrowy:

- Z całkowaniem
- Bezpośrednie
- Z kolejnym porównaniem

# Przetworniki cyfrowo-analogowe DAC

Cztery rodzaje przetworników cyfrowo-analogowych DAC:

- Z napięciowymi źródłami odniesienia:
  - Z siecią rezystorów wagowych
  - Z drabinką rezystorów R-2R
- Z prądowymi źródłami odniesienia:
  - Z siecią rezystorów wagowych
  - Z drabinką rezystorów R-2R

# Aksjomaty algebry Boole'a

1854 George Boole sformułował dwu-wartościowy system algebraiczny

1938 Claude Shannon zastosował system do Boole'a do opisu obwdów zbudowanych z przekaźników

| | | |
|---|---|---|
| $x = 0$ jeśli $x \neq 1$ | Tylko dwie wartości | $x = 1$ jeśli $x \neq 0$ |
| $x = 0 \rightarrow x' = 1$ | Element odwrotny | $x = 1 \rightarrow x' = 0$ |
| $0 \cdot 0 = 0$ | Formalna definicja operacji koniunkcji i alternatywy | $1 + 1 = 1$ |
| $1 \cdot 1 = 1$ | | $0 + 0 = 0$ |
| $0 \cdot 1 = 1 \cdot 0 = 0$ | | $1 + 0 = 0 + 1 = 1$ |

# Zasada dualności

W dowolnej tożsamości algebry Boole'a zastępując
- symbol + (OR) symbolem • (AND) i na odwrót oraz
- zastępując jedynkę „1" zerem „0" i na odwrót

otrzymamy również tożsamość.

$$f^D(x_1, x_2, \ldots x_n, +, \bullet, ') = f(x_1, x_2, \ldots x_n, \bullet, +, ')$$

# Podstawowe własności algebry Boole'a

| | | |
|---|---|---|
| x + 0 = x | Element identycznościowy | x · 1 = x |
| x + 1 = 1 | Element zerowy | x · 0 = 0 |
| x + x =x | Idempotencja | x · x = x |
| x + x' = 1 | Komplementarność | x · x' = 0 |
| x'' = x | Inwolucja | |
| x + y = y + x | Przemienność | x · y = y · x |
| x + (y + z) = (x + y) + z | Łączność | x · (y · z) = ( x · y) · z |
| x · (y + z) = xy + xz | Rozdzielność | x + (y · z) = (x + y) ·(x + z) |
| x + xy = x | Przekrywanie | x · (x + y) = x |
| xy + xy' = x | Kombinacja | (x+y) · (x+y') = x |
| xy + x'z + yz = xy + x'z | Zgoda | (x+y) · (x'+z) · (y+z) = (x+y) · (x'+z) |

Czynnik yz jest czynnikiem zgody między czynnikami xy i x'z.
Jeśli yz=1 to albo xy albo x'z musi także być 1 ponieważ y i z są równe 1, a albo x albo x' jest równe 1.
Tak więc czynnik yz jest nadmiarowy i może być opuszczony.

# Prawa de Morgana

$$\overline{\sum_i x_i} = \prod_i \overline{x_i}$$

$$\overline{\prod_i x_i} = \sum_i \overline{x_i}$$

Uogólnione prawo de Morgana:

$$\overline{f(x_1, x_2, \ldots, x_n, +, \bullet)} = f(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}, \bullet, +)$$

Uogólnione prawo de Morgana zapisane za pomocą zasady dualności:

$$\overline{f(x_1, x_2, \ldots, x_n)} = f^D(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$$

# Funkcja logiczna

Funkcja logiczna o $n$ wejściach (argumentach) i $m$ wyjściach (wartościach) jest to odwzorowanie dwóch przestrzeni logicznych:

$$f : \mathbf{B}^n \rightarrow \mathbf{B}^m \qquad \text{gdzie } \mathbf{B} = \{0, 1\}$$

Jeżeli $m=1$ to funkcja logiczna jest skalarna.

Funkcja logiczna może mieć nieokreśloności tzn. nigdy nie wytępujące kombinacje zmiennych wejściowych, dla których wyjście nie ma określonej wartości.

Funkcję logiczną nie wszędzie określona zapisujemy więc:

$$f : \mathbf{B}^n \rightarrow \{0,1,-\}^m$$ gdzie - to warunek zaniedbywalny

Dla każdego wyjścia podzbiory stanów wejść, dla których wyjście ma 0, 1, - nazywa się zbiorami wyłączenia, włączenia i nieokreśloności.

# Funkcja logiczna

| | x | y | z | F(x,y,z) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | X |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | X |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | X |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | X |

$$F(x,y,z) = \sum(0,4,6) + \sum n(1,3,5,7)$$

# Funkcja logiczna - SOP

Funkcja logiczna (boolowska) jest to wyrażenie algebraiczne utworzone ze zmiennych dwójkowych, symboli operacji logicznych, nawiasów i znaku równości.

Dowolną funkcję logiczną $n$ zmiennych można jednoznacznie przedstawić w postaci sumy iloczynów pełnych (kanoniczna postać sumy, SOP – Sum Of Products):

$$f(x_1, x_2, ..., x_n) = \sum_{i=0}^{2^n-1} \alpha_i m_i \qquad \alpha_i = f(i)$$

gdzie $f(i)$ oznacza wartość funkcji $f(x_1, x_2, ..., x_n)$ dla $i$-tej kombinacji zmiennych.

Iloczynem pełnym $n$ zmiennych nazywamy taki iloczyn tych zmiennych (bezpośrednich lub zanegowanych), w którym każda zmienna występuje dokładnie jeden raz.

Tabela przedstawia iloczyny pełne dla $n=3$.
Przyjęto konwencję, w której bezpośredniej zmiennej odpowiada wartość *1* a zmiennej zanegowanej wartość *0*.

| x y z | Iloczyny pełne | $m_i$ |
|---|---|---|
| 0 0 0 | x'y'z' | $m_0$ |
| 0 0 1 | x'y'z | $m_1$ |
| 0 1 0 | x'yz' | $m_2$ |
| 0 1 1 | x'yz | $m_3$ |
| 1 0 0 | xy'z' | $m_4$ |
| 1 0 1 | xy'z | $m_5$ |
| 1 1 0 | xyz' | $m_6$ |
| 1 1 1 | xyz | $m_7$ |

# Przykład - SOP

| *i* | *x* | *y* | *z* | *f(x, y, z) = x + yz'* |
|---|---|---|---|---|
| *0* | *0* | *0* | *0* | *0* |
| *1* | *0* | *0* | *1* | *0* |
| *2* | *0* | *1* | *0* | *1* |
| *3* | *0* | *1* | *1* | *0* |
| *4* | *1* | *0* | *0* | *1* |
| *5* | *1* | *0* | *1* | *1* |
| *6* | *1* | *1* | *0* | *1* |
| *7* | *1* | *1* | *1* | *1* |

$$f(x,y,z)=\sum_{i=0}^{7}\alpha_i m_i =$$
$$0\cdot m_0+0\cdot m_1+1\cdot m_2+0\cdot m_3+$$
$$+1\cdot m_4+1\cdot m_5+1\cdot m_6+1\cdot m_7$$

$$f(x,y,z)=m_2+m_4+m_5+m_6+m_7=$$
$$x'yz'+xy'z'+xy'z+xyz'+xyz$$

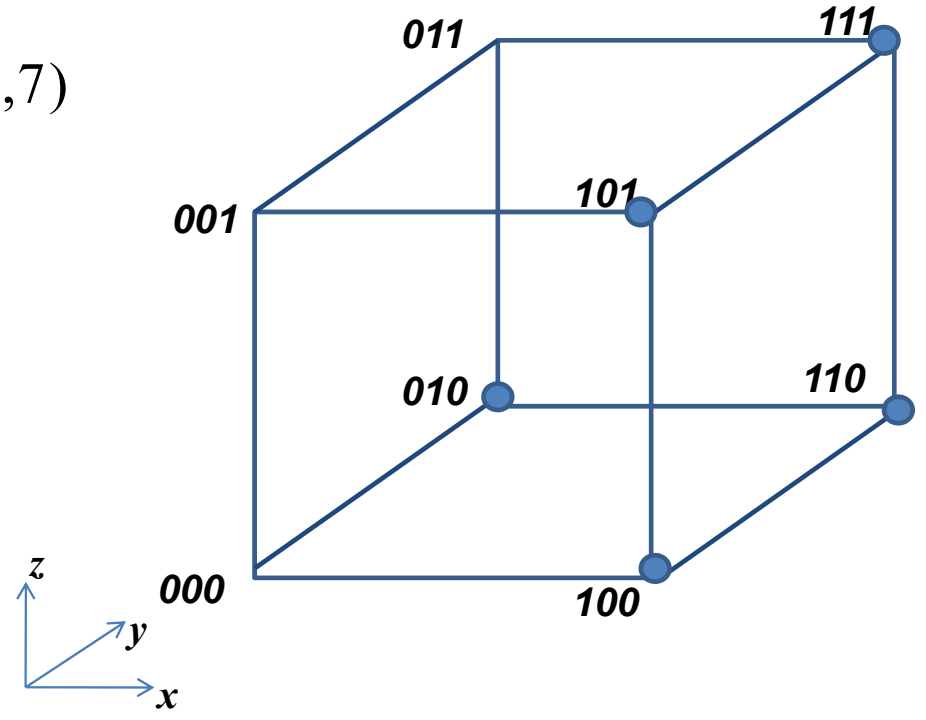
$$f(x,y,z)=\sum(2,4,5,6,7)$$

Iloczyny pełne, dla których funkcja przyjmuje wartość *1* nazywamy **minitermami**.
Funkcję logiczną można interpetować jak zbiór jej minitermów.
Działania na funkcjach logicznych można interpretować jako działania na zbiorach minitermów.

# Przykład - SOP

W przestrzeni $\mathbf{B}^3$ funkcję logiczną trzech zmiennych możemy przedstawić w postaci kostki logicznej:

$$f(x, y, z) = \sum (2,4,5,6,7)$$



# Funkcja logiczna - POS

Dualnie udowodnić można twierdzenie:

Dowolną funkcję logiczną $n$ zmiennych można jednoznacznie przedstawić w postaci iloczynu sum pełnych (kanoniczna postać iloczynu, POS –Product Of Sums):

$$f(x_1, x_2, ..., x_n) = \prod_{i=0}^{2^n-1} (\alpha_i + M_i) \qquad \alpha_i = f(i)$$

gdzie $f(i)$ oznacza wartość funkcji $f(x_1, x_2, ..., x_n)$ dla $i$-tej kombinacji zmiennych.

Sumą pełną $n$ zmiennych nazywamy taką sumę tych zmiennych (bezpośrednich lub zanegowanych), w którym każda zmienna występuje dokładnie jeden raz.

Tabela przedstawia sumny pełne dla $n=3$. Przyjęto konwencję, w której bezpośredniej zmiennej odpowiada wartość $0$ a zmiennej zanegowanej wartość $1$.

| x y z | Sumy pełne | $m_i$ |
|---|---|---|
| 0 0 0 | x+ y+ z | $M_0$ |
| 0 0 1 | x+y+z' | $M_1$ |
| 0 1 0 | x+y' +z | $M_2$ |
| 0 1 1 | x+y'+z' | $M_3$ |
| 1 0 0 | x' +y+z | $M_4$ |
| 1 0 1 | x'+y+z' | $M_5$ |
| 1 1 0 | X'+y'+z | $M_6$ |
| 1 1 1 | x'+y'+z' | $M_7$ |

# Przykład - POS

| *i* | *x* | *y* | *z* | *f(x, y, z) = x + yz'* |
|---|---|---|---|---|
| *0* | *0* | *0* | *0* | *0* |
| *1* | *0* | *0* | *1* | *0* |
| *2* | *0* | *1* | *0* | *1* |
| *3* | *0* | *1* | *1* | *0* |
| *4* | *1* | *0* | *0* | *1* |
| *5* | *1* | *0* | *1* | *1* |
| *6* | *1* | *1* | *0* | *1* |
| *7* | *1* | *1* | *1* | *1* |

$$f(x,y,z) = \prod_{i=0}^{7} (\alpha_i + M_i) =$$
$$(0 + M_0) \cdot (0 + M_1) \cdot (1 + M_2) \cdot (0 + M_3) \cdot$$
$$\cdot (1 + M_4) \cdot (1 + M_5) \cdot (1 + M_6) \cdot (1 + M_7)$$

$$f(x,y,z) = M_0 \cdot M_1 \cdot M_3 =$$
$$= (x + y + z) \cdot (x + y + z') \cdot (x + y' + z')$$

$$f(x,y,z) = \prod (0,1,3)$$

Sumy pełne, dla których funkcja przyjmuje wartość *0* nazywamy **maxtermami**.
Funkcję logiczna można interpetować jak zbiór jej maxtermów.
Działania na funkcjach logicznych można interpretować jako działania na zbiorach maxtermów.

# Przykład

| *i* | *x* | *y* | *z* | *f(x, y, z)* |
|---|---|---|---|---|
| *0* | *0* | *0* | *0* | *0* |
| *1* | *0* | *0* | *1* | *0* |
| *2* | *0* | *1* | *0* | *1* |
| *3* | *0* | *1* | *1* | *0* |
| *4* | *1* | *0* | *0* | *1* |
| *5* | *1* | *0* | *1* | *1* |
| *6* | *1* | *1* | *0* | *1* |
| *7* | *1* | *1* | *1* | *1* |

$$f(x,y,z) = \sum (2,4,5,6,7)$$

$$f(x,y,z) =$$

$$x'yz' + xy'z' + xy'z + xyz' + xyz$$

$$f(x,y,z) = x'yz' + xy'(z+z') + xy(z'+z) =$$

$$x'yz' + xy' + xy = x'yz' + x(y'+y) = x'yz' + x$$

# Minimalizacja funkcji logicznych

**Metody:**

- **przekształceń formalnych**
- **tablic Karnaugha**
- **Quine'a-McCluskeya**

## Przykład dla metody przekształceń formalnych

$$f(x,y,z) = x'yz + x'yz' + xz =$$
$$= x'y(z+z') + xz = x'y \cdot 1 + xz = x'y + xz$$

# Przykład dla metody tablic Karnaugha (1)

$$f(x,y,z) = \sum(2,4,5,6,7) = \prod(0,1,3)$$

| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Interpretując tablice w sensie SOP:

$$f(x,y,z) = x + yz'$$

$$f' = x'y' + x'z$$

| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Interpretując tablice w sensie POS:

$$f(x,y,z) = (y + x)\cdot(z' + x)$$

$$f' = x'\cdot(y' + z)$$

# Przykład dla metody tablic Karnaugha (2)

$$f(x,y,z) = \sum(0,4,6) + \sum n(1,3,5,7)$$

| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 x | 0 x | 0 |
| 1 | 1 | 1 x | 1 x | 1 |

Interpretując tablice w sensie SOP:

$$f(x,y,z) = y' + x \qquad f' = x'y$$

| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 x | 0 x | 0 |
| 1 | 1 | 1 x | 1 x | 1 |

Interpretując tablice w sensie POS:

$$f(x,y,z) = y' + x \qquad f' = x' \cdot y$$

# Przykład dla metody Quine'a-McCluskeya

$$f(a,b,c,d) = \sum (5,7,8,9,10,11,13,15)$$

Metoda implikantów prostych

| i | a | b | c | d | f | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | 0 | |
| 2 | 0 | 0 | 1 | 0 | 0 | |
| 3 | 0 | 0 | 1 | 1 | 0 | |
| 4 | 0 | 1 | 0 | 0 | 0 | |
| 5 | 0 | 1 | 0 | 1 | 1 | Dwie „1” |
| 6 | 0 | 1 | 1 | 0 | 0 | |
| 7 | 0 | 1 | 1 | 1 | 1 | Trzy „1” |
| 8 | 1 | 0 | 0 | 0 | 1 | Jedna „1” |
| 9 | 1 | 0 | 0 | 1 | 1 | Dwie „1” |
| 10 | 1 | 0 | 1 | 0 | 1 | Dwie „1” |
| 11 | 1 | 0 | 1 | 1 | 1 | Trzy „1” |
| 12 | 1 | 1 | 0 | 0 | 0 | |
| 13 | 1 | 1 | 0 | 1 | 1 | Trzy „1” |
| 14 | 1 | 1 | 1 | 0 | 0 | |
| 15 | 1 | 1 | 1 | 1 | 1 | Cztery „1” |

Tablica minitermów funkcji $f$ uporządkowana według ilości jedynek:

| i | a | b | c | d | |
|---|---|---|---|---|---|
| 8 | 1 | 0 | 0 | 0 | Jedna „1” |
| 5 | 0 | 1 | 0 | 1 | Dwie „1” |
| 9 | 1 | 0 | 0 | 1 | |
| 10 | 1 | 0 | 1 | 0 | |
| 7 | 0 | 1 | 1 | 1 | Trzy „1” |
| 11 | 1 | 0 | 1 | 1 | |
| 13 | 1 | 1 | 0 | 1 | |
| 15 | 1 | 1 | 1 | 1 | Cztery „1” |

# Przykład dla metody Quine'a-McCluskeya

| i | a | b | c | d |
|---|---|---|---|---|
| 8 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 11 | 1 | 0 | 1 | 1 |
| 13 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 |

Każdą kombinację z jednej grupy porównujemy z kominacjami grupy następnej szukając takie, które różnią się tylko jedną pozycją. Powstaje nowa tabela z znakiem „-" na pozycji rozbieżności:

| i,j | a | b | c | d |
|---|---|---|---|---|
| 8.9 | 1 | 0 | 0 | - |
| 8,10 | 1 | 0 | - | 0 |
| 5,7 | 0 | 1 | - | 1 |
| 5,13 | - | 1 | 1 | 1 |
| 9,11 | 1 | 0 | - | 1 |
| 9,13 | 1 | - | 0 | 1 |
| 10,11 | 1 | 0 | 1 | - |
| 7,15 | - | 1 | 1 | 1 |
| 11,15 | 1 | - | 1 | 1 |
| 13,15 | 1 | 1 | - | 1 |

# Przykład dla metody Quine'a-McCluskeya

Kolejny krok łączenia:

| i,j | a | b | c | d |
|---|---|---|---|---|
| 8.9 | 1 | 0 | 0 | - |
| 8,10 | 1 | 0 | - | 0 |
| 5,7 | 0 | 1 | - | 1 |
| 5,13 | - | 1 | 1 | 1 |
| 9,11 | 1 | 0 | - | 1 |
| 9,13 | 1 | - | 0 | 1 |
| 10,11 | 1 | 0 | 1 | - |
| 7,15 | - | 1 | 1 | 1 |
| 11,15 | 1 | - | 1 | 1 |
| 13,15 | 1 | 1 | - | 1 |

| i,j | a | b | c | d |
|---|---|---|---|---|
| 8.9,10,11 | 1 | 0 | - | - |
| 5,7,13,15 | - | 1 | - | 1 |
| 9,11,13,15 | 1 | - | - | 1 |

Kombinacje nie podlegajace dalszemu łączeniu to implikanty proste.

| | 5 | 7 | 8 | 9 | 10 | 11 | 13 | 15 | |
|---|---|---|---|---|---|---|---|---|---|
| 8.9,10,11 | | | • | • | • | • | | | ab' |
| 5,7,13,15 | • | • | | | | | • | • | bd |
| 9,11,13,15 | | | | • | | • | • | • | ad |

Funkcja minimalna jest sumą minimalnej liczby wybranych implikantów prostych, ale taką która zawiera wszystkie iloczyny pełne funkcji wyjściowej.

$$f(a,b,c,d) = ab'+bd$$

# Przykład dla metody Quine'a-McCluskeya

$$f(a,b,c,d) = \sum(0,1,4,6,8,12,14) + \\ + \sum n(5,10,13,15)$$

Tablica minitermów funkcji $f$ przy założeniu, że wszystkie nieokreślone iloczyny pełne należą do minitermów. Uporządkowana według ilości jedynek:

| *i* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 |

Pierwszy krok łączenia:

| *i* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|
| 0,1 | 0 | 0 | 0 | - |
| 0,4 | 0 | - | 0 | 0 |
| 0,8 | - | 0 | 0 | 0 |
| 1,5 | 0 | - | 0 | 1 |
| 4,5 | 0 | 1 | 0 | - |
| 4,6 | 0 | 1 | - | 0 |
| 4,12 | - | 1 | 0 | 0 |
| 8,10 | 1 | 0 | - | 0 |
| 8,12 | 1 | - | 0 | 0 |
| 5.13 | - | 1 | 0 | 1 |
| 6,14 | - | 1 | 1 | 0 |
| 10,14 | 1 | - | 1 | 0 |
| 12,13 | 1 | 1 | 0 | - |
| 12,14 | 1 | 1 | - | 0 |
| 13,15 | 1 | 1 | - | 1 |
| 14,15 | 1 | 1 | 1 | - |

# Przykład dla metody Quine'a-McCluskeya

| *i* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|
| 0,1 | 0 | 0 | 0 | - |
| 0,4 | 0 | - | 0 | 0 |
| 0,8 | - | 0 | 0 | 0 |
| 1,5 | 0 | - | 0 | 1 |
| 4,5 | 0 | 1 | 0 | - |
| 4,6 | 0 | 1 | - | 0 |
| 4,12 | - | 1 | 0 | 0 |
| 8,10 | 1 | 0 | - | 0 |
| 8,12 | 1 | - | 0 | 0 |
| 5.13 | - | 1 | 0 | 1 |
| 6,14 | - | 1 | 1 | 0 |
| 10,14 | 1 | - | 1 | 0 |
| 12,13 | 1 | 1 | 0 | - |
| 12,14 | 1 | 1 | - | 0 |
| 13,15 | 1 | 1 | - | 1 |
| 14,15 | 1 | 1 | 1 | - |

Drugi krok łączenia:

| *i* | *a* | *b* | *c* | *d* |
|---|---|---|---|---|
| 0,1,4,5 | 0 | - | 0 | - |
| 0,8,4,12 | - | - | 0 | 0 |
| 4,5,12,13 | - | 1 | 0 | - |
| 4,6,12,14 | - | 1 | - | 0 |
| 8,10,12,14 | 1 | - | - | 0 |
| 12,13,14,15 | 1 | 1 | - | - |

*Lista implikantów prostych*

$\overline{a} \cdot \overline{c}$

$\overline{c} \cdot \overline{d}$

$b \cdot \overline{c}$

$b \cdot \overline{d}$

$a \cdot \overline{d}$

$a \cdot b$

# Przykład dla metody Quine'a-McCluskeya

| i | a | b | c | d |
|---|---|---|---|---|
| 0,1,4,5 | 0 | - | 0 | - |
| 0,8,4,12 | - | - | 0 | 0 |
| 4,5,12,13 | - | 1 | 0 | - |
| 4,6,12,14 | - | 1 | - | 0 |
| 8,10,12,14 | 1 | - | - | 0 |
| 12,13,14,15 | 1 | 1 | - | - |

$$f(a,b,c,d) = \sum(0,1,4,6,8,12,14) + \sum n(5,10,13,15)$$

| i | 0 | 1 | 4 | 6 | 8 | 12 | 14 | |
|---|---|---|---|---|---|---|---|---|
| 0,1,4,5 | • | • | • | | | | | a'c' |
| 0,8,4,12 | • | | • | | • | • | | c'd' |
| 4,5,12,13 | | | • | | | • | | bc' |
| 4,6,12,14 | | | • | • | | • | • | bd' |
| 8,10,12,14 | | | | | • | • | • | ad' |
| 12,13,14,15 | | | | | | • | • | ab |

Są dwie możliwości:

$$f(a,b,c,d) = a'c' + bd' + ad'$$

$$f(a,b,c,d) = a'c' + bd' + c'd'$$

# Implikant formuły logicznej

Formułę $f$ nazywamy implikantem formuły $F$ gdy:

$$(f \rightarrow F) = 1 \quad \text{albo} \quad \bar{f} + F = 1$$

Implikantami formuły kanonicznej SOP są wszystkie minitermy i ich połączenia.

Formułę $f^*$ nazywamy implikantem prostym formuły $F$ gdy $f^*$ jest implikantem formuły $F$ oraz nie istnieje formuła $f^{**}$ taka, że

$$(f^* \rightarrow f^{**}) = 1 \quad \text{oraz} \quad (f^{**} \rightarrow F) = 1$$

Dla formuły: $xyz'+xyz+x'y'z'$
Implikantami są wszystkie trzy występujące w niej minitermy, a implikantami prostymi są: $xy$, $x'y'z'$ .

# Metoda Quine'a-McCluskeya

**Wyznaczenie zbioru implikantów prostych:**

- Wypisujemy minitermy funkcji czyli kombinacje zer i jedynek argumentów odpowiadające wszystkim iloczynom pełnym formuły, dla których wartość formuły jest *1* i indeksujemy je ich wartościami dziesiętnymi.
- Kombinacje te szeregujemy według ilości występujących w nich jedynek formując grupy z $n=0, 1, 2, \ldots$ jedynek.
- Porównujemy każdą kombinację $i$-tej grupy z każdą kombinacją grupy $i+1$ poszukując takich par, które różnią się tylko jedną pozycją. Pary takie łączymy w jedną nową kombinację stawiając znak "–" na pozycji gdzie występuje rozbieżność.
- Kontynuujemy procedurę w kolejnych krokach zwiększając liczbę znaków "–" w kombinacjach.
- Procedurę kończymy gdy nie ma już możliwości dokonywania dalszych połączeń.
- Końcowy zestaw kombinacji uzupełniony o te kombinacje, które nie wzięły udziału w procedurze łączenia tworzy zbiór implikantów prostych.

**Wyznaczenie minimalnego pokrycia:**

- Tworzymy tabelę implikantów prostych: kolumny to minitermy, wiersze to implikanty proste. W każdym wierszu znaczymy te kolumny, których minitermy były wykorzystane w procedurze tworzącej ten impliakant prosty.
- Na podstawie tej tabeli wybieramy minimalny zestaw implikantów prostych, który zpewnia pokrycie wszystkich kolumn tabeli.

# Minimalizacja funkcji logicznych

**Optymalizacja dwupoziomowa:**

Metody dokładne:
- Quine'a-McCluskeya
- ESPRESSO-EXACT
- McBOOLE
- ESPRESSO-SIGNATURE

Metody heurystyczne:
- MINI
- RESTO
- ESPRESSO
- CAPPUCCINO

**Optymalizacja wielopoziomowa:**

Metody dokładne:

Metody heurystyczne:
- MIS
- BOLD
- LLS
- SIS

Praktyczne znaczenie mają dziś metody wielopoziomowe, heurystyczne.

# Rozwinięcie Shannona

Każdą fukcję logiczna $n$ zmiennych można rozwinąć względem dowolnej zmiennej $x_i$ w nastepujący sposób:

$$f(x_1,\ldots,x_n) = x_i \cdot f_{x_i} + \overline{x_i} \cdot f_{\overline{x_i}}$$

i dualnie

$$f(x_1,\ldots,x_n) = (x_i + f_{x_i}) \cdot (\overline{x_i} + f_{\overline{x_i}})$$

gdzie współczynniki rozwinięcia zwane kofaktorami (dopełnieniami algebraicznymi) funkcji $f$, są funkcjami $n$-$1$ zmiennych powstałymi z funkcji $f$ przez zastąpienie zmiennej $x_i$ odpowiednio wartościami jeden „1” i zero „0”:

$$f_{x_i} = f(x_i = 1) \qquad f_{\overline{x_i}} = f(x_i = 0)$$

Przykład:

$$f(a,x) = \overline{x}a + x$$

$$f_a = \overline{x} + x$$

$$f_{\overline{a}} = x$$

$$f(a,x) = f_a a + f_{\overline{a}}\overline{a} =$$

$$= (\overline{x} + x)a + x\overline{a} =$$

$$= \overline{x}a + x(a + \overline{a}) = \overline{x}a + x$$

Rozwijając funkcje rekurencyjnie dochodzimy do jej postaci SOP (lub dualnie POS).

# System funkcjonalnie pełny

Systemem funkcjonalnie pełnym nazywamy system operatorów (binarnych i unitarnych) i stałych (*0*, *1*) taki, że każda funkcja zmiennych $x_1$, ..., $x_n$ może być przedstawiona za pomocą formuły zbudowanej z tych zmiennych przy użyciu operatorów wchodzących do tego systemu.

Przykłady systemów funkcjonlnie pełnych:

- AND, NOT
- NAND
- NOR

# Sekwencjonowanie układów

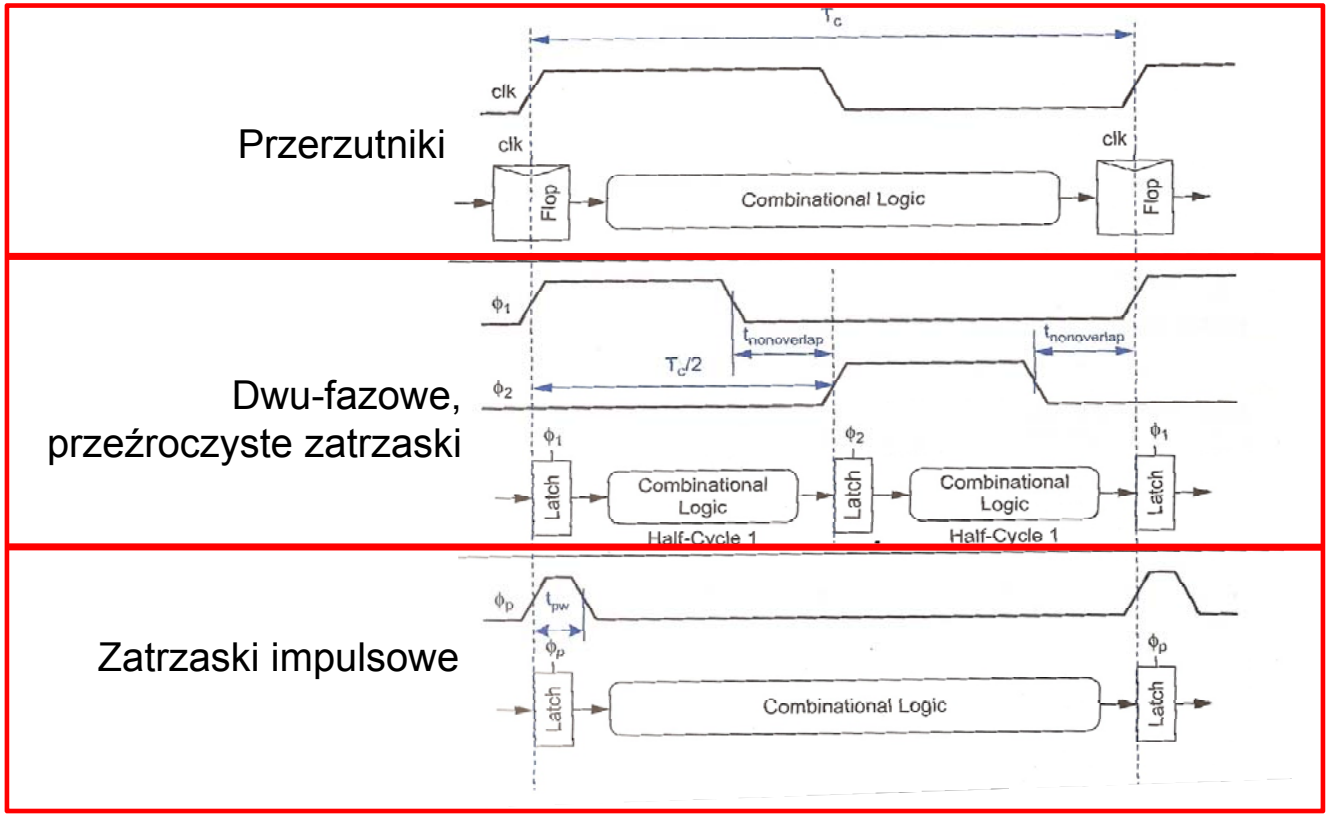
Trzy sposoby sekwencjonowania:

- Oparty ma przerzutnikach
- Dwu-fazowy, oparty na przeźroczystych zatrzaskach
- Oparty na zatrzaskach impulsowych

Wybór metody sekwencjonowania rzutuje na wszystkie fazy projektowania od zapisu, poprzez syntezę do weryfikacji. Zależy od rodzaju narzędzi (STA, ATG) i bibliotek (SCL) jakie mamy do dyspozycji.

| | |
|---|---|
| STA | Static Timing Analysis |
| ATG | Automatic Test Generation |
| SCL | Standard Cell Libraries |

# Sekwencjonowanie układów



# Układ synchroniczny

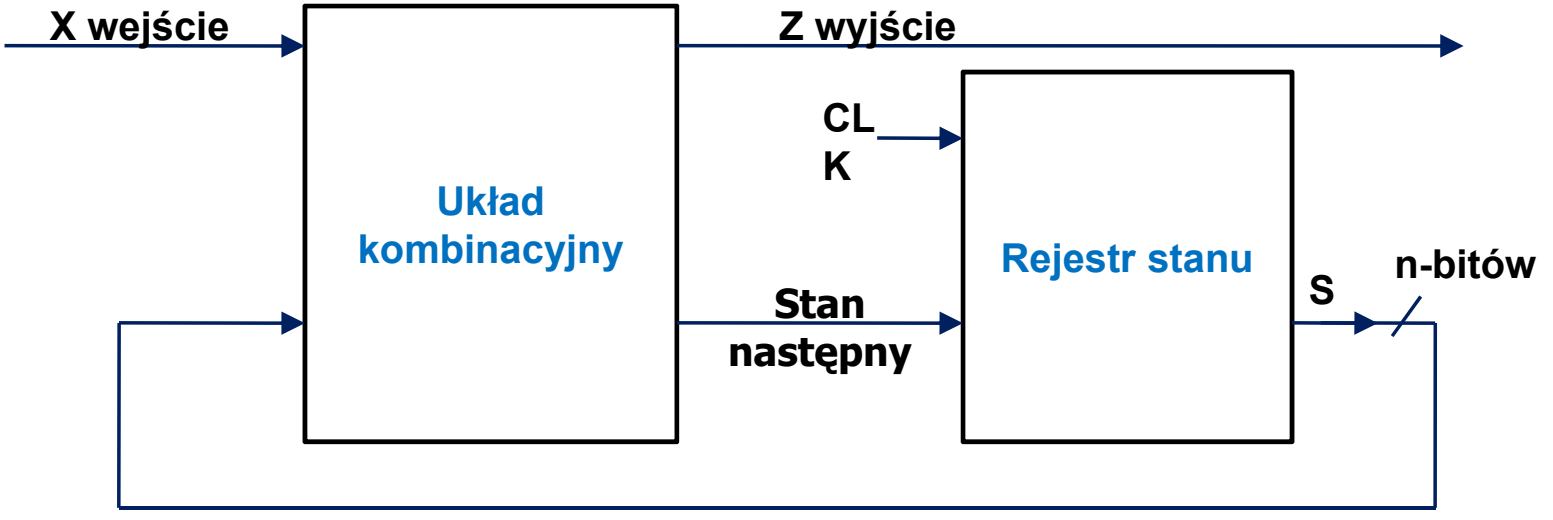
**Układ synchroniczny** charakteryzuje się:

- jednym sygnałem zegarowym (*master clock*) i
- jednym sygnałem ustawiania/kasowania (*master set/reset*),

które sterują wszystkimi elementami sekwencyjnymi w projekcie.

Za najbardziej bezpieczne podejście w dziedzinie zapewnienia właściwych relacji czasowych w układzie scalonym uważa się metodologię układu synchronicznego.

# Układ synchroniczny – zależności czasowe




$t_{cmax}$ – maksymalny czas propagacji przez układ kombinacyjny

$t_{pmax}$ - maksymalny czas propagacji od zmiany CLK do zmiany S (rejestr stanu)

$t_{pmax} = max(t_{pHL}, t_{pLH})$

$t_{pmax} + t_{cmax}$ – maksymalny czas propagacji od zmiany CLK do chwili gdy zmiany rejestru stanu pojawią się na wejściu przerzutników rejestru stanu

$$t_{c\max} + t_{p\max} \leq t_{clk} - t_{su}$$

# Zależności czasowe

**Warunek na czas przygotowania:**

$$t_{clk} \geq t_{c\max} + t_{p\max} + t_{su}$$

Przykład:
Układ sekwencyjny składa się z dwupoziomowego układu kombinacyjnego i rejestru stanów. Czas propagacji przez bramkę wynosi: 15ns, a czas propgacji przez przerzutnik 15ns. Czas przygotowania przerzutnika wynosi 5ns. Z jaką nawiększa częstotliowością może pracować poprawnie ten układ ?

$t_{clk} \geq 2 \cdot 15 + 15 + 5 = 50$ ns

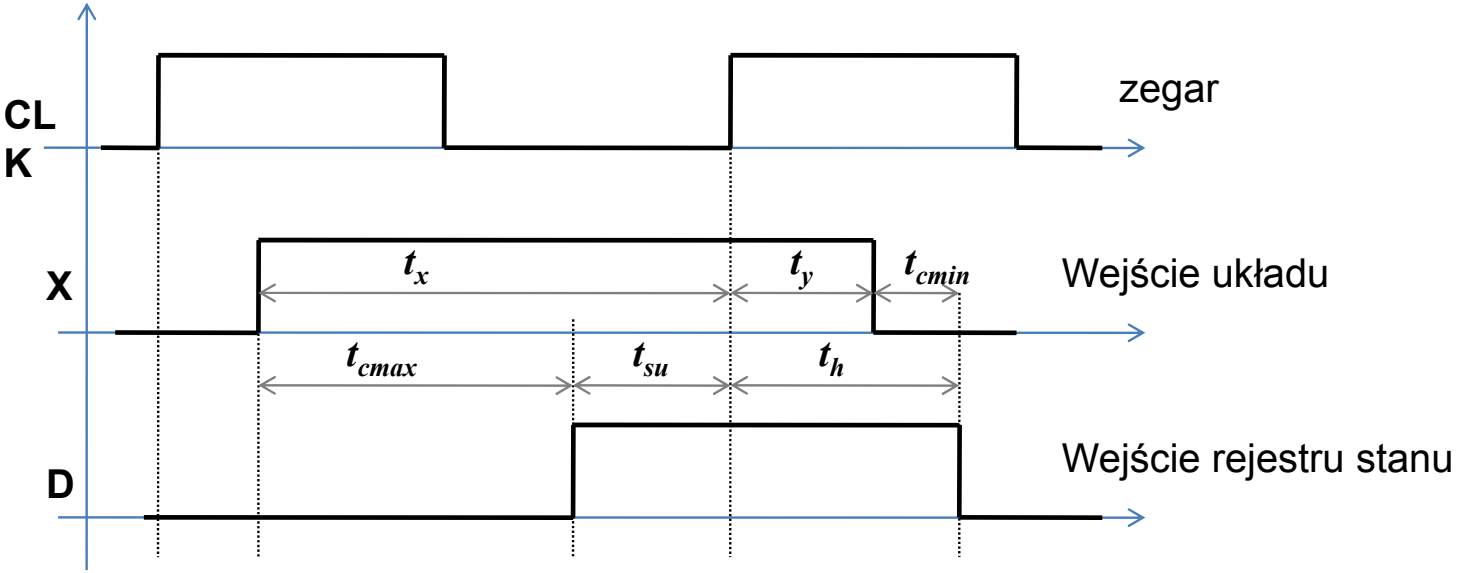
$f_{clk\ max} = (t_{clk})^{-1} = 20$ MHz

**Warunek na czas trzymania:**

$$t_{pmax} + t_{cmax} \geq t_h$$

Jest złamany gdy wejście informacyjne zmienia się za wcześnie **po** aktywnym zboczu zegara.

# Zależności czasowe




$t_{cmax}$ – maksymalny czas propagacji przez układ kombinacyjny od wejścia $X$ do wejścia $D$ przerzutników rejestru stanu

$t_{cmin}$ – minimalny czas propagacji przez układ kombinacyjny od wejścia $X$ do wejścia $D$ przerzutników rejestru stanu

Warunek na setup time: $t_x \geq t_{cmax} + t_{su}$

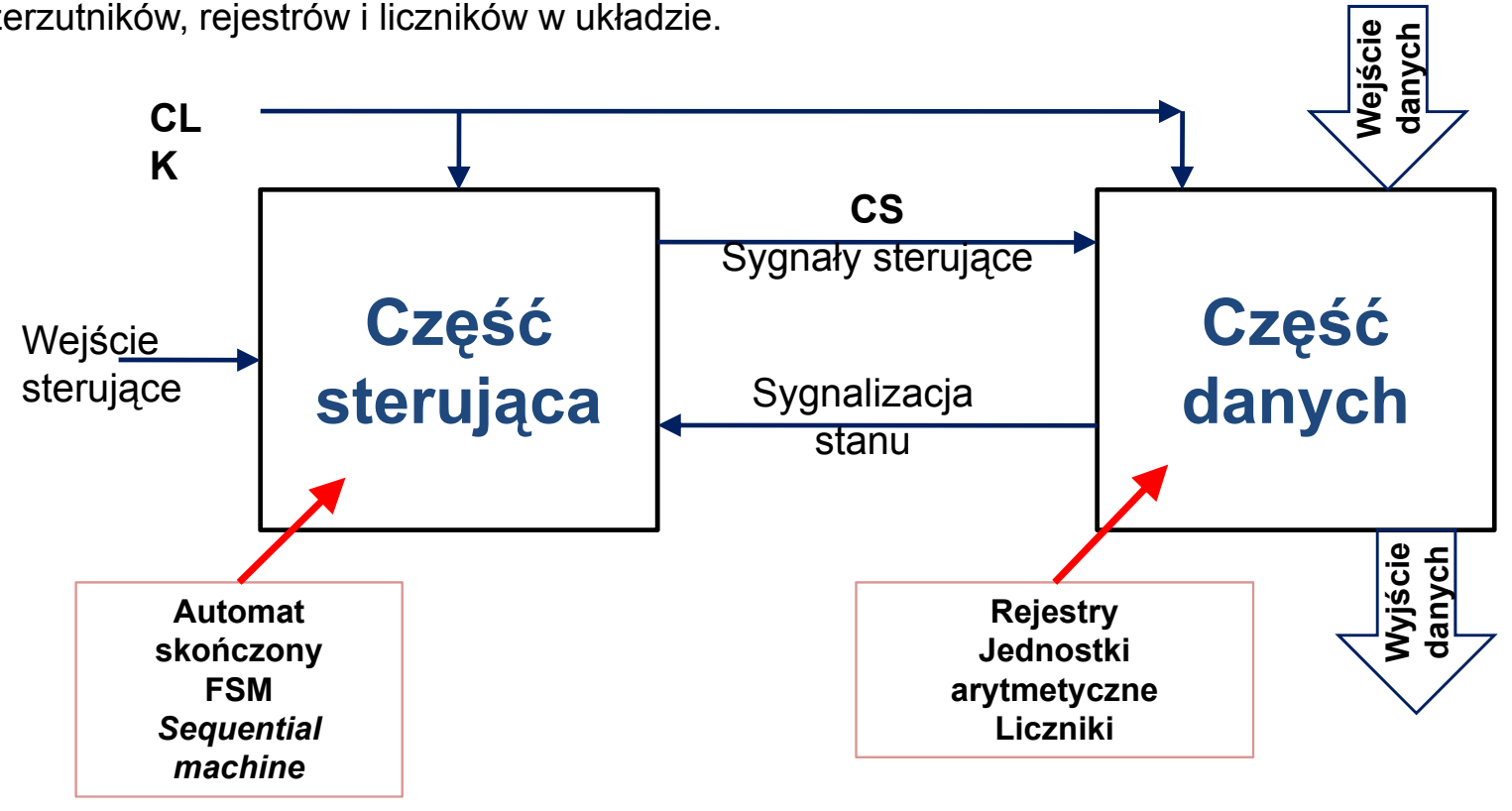
Zapewnić aby X zmieniało się odpowiednio wcześnie przed zboczem zegara

Warunek na hold time: $t_y \geq t_h - t_{cmin}$

Zapewnić aby X nie zmieniało się zbyt szybko po zboczu zegara

# Układ synchroniczny

Układ taki używa specjalnego sygnału zwanego zegarem **CLK** do synchronizacji pracy wszystkich przerzutników, rejestrów i liczników w układzie.




Dla poprawnej pracy układu synchronicznego należy zapewnić:

- Równoczesność zmian wszystkich przerzutników,
- Odpowiednią długość okresu CLK dla stabilizacji stanu wszystkich przerzutników przed następnym aktywnym zboczem zegara.

# Układ synchroniczny

Sygnał sterujący – *Control Signal* – **CS** – np. Load i Shift dla rejestru przesuwnego.
Sygnalizacja stanu – *Condition Signals* – np. Overload z jednostki artmetycznej.

**Zasady projektowania:**

- Wszystkie wejścia zegarowe przerzutników (rejestrów, liczników i in.) są sterowane bezpośrednio z zegara systemowego CLK lub z zegara bramkowanego sygnałem sterującym CS.
- Wszystkie zmiany stanu odbywaja się natychmiast po zboczu aktywnym CLK.
- Wszystkie stany przejściowe i zakłócenia wydarzają się pomiędzy zboczami zegara i nie mają wpływu na działanie układu.

# Układ synchroniczny – asynchroniczny

**Wady rozwiązania synchronicznego:**

- Rozprowadzanie sygnału zegarowego tak aby docierał wszędzie w tej samej chwili.
- Maksymalna szybkość jest ograniczona przez najgorszy przypadek opóźnienia najdłuższej ścieżki sygnału.
- Konieczność synchronizacji wejść z zegarem.

**Układy asynchroniczne:**

- Brak zegara.
- Problemy z synchronizacją.
- Specjalne techniki eliminacji hazardów.