PODSTAWY UKŁADÓW LOGICZNYCH

Prowadzi:
mgr inż. Piotr Godlewski

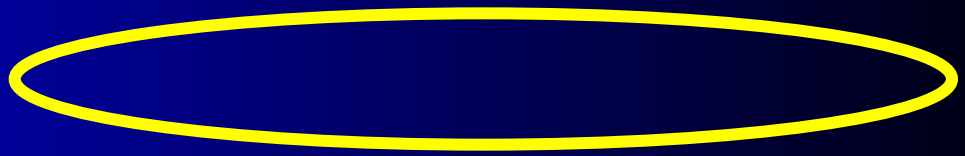
Organizacja

P. Godlewski

Wykład

P. Godlewski

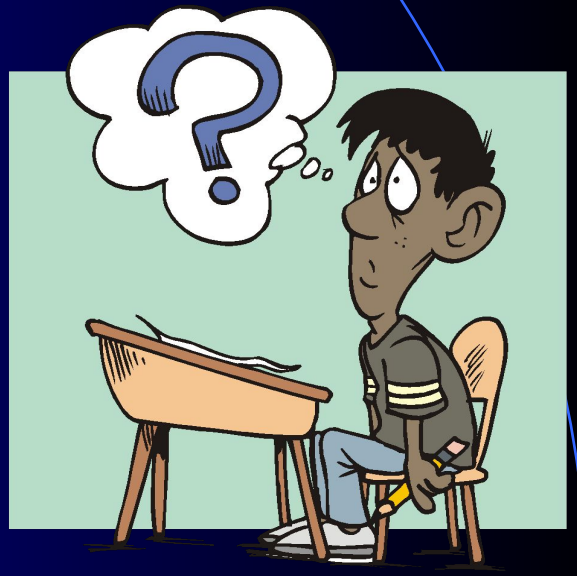
Ćwiczenia



Egzamin...

Ćwiczenia 50 pkt.

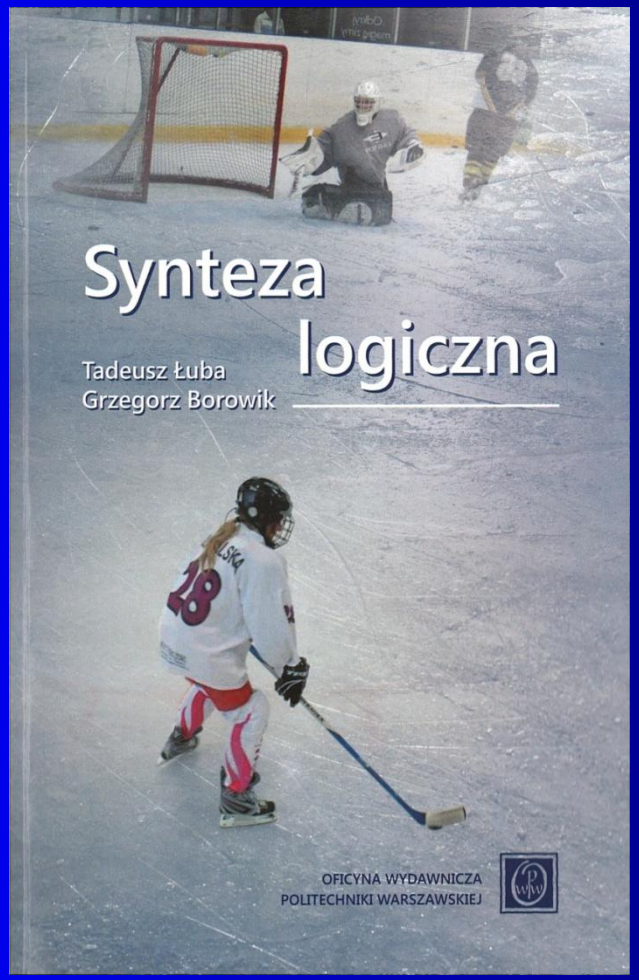
Egzamin uzupełnienie do 100 pkt

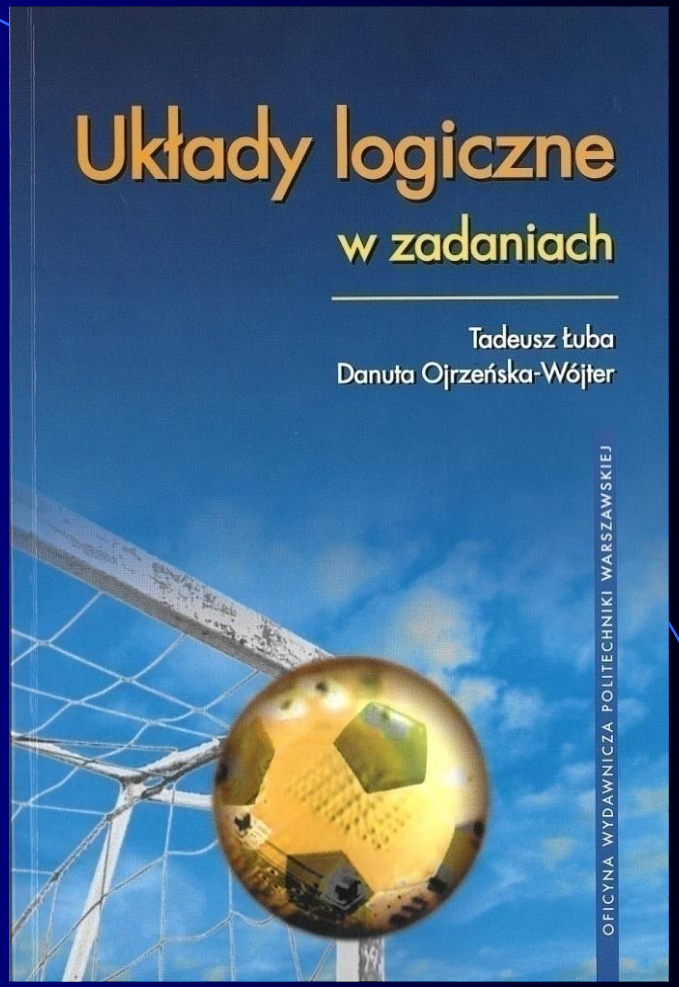


Literatura

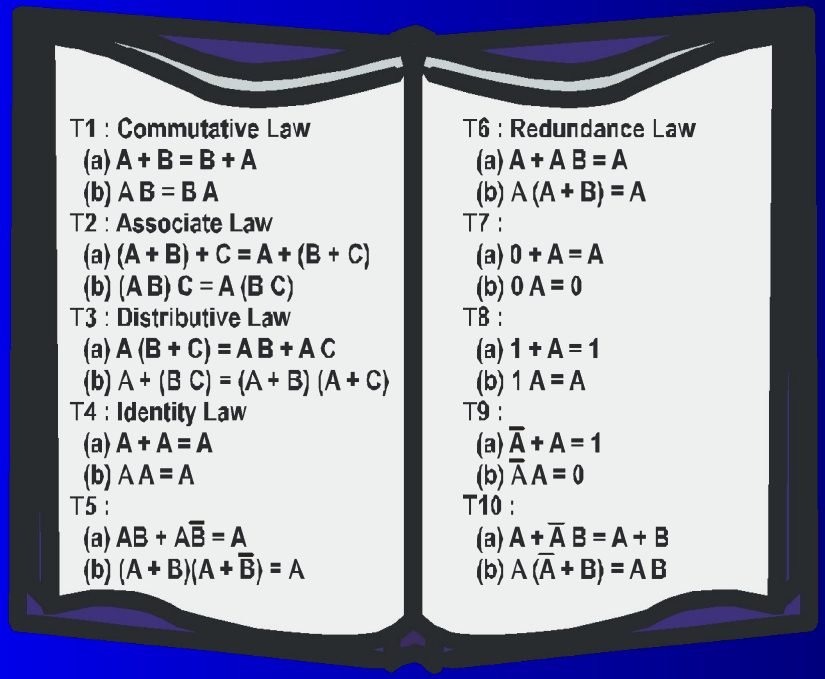
1. Ashar P., Devadas S., Newton A. ...is. Kluwer Academic
2. *Minimization Algorith...* Publishers, Bost...
3. Brown F. M.: *Bo...* Kluwer Aca...
4. Brzozowski ... New Yor...
5. Brzozows... *by Cu...* Vol. ...
6. De Mi... New... *uk...*
7. Deva... Ne...
8. Gajs... Ne...
9. Hass... Klu...
10. Iman ... Aca...
11. Kamion... *i przy...* Gliwice...
12. Kania D.: *Sy... programov...* Nr 1619. Gliv...
13. Katz R. H.: *Conte...* Publishing Comp...
14. Kohavi Z.: *Switching a...* New York, 1978.
15. Kuźmicz W.: *Układy ASIC w pols...* kacyjny i Wiadomości Telekomunikacyj...
16. Lala P.K.: *Practical digital logic design and testing* New Jersey 1996.
17. Łuba T.(red.), Rawski M., Tomaszewicz P., Zbierzchowski B.: *Synteza układów cyfrowych.* WKŁ Warszawa 2003.
18. Łuba T., Zbierzchowski B., Zbysiński P.: *Układy reprogramowalne dla potrzeb telekomunikacji cyfrowej*. Przegląd Telekomunikacyjny i Wiadomości Telekomunikacyjne, nr 5, 2002.
19. Łuba T.: *Synteza układów logicznych*. Wyższa Szkoła Informatyki Stosowanej i Zarządzania, Wyd. 2, Warszawa 2001.
20. Łuba T.: *Rola i znaczenie syntezy logicznej w technice cyfrowej układów programowalnych.* Elektronika, str. 15 ¸ 19, nr 7-8, 2002.
21. ...uba T., Jasiński K., Zbierzchowski B.: *Programowalne układy przetwarzania sygnałów i informacji - technika cyfrowa ...w multimediach i kryptografii,* Przegląd Telekomunikacyjny i Wiado-...ości Telekomunikacyjne, str. 408-418, nr 8-9, 2003.
22. ...W., Albicki A.: *Algebraiczna teoria automatów.* WNT, ...awa 1980.
23. ...B.: *Algebraiczna i strukturalna teoria automatów.* PWN, ...a – Łódź 1985.
24. ...stemy ekspertowe.* WNT, Warszawa 1996.
25. ...ka L.: *Analiza danych metodą zbiorów ...ych. Zastosowania w ekonomii, medycynie i sterowaniu.* ...a Oficyna Wydawnicza PLJ, Warszawa 1999.
26. ... *Logic design principles, with emphasis on testable ... circuits.* Prentice-hall International, Inc., New Jersey
27. ... Sets. *Theoretical Aspects of Reasoning about ...r* Academic Publishers, 1999.
28. ...*amentals of Logic Design.* West Publ. CO., 1985.
29. ...gic A.: *Digital systems design and prototyping ...mmable logic.* Kluwer Academic Publishers, 1997.
30. ... *...y for Logic Synthesis,* Kluwer Academic
31. ...*timization.* Kluwer Academic
32. ...i R.(ed.): *Int... ...pport - Handbook of ...pplications and Adv... ...Sets Theory,* Kluwer Academic Publishers, Dor...
33. ...choll C.: *Functional Decomposition ... FPGA Synthesis.* Kluwer Academic Publis...
34. Tyszer J.: *Układy cyfrowe. Materiały pomocni... ...o wy...* Wyd. Politechniki Poznańskiej. Poznań 2000.
35. Zieliński C.: *Podstawy projektowania układów cyfrowych.* PWN, Warszawa 2003.
36. Zbysiński P., Pasierbiński J.: *Układy programowalne – pierwsze kroki.* Wyd. II, Wydawnictwo BTC. Warszawa 2004.

Literatura


Synteza
logiczna
Tadeusz Łuba
Grzegorz Borowik
OFICYNA WYDAWNICZA
POLITECHNIKI WARSZAWSKIEJ

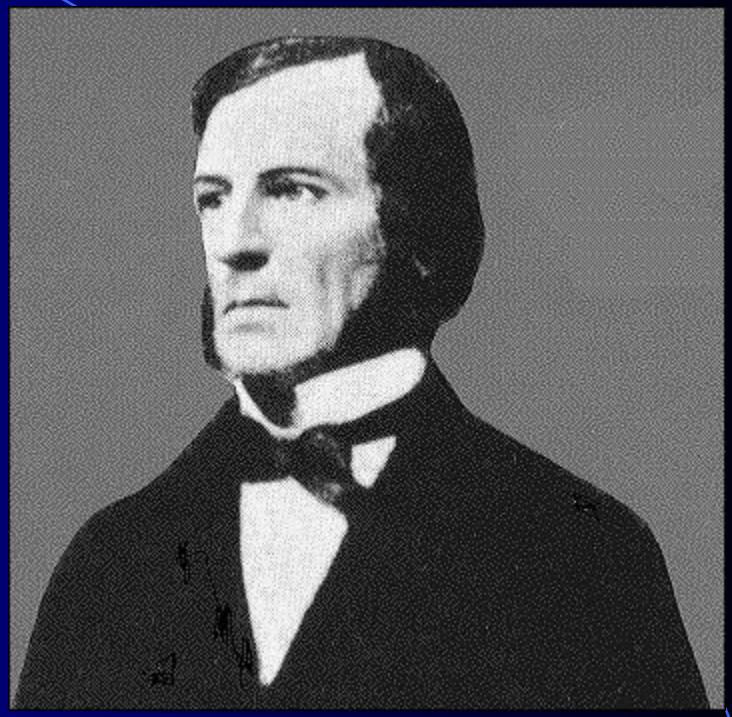

Układy logiczne
w zadaniach
Tadeusz Łuba
Danuta Ojrzeńska-Wójter
OFICYNA WYDAWNICZA POLITECHNIKI WARSZAWSKIEJ

Z układami logicznymi mamy do czynienia od dawna...



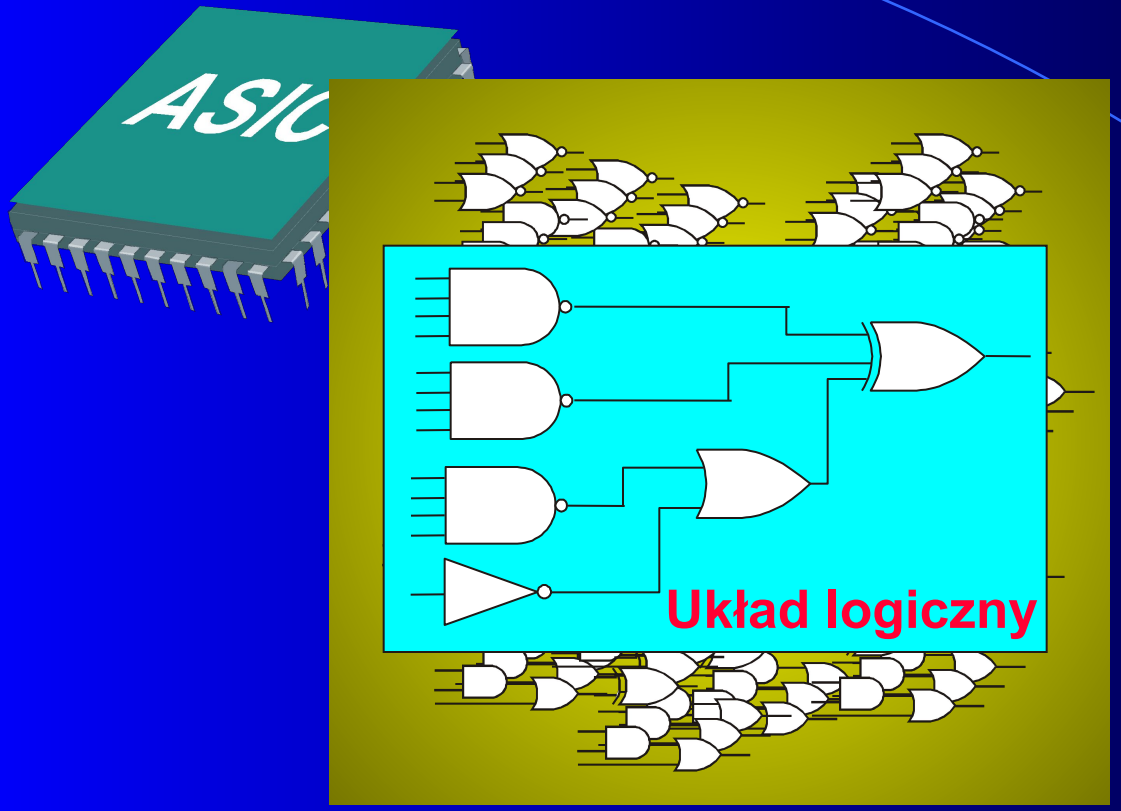
T1 : Commutative Law
(a) $A + B = B + A$
(b) $A\,B = B\,A$
T2 : Associate Law
(a) $(A + B) + C = A + (B + C)$
(b) $(A\,B)\,C = A\,(B\,C)$
T3 : Distributive Law
(a) $A\,(B + C) = A\,B + A\,C$
(b) $A + (B\,C) = (A + B)\,(A + C)$
T4 : Identity Law
(a) $A + A = A$
(b) $A\,A = A$
T5 :
(a) $AB + A\overline{B} = A$
(b) $(A + B)(A + \overline{B}) = A$

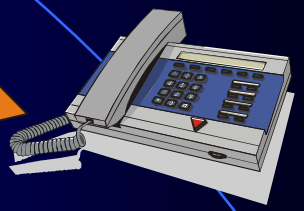
T6 : Redundance Law
(a) $A + A\,B = A$
(b) $A\,(A + B) = A$
T7 :
(a) $0 + A = A$
(b) $0\,A = 0$
T8 :
(a) $1 + A = 1$
(b) $1\,A = A$
T9 :
(a) $\overline{A} + A = 1$
(b) $\overline{A}\,A = 0$
T10 :
(a) $A + \overline{A}\,B = A + B$
(b) $A\,(\overline{A} + B) = A\,B$

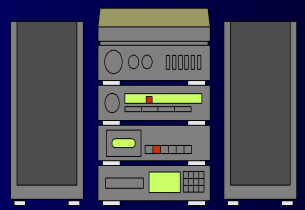


Rok 1847

Już w latach 80. 20 wieku

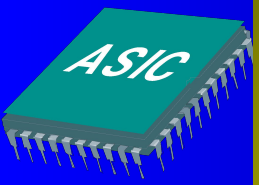


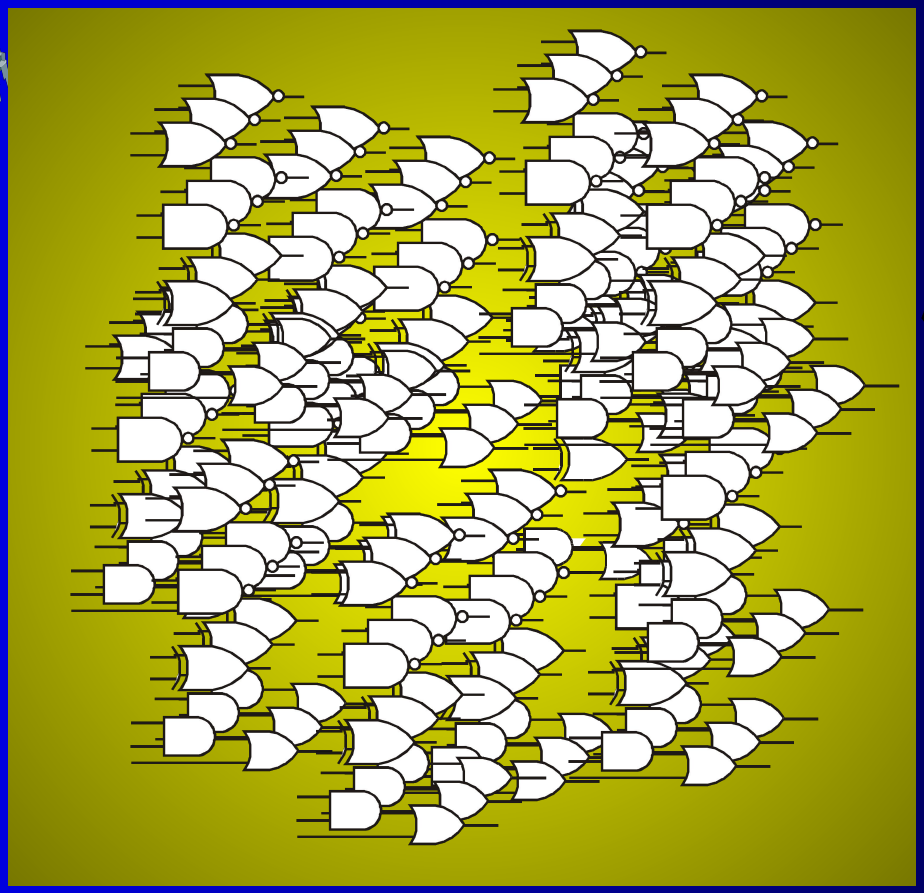




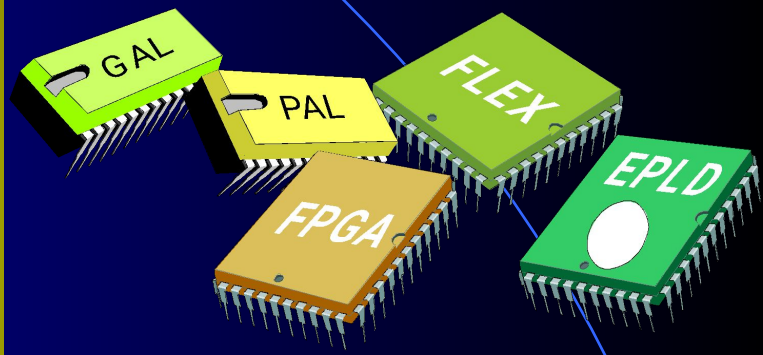
…dzisiaj

miliony bramek logicznych





Nowa jakość…

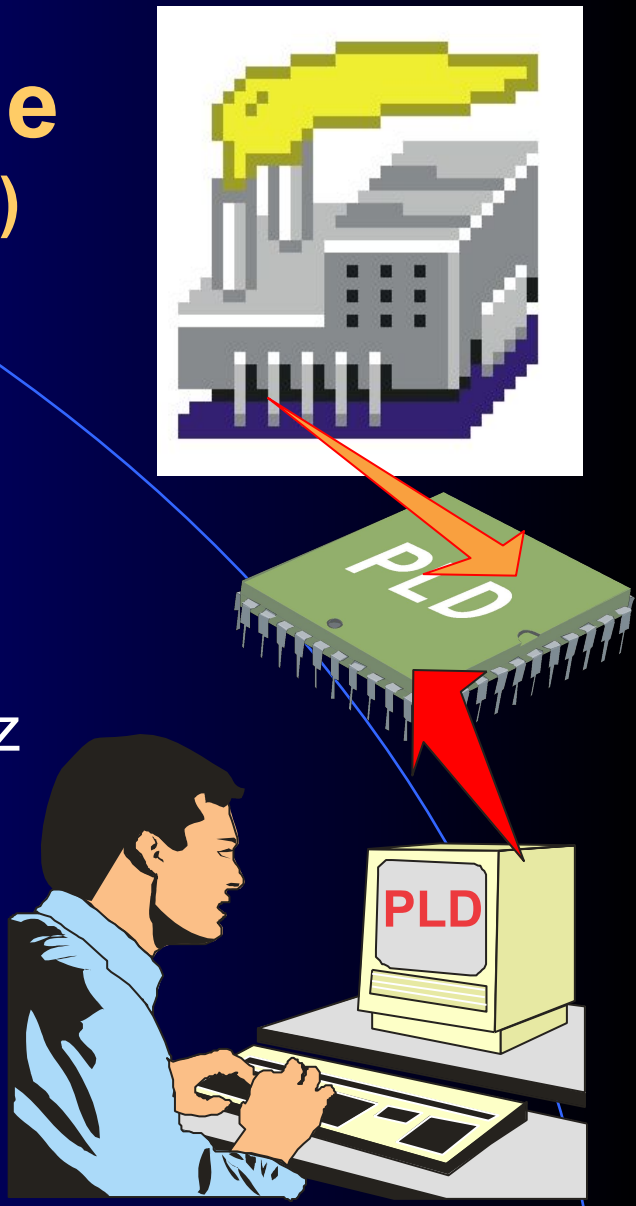


…programowanie połączeń

Programowalne moduły logiczne (Programmable Logic Devices)

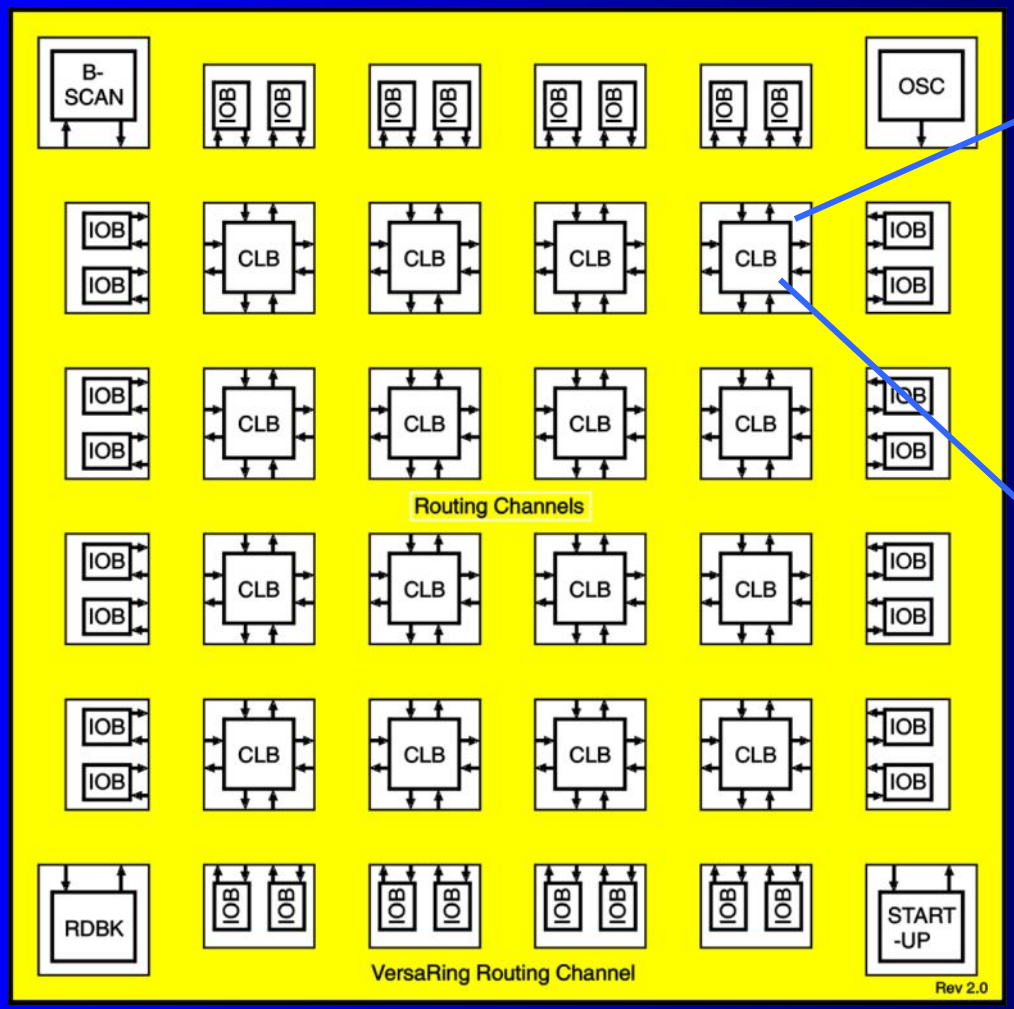
Układy programowalne
(Programmable Logic Devices)

to układy scalone, których właściwości funkcjonalne są definiowane nie przez producenta, lecz przez końcowego użytkownika. Najważniejsza cechą tych układów jest możliwość nadawania im (przez programowanie) określonych przez użytkownika cech funkcjonalnych, w jego laboratorium czy na biurku, a nie w fabryce.



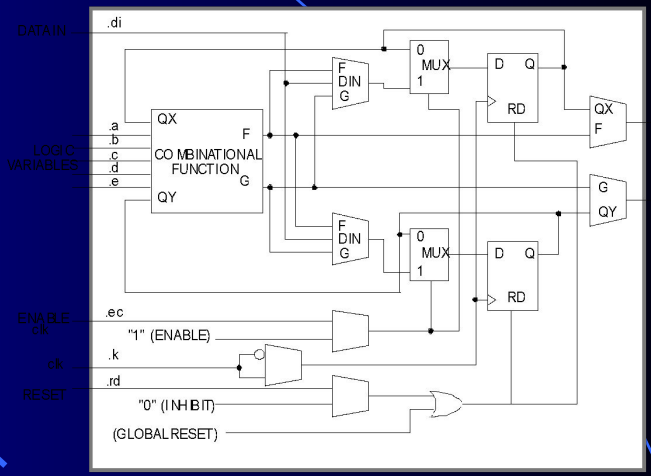
Układy FPGA

(Field Programmable Gate Array)



Configurable Logic Block (CLB)

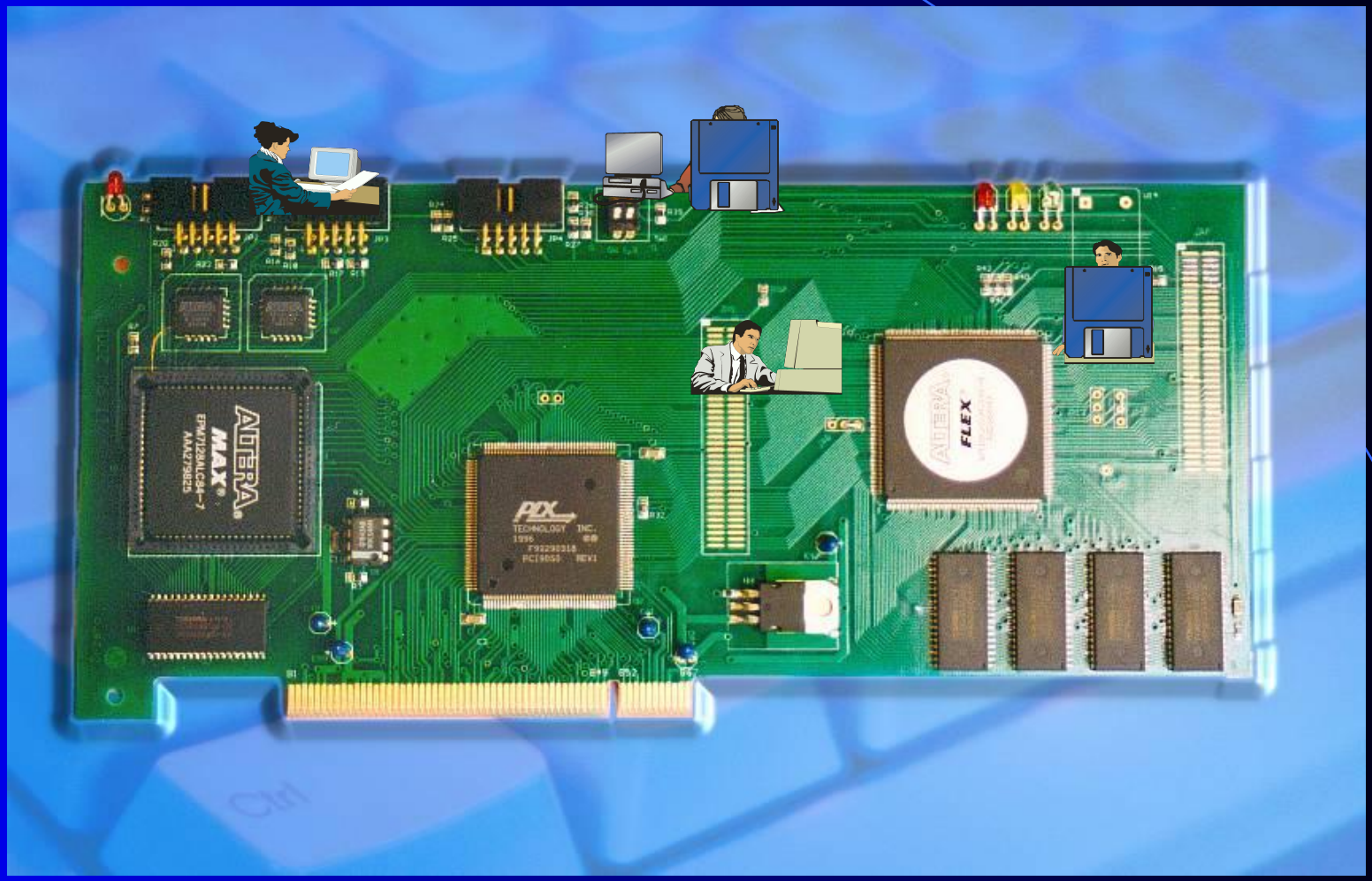
Logic Element (LE)



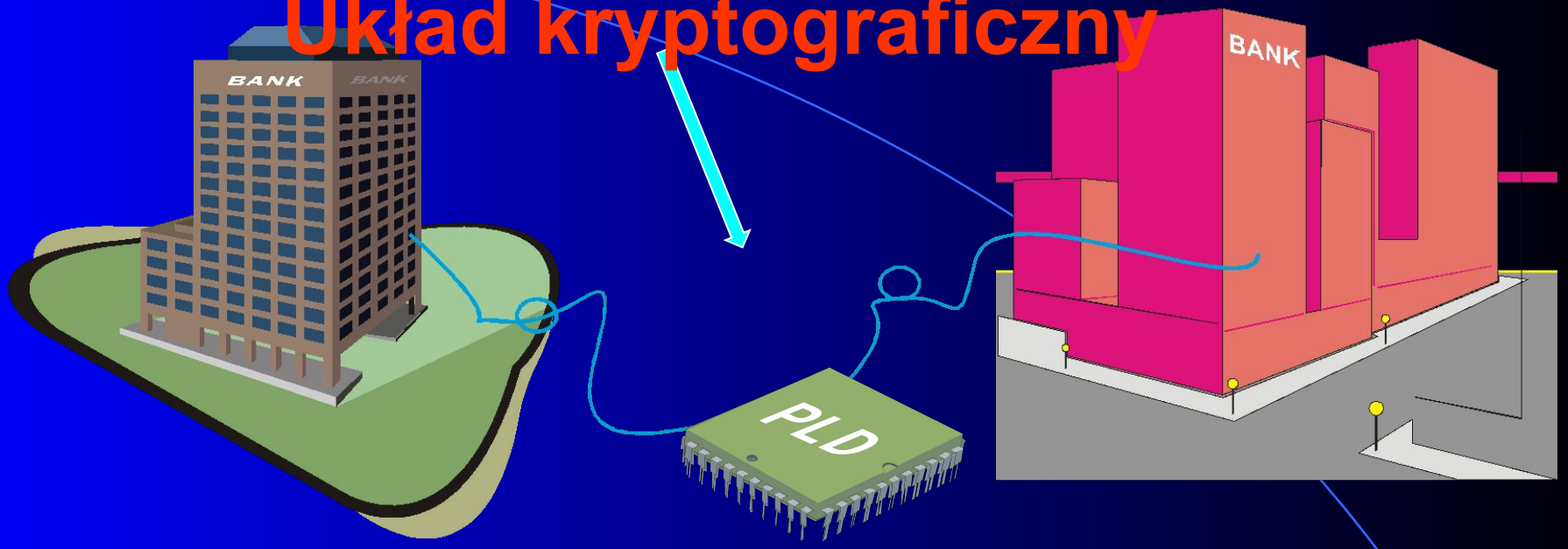
Reprogramowania i rekonfiguracji

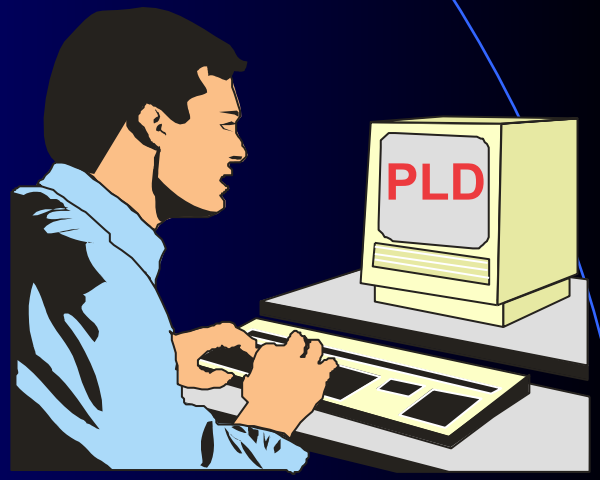
Układy programowalne wyrównują szanse w dostępie do najnowszych technologii…

…niezależnie od miejsca zatrudnienia!



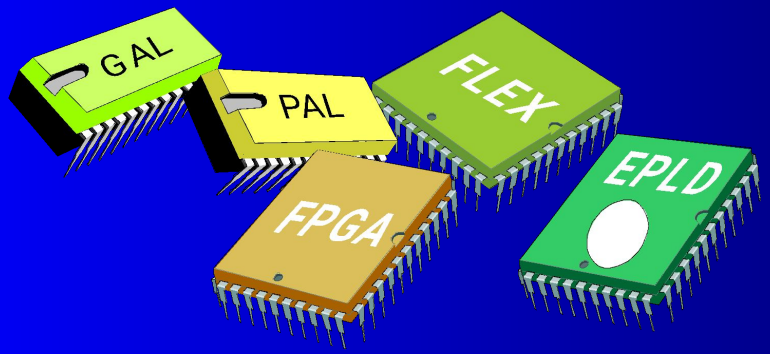
Układ kryptograficzny

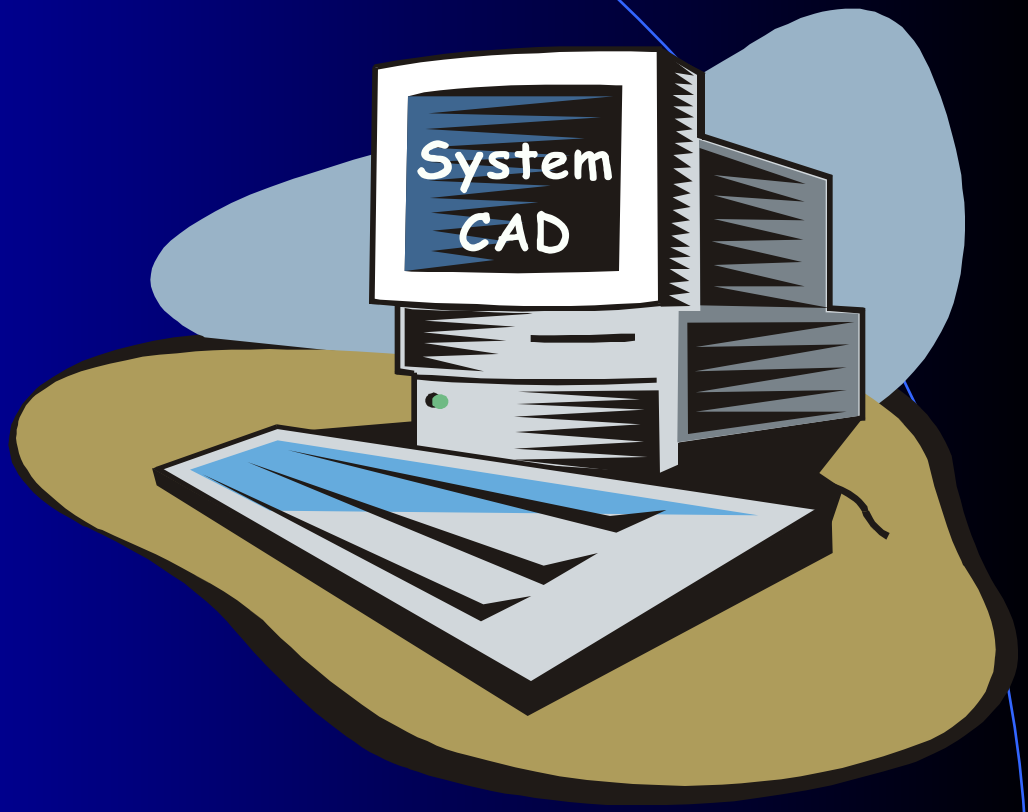




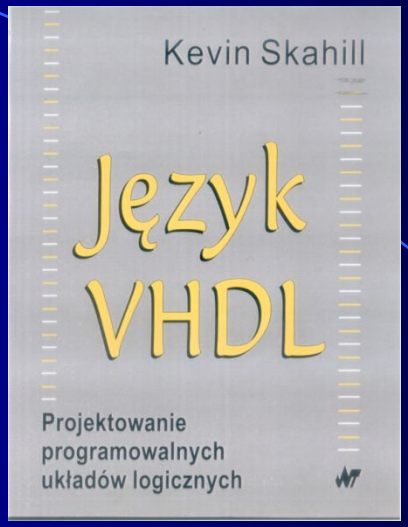
Komputerowe systemy projektowania

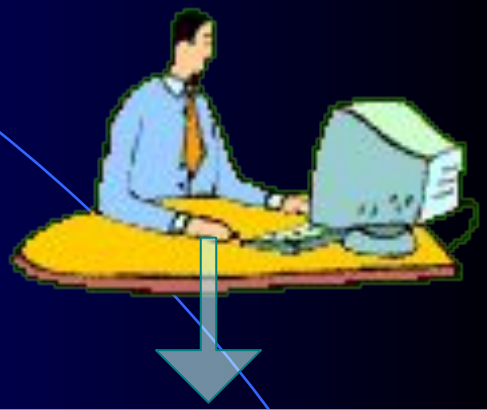
Ze względu na skomplikowaną budowę struktur programowalnych ich realizacja nie może się odbywać bez…

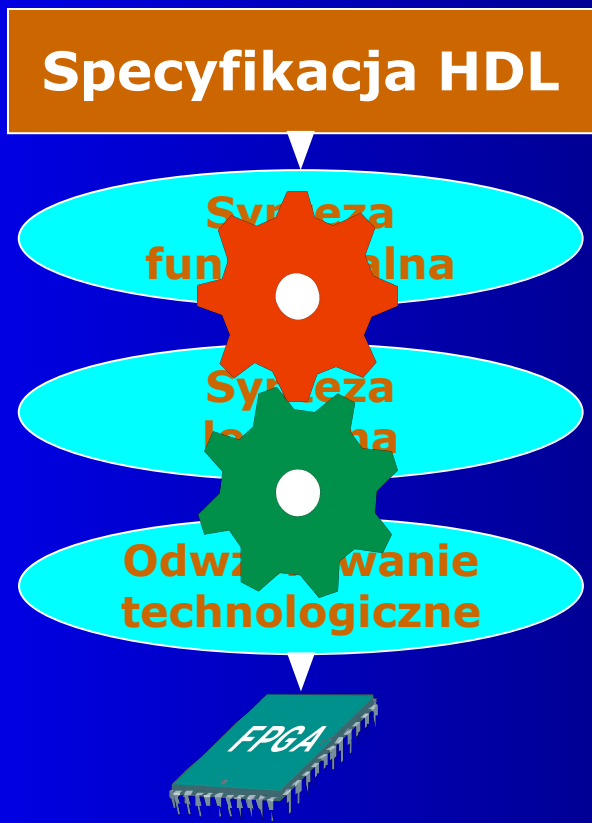


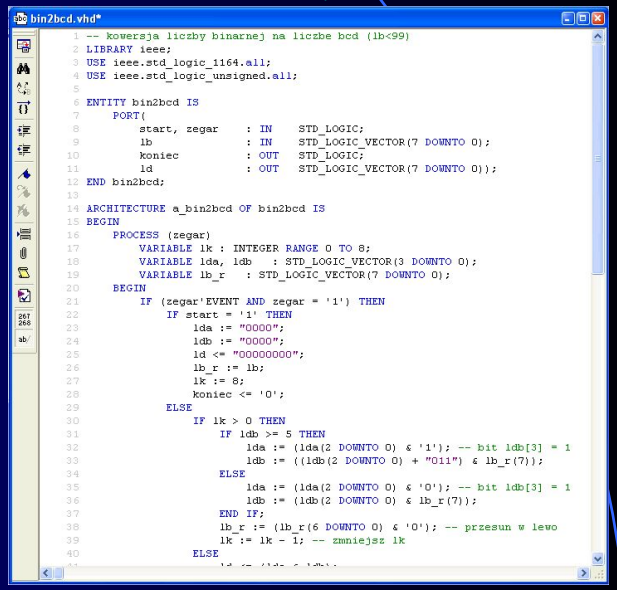


Komputerowe projektowanie...



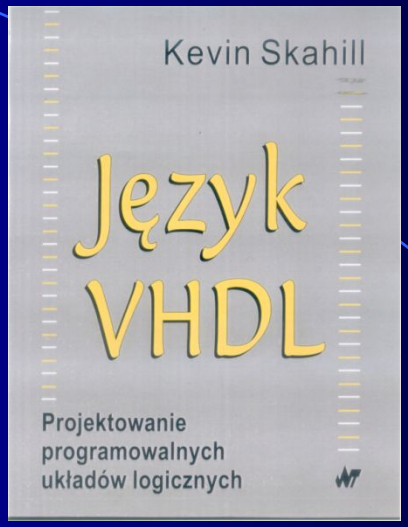


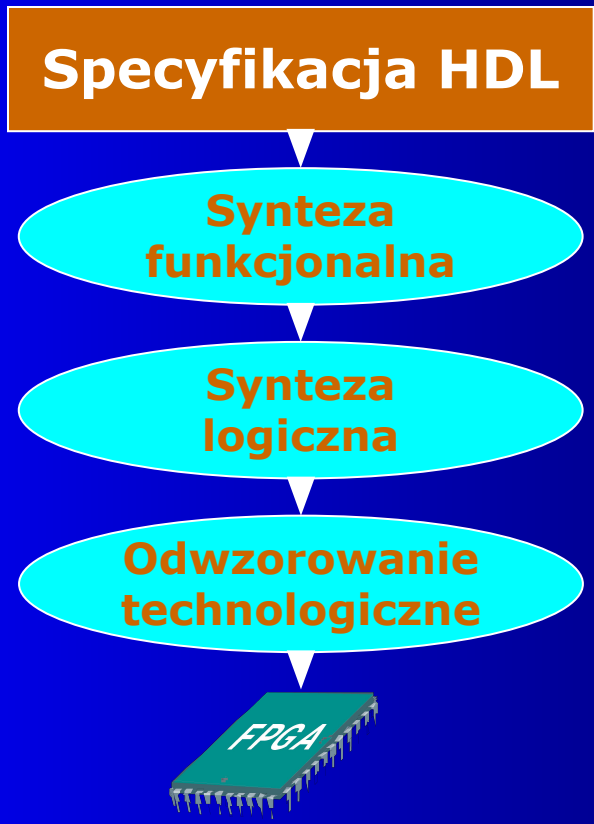




14

Komputerowe projektowanie…



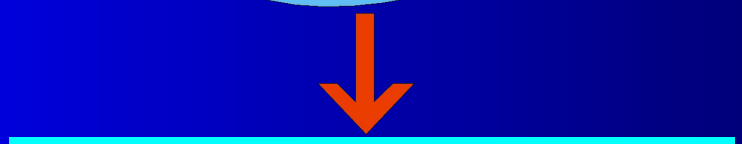


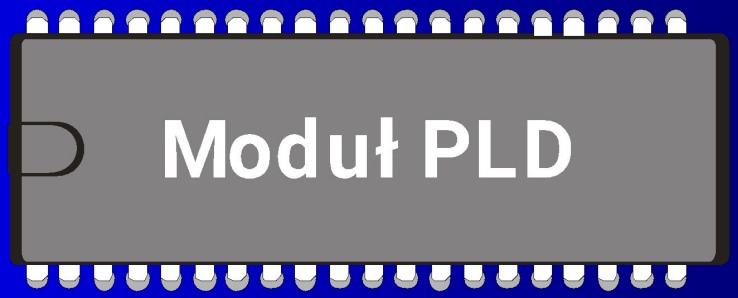
…aż do zaprogramowania układu!

15

Projektowanie jest proste?

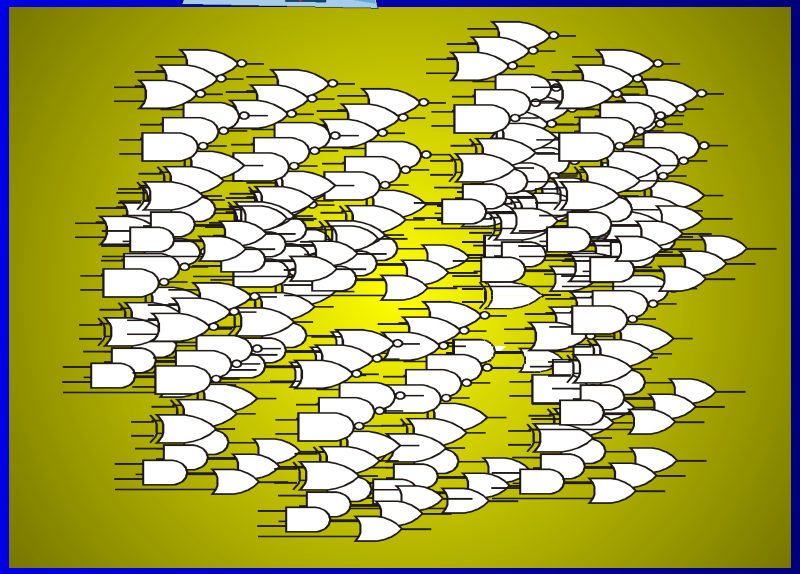
Język opisu sprzętu

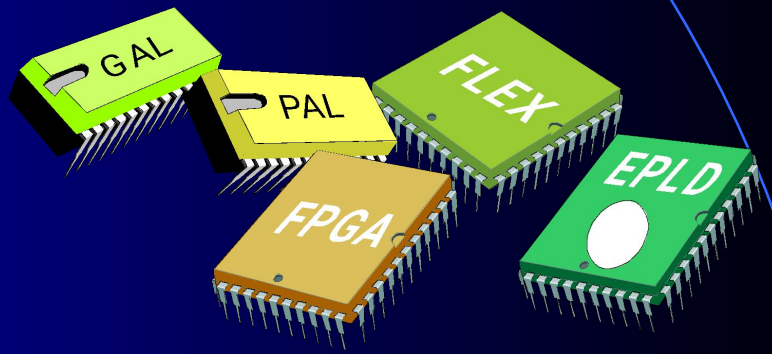




Niestety…

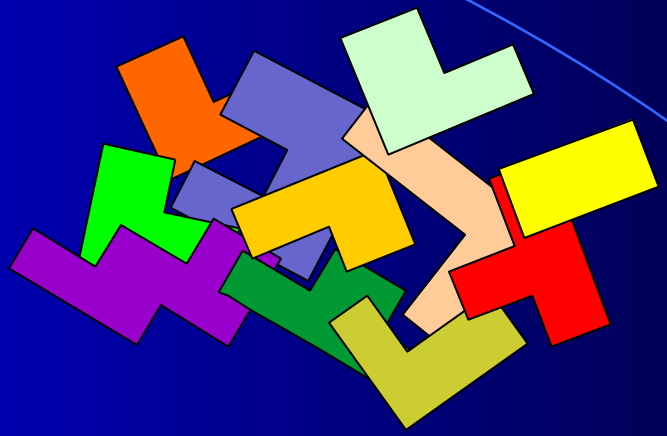
Jak je skonfigurować ???

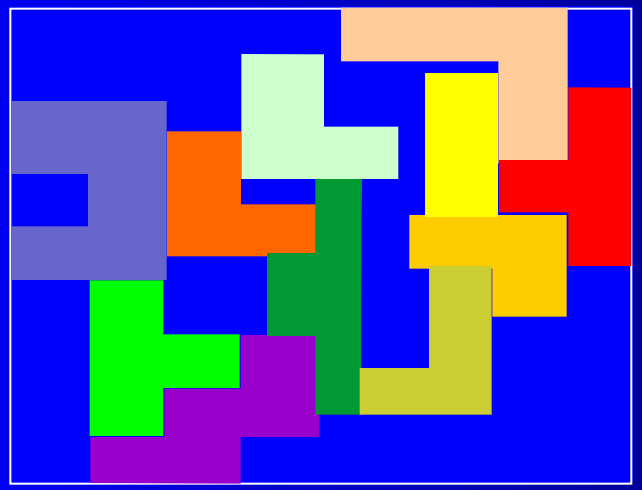


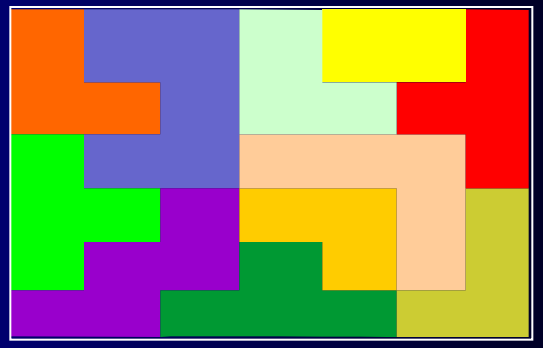


MILIONY BRAMEK !!!

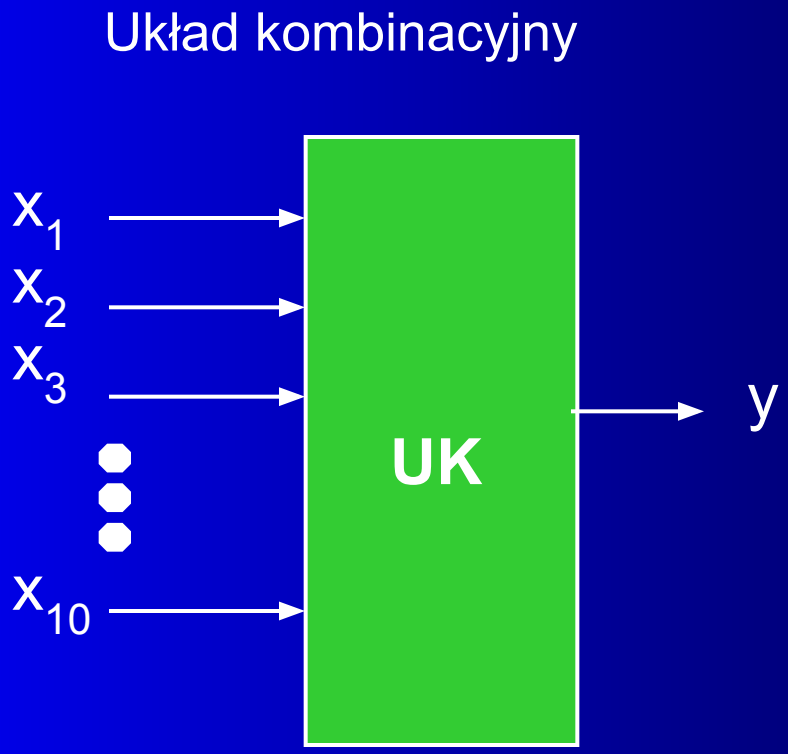
Metoda puzzli







Przykład – prosty układ kombinacyjny



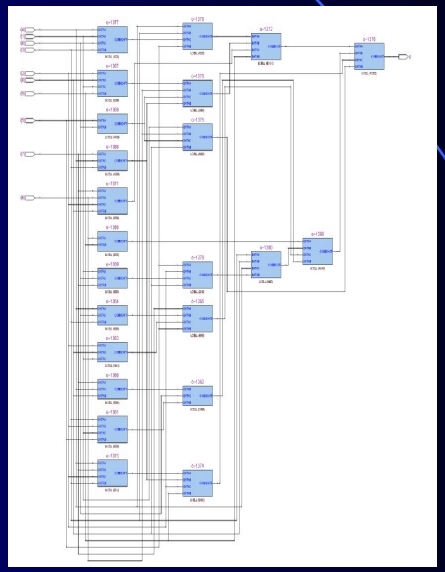

```
.type fr
.i 10
.o 1
.p 25
0010111010 0
1010010100 0
0100011110 0
1011101011 0
1100010011 0
0100010110 0
1110100110 0
0100110000 0
0101000010 0
0111111011 1
0000010100 1
1101110011 1
0100100000 1
0100011111 1
0010000110 1
1111010001 1
1111101001 1
1111111111 1
0010000000 1
1101100111 1
0010001111 1
1111100010 1
1010111101 1
0110000110 1
0100111000 1
.e
```

Realizacja funkcji F w systemie Quartus

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tl27 IS
  PORT (
       in: IN STD_LOGIC_VECTOR(9
DOWNTO 0);
       out: OUT STD_LOGIC_VECTOR(0
DOWNTO 0)
      );
END tl27;

ARCHITECTURE tl27_arch OF tl27 IS
BEGIN
  pandor: PROCESS (in)
  BEGIN
    CASE in IS
      WHEN "0010111010" => out <= "0";
      WHEN "1010010100" => out <= "0";
      WHEN "0100011110" => out <= "0";
      WHEN "1011101011" => out <= "0";
      WHEN "1100010011" => out <= "0";
      WHEN "0100010110" => out <= "0";
      WHEN "1110100110" => out <= "0";
      WHEN "0100110000" => out <= "0";
      WHEN "0101000010" => out <= "0";
      WHEN "0111111011" => out <= "1";
      WHEN "0000010100" => out <= "1";
      WHEN "1101110011" => out <= "1";
      WHEN "0100100000" => out <= "1";
      WHEN "0100011111" => out <= "1";
      WHEN "0010000110" => out <= "1";
      WHEN "1111010001" => out <= "1";
      WHEN "1111101001" => out <= "1";
      WHEN "1111111111" => out <= "1";
      WHEN "0010000000" => out <= "1";
      WHEN "1101100111" => out <= "1";
      WHEN "0010001111" => out <= "1";
      WHEN "1111100010" => out <= "1";
      WHEN "1010111101" => out <= "1";
      WHEN "0110000110" => out <= "1";
      WHEN "0100111000" => out <= "1";
      WHEN OTHERS => out <= "0";
    END CASE;
  END PROCESS pandor;
END tl27_arch;
```

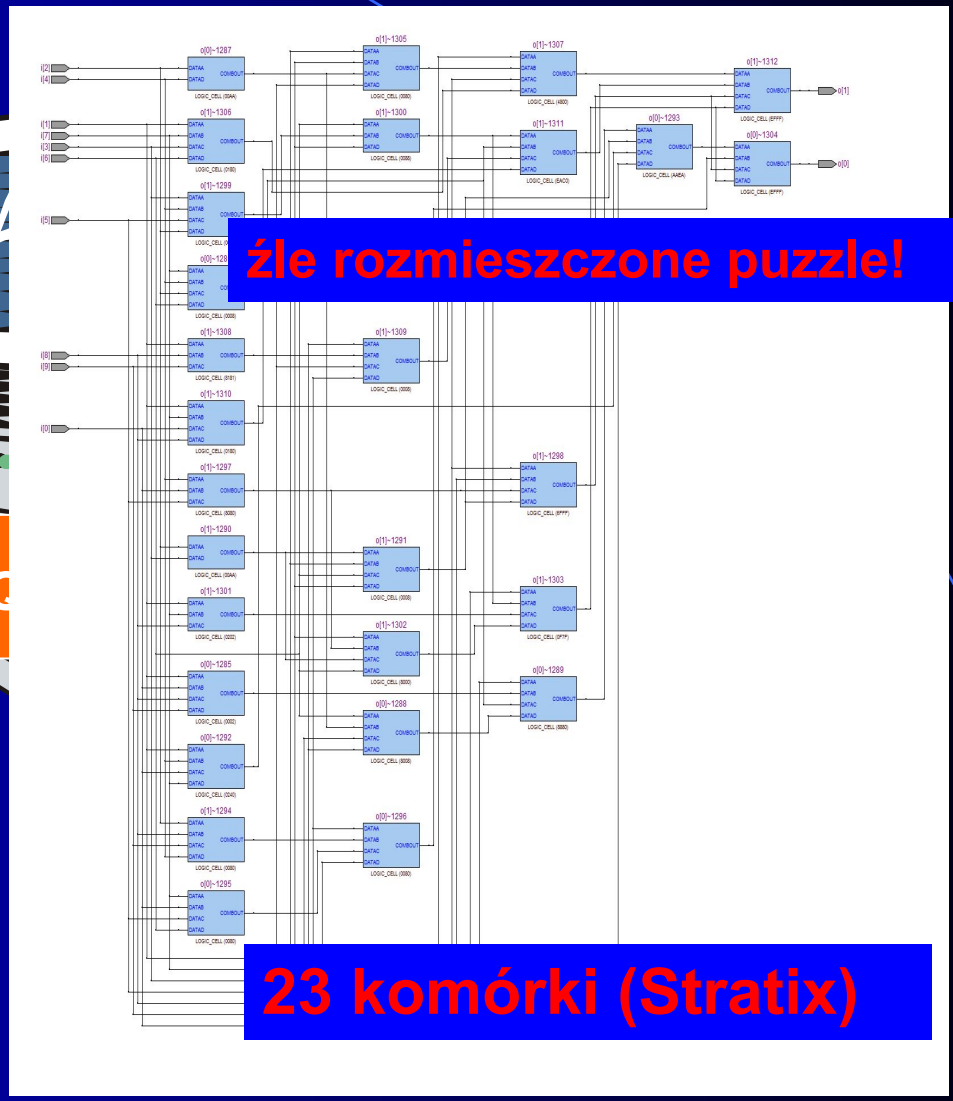


Realizacja funkcji F w systemie Quartus

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tl27 IS
  PORT (
       in: IN STD_LOGIC_VECTOR(9
DOWNTO 0);
       out: OUT STD_LOGIC_VECTOR(0
DOWNTO 0)
      );
END tl27;

ARCHITECTURE tl27_arch OF tl27 IS
BEGIN
  pandor: PROCESS (in)
  BEGIN
    CASE in IS
      WHEN "0010111010" => out <= "0";
      WHEN "1010010100" => out <= "0";
      WHEN "0100011110" => out <= "0";
      WHEN "1011101011" => out <= "0";
      WHEN "1100010011" => out <= "0";
      WHEN "0100010110" => out <= "0";
      WHEN "1110100110" => out <= "0";
      WHEN "0100110000" => out <= "0";
      WHEN "0101000010" => out <= "0";
      WHEN "0111111011" => out <= "1";
      WHEN "0000010100" => out <= "1";
      WHEN "1101110011" => out <= "1";
      WHEN "0100100000" => out <= "1";
      WHEN "0100011111" => out <= "1";
      WHEN "0010000110" => out <= "1";
      WHEN "1111010001" => out <= "1";
      WHEN "1111101001" => out <= "1";
      WHEN "1111111111" => out <= "1";
      WHEN "0010000000" => out <= "1";
      WHEN "1101100111" => out <= "1";
      WHEN "0010001111" => out <= "1";
      WHEN "1111100010" => out <= "1";
      WHEN "1010111101" => out <= "1";
      WHEN "0110000110" => out <= "1";
      WHEN "0100111000" => out <= "1";
      WHEN OTHERS => out <= "0";
    END CASE;
  END PROCESS pandor;
END tl27_arch;
```



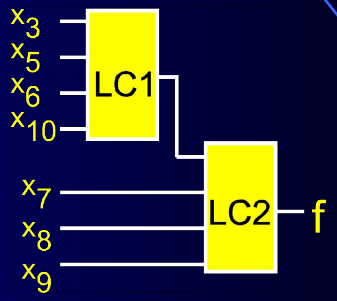
Ciekawe jak zachowa się Quartus z nową procedurą syntezy logicznej?

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tl27 IS
  PORT (
       in: IN STD_LOGIC_VECTOR(9
DOWNTO 0);
       out: OUT STD_LOGIC_VECTOR(0
DOWNTO 0)
      );
END tl27;

ARCHITECTURE tl27_arch OF tl27 IS
BEGIN
  pandor: PROCESS (in)
  BEGIN
    CASE in IS
      WHEN "0010111010" => out <= "0";
      WHEN "1010010100" => out <= "0";
      WHEN "0100011110" => out <= "0";
      WHEN "1011101011" => out <= "0";
      WHEN "1100010011" => out <= "0";
      WHEN "0100010110" => out <= "0";
      WHEN "1110100110" => out <= "0";
      WHEN "0100110000" => o      = "0";
      WHEN "0101000011" => o      = "0";
      WHEN "0111      1" => o      = "1";
      WHEN "1         1" => o      = "1";
      WH                 => o      = "1";
                         => o      = "1";
                         => out <= "1";
      W  1101001" => out <= "1";
      WH 1111111" => out <= "1";
      WHEN "0010000000" => out <= "1";
      WHEN "1101100111" => out <= "1";
      WHEN "0010001111" => out <= "1";
      WHEN "1111100010" => out <= "1";
      WHEN "1010111101" => out <= "1";
      WHEN "0110000110" => out <= "1";
      WHEN "0100111000" => out <= "1";
      WHEN OTHERS => out <= "0";
    END CASE;
  END PROCESS pandor;
END tl27_arch;
```

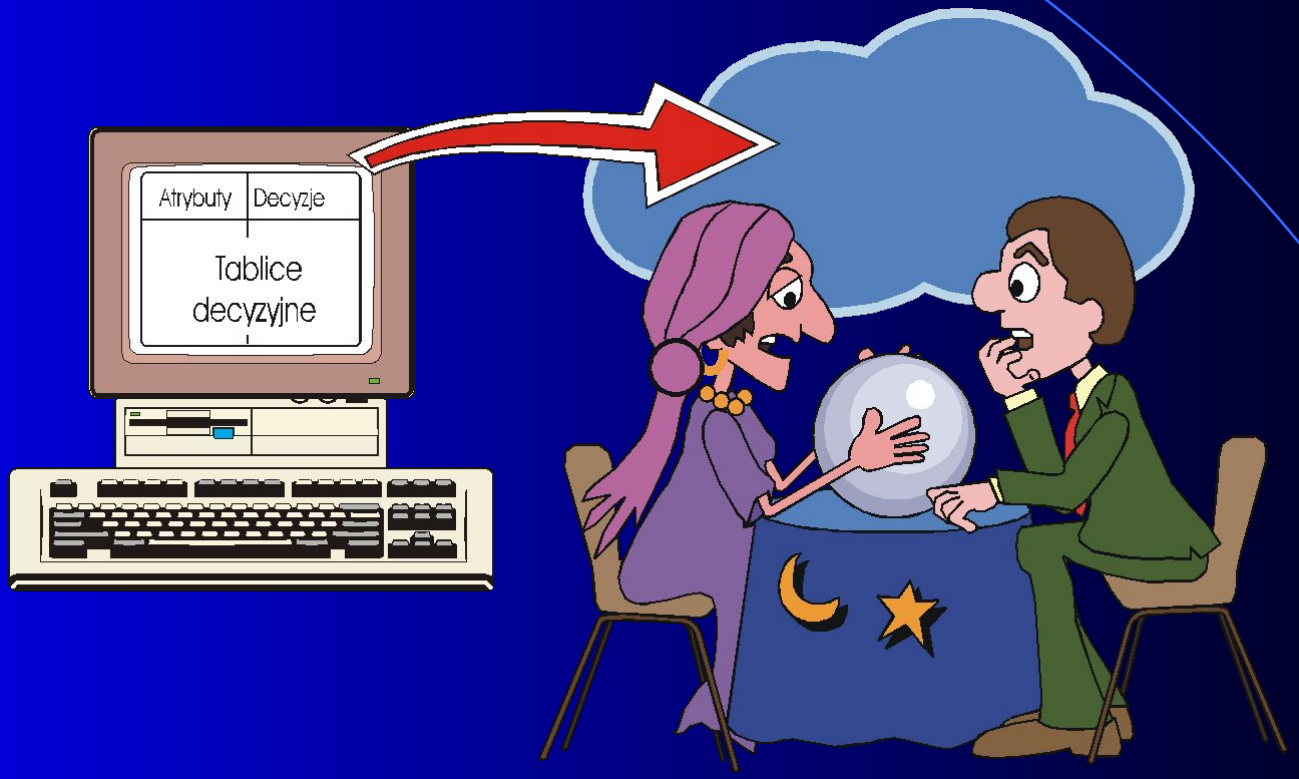
VHDL



2 komórki (Stratix)

Dobrze rozmieszczone puzzle

Metody syntezy logicznej w zadaniach pozyskiwania wiedzy i analizy danych



Przetwarzanie danych zapisanych w tablicach decyzyjnych

Atrybuty:	wiek	płeć	Stan cywilny	zawód	Klasa decyzyjna
x_1	20	Female	Married	Farm	1
x_2	17	Female	Single	Farm	2
x_3	25	Male	Single	Business	3
x_4	16	Female	Single	Farm	2
x_5	38	Male	Single	Business	3
x_6	25	Female	Single	Pleasure	4
x_7	48	Female	Single	Pleasure	4
x_8	20	Female	Single	Farm	2
x_9	21	Male	Married	Business	5
x_{10}	22	Male	Married	Business	5
x_{11}	23	Male	Married	Business	5
x_{12}	24	Male	Married	Business	5

Zastosowania

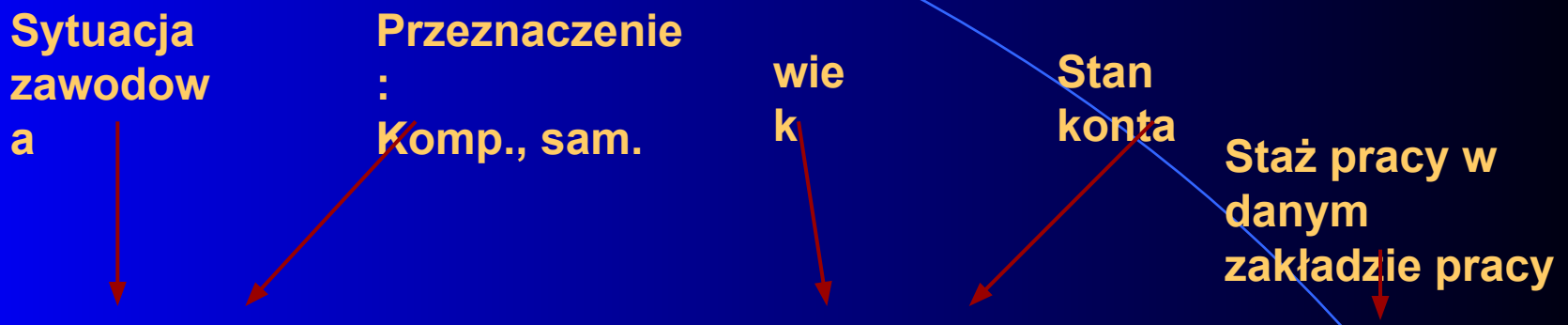
Tablice decyzyjne stosuje się np. przy wnioskach kredytowych składanych w bankach. Ponieważ część z nich jest akceptowana, a część odrzucana, można dane zebrane w dłuższym okresie czasu zapisać w tablicy decyzyjnej, uogólnić i dalej stosować w uproszczonej formie do podejmowania decyzji.

Klientów charakteryzuje się za pomocą następujących cech jakościowych i ilościowych:

Przykładowo:

- **Sytuacja zawodowa: B (bezrobotny), P (pracujący)**
- **przeznaczenie kredytu: komputer (K), sprzęt audio (A), biżuteria (B)…**
- **wiek w latach**
- **stan konta**

Przykładowa tablica danych...



C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	Klasa
P	K	K	S	nie	18	200	20	15	1	tak
P	K	K	S	nie	20	100	20	20	2	tak
B	K	K	R	tak	25	50	40	12	0	nie
					⋮					
P	S	M	R	nie	21	1500	30	20	3	tak
P	S	M	S	nie	25	1500	100	20	2	tak
P	S	M	R	nie	38	1000	100	20	15	tak

Zastosowania

Po uogólnieniu reguł decyzyjnych…

[wiek > 25] & [stan konta > 70] & [staż pracy > 2] → tak

. . . . . .

[płeć = kobieta] & [wiek < 25] → nie

LERS

Podsumowanie

Sytuacja ta czyni z Układów logicznych jedną z najciekawszych dziedzin techniki, której opanowanie może być kluczem do sukcesu zawodowego wielu specjalistów elektroniki, inżynierii komputerowej i telekomunikacji.