Wprowadzenie

Kontakt z prowadzącym

dr inż. Paweł J. Rajda pjrajda@agh.edu.pl
http://www.embedded.agh.edu.pl
C-3, p.502, tel. (12) 617 3980

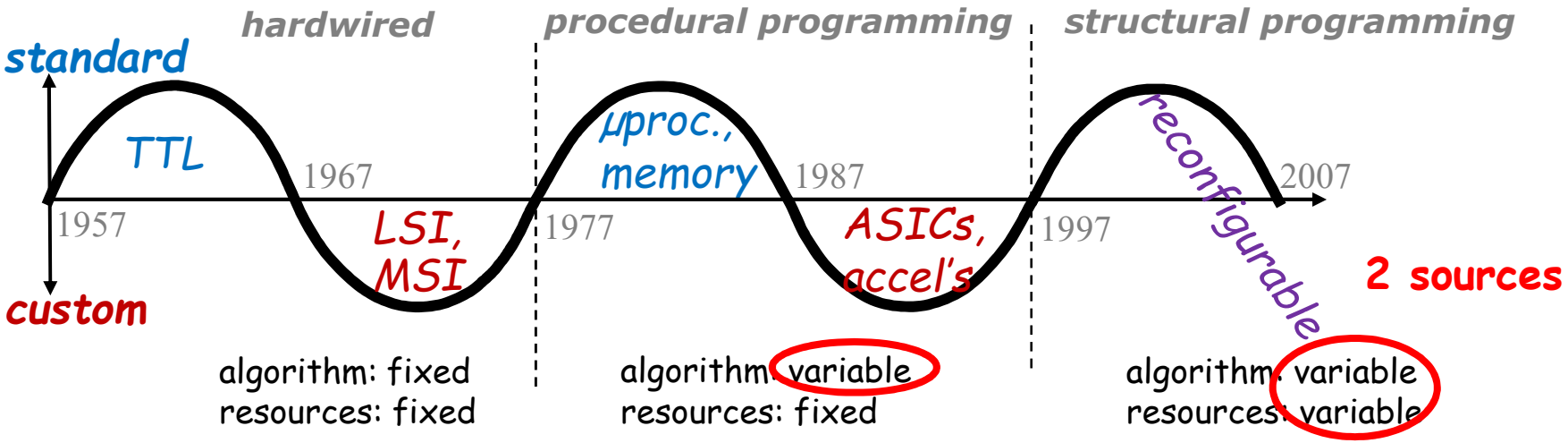
Program wykładu

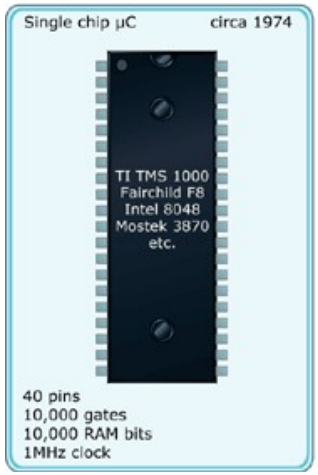
HDL - Dekalog

1. HDL – po co?
2. HDL – co to jest?
3. HDL – jak, gdzie, kiedy?
4. HDL – i co z tego?
5. HDL – wczoraj, dziś i jutro
6. HDL – standardy
7. HDL – literatura
8. HDL – źródła
9. HDL – firmy
10. HDL – Aldec: Active-HDL

Makimoto's Wave
Technologiczne wahadło

Reiner Hartenstein
Model systemu





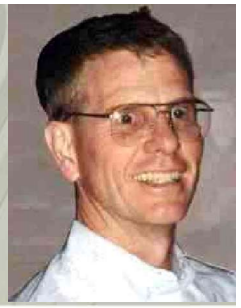
vN machine paradigm

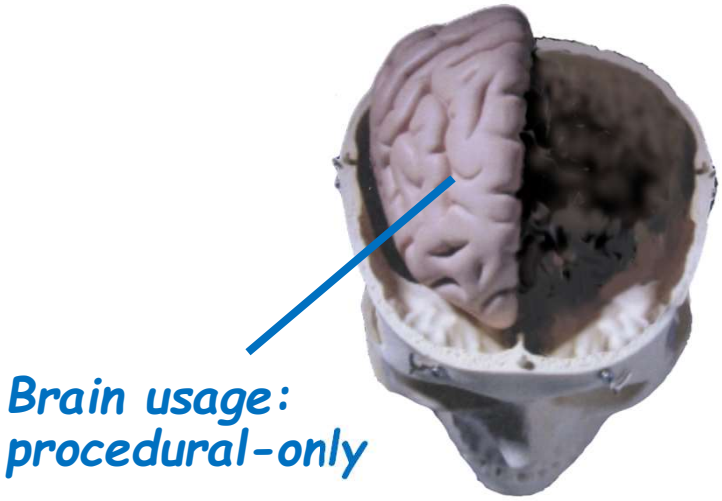
new machine paradigm needed

The Programmable System-on-a-Chip IS THE NEXT WAVE

Reiner Hartenstein

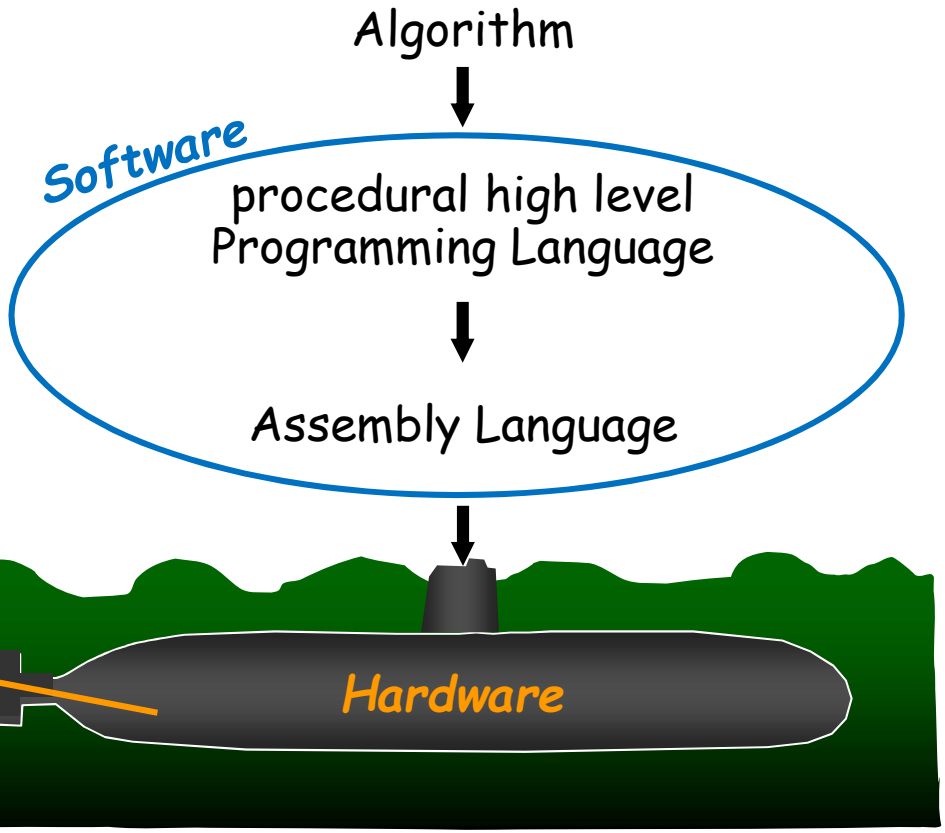
Model dotychczasowy



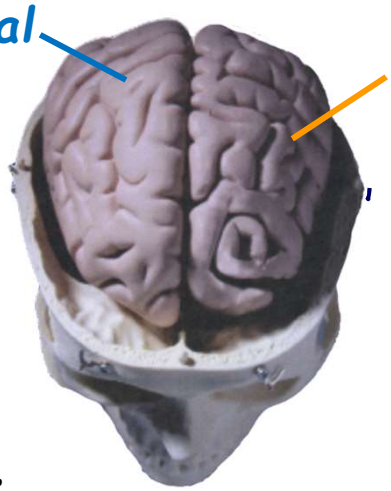


Brain usage:
procedural-only

Hardware invisible:
under the surface

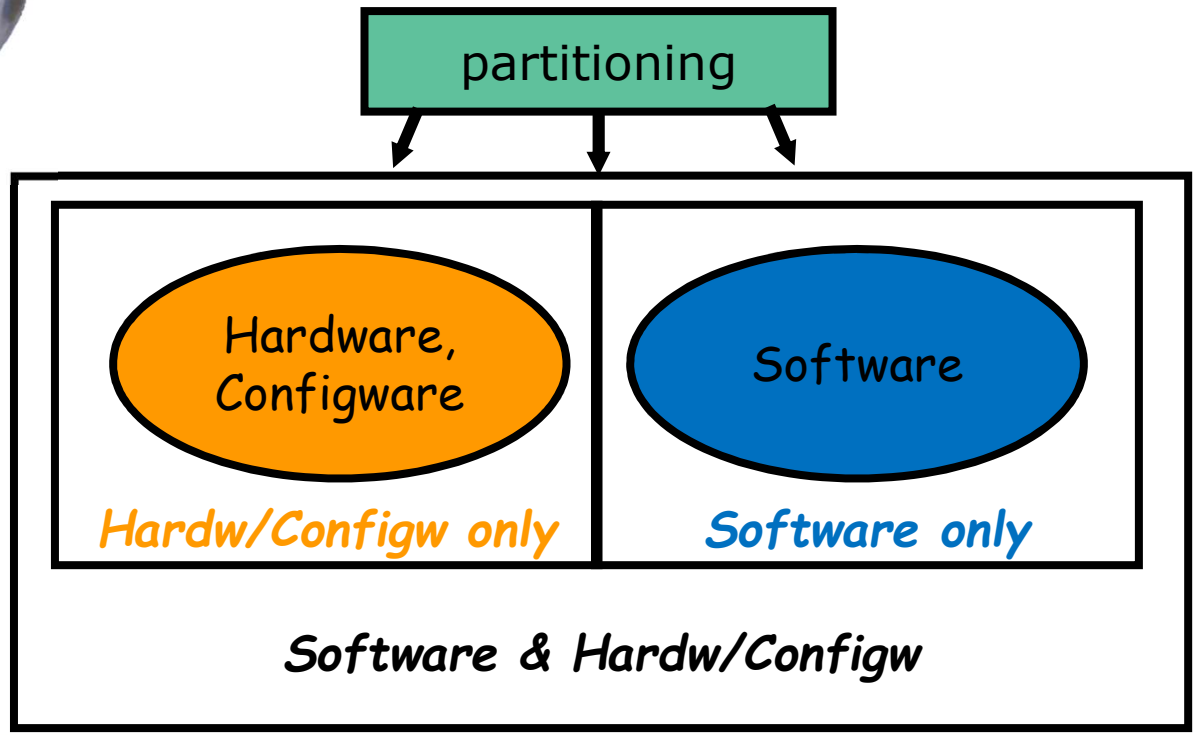


Nowy model

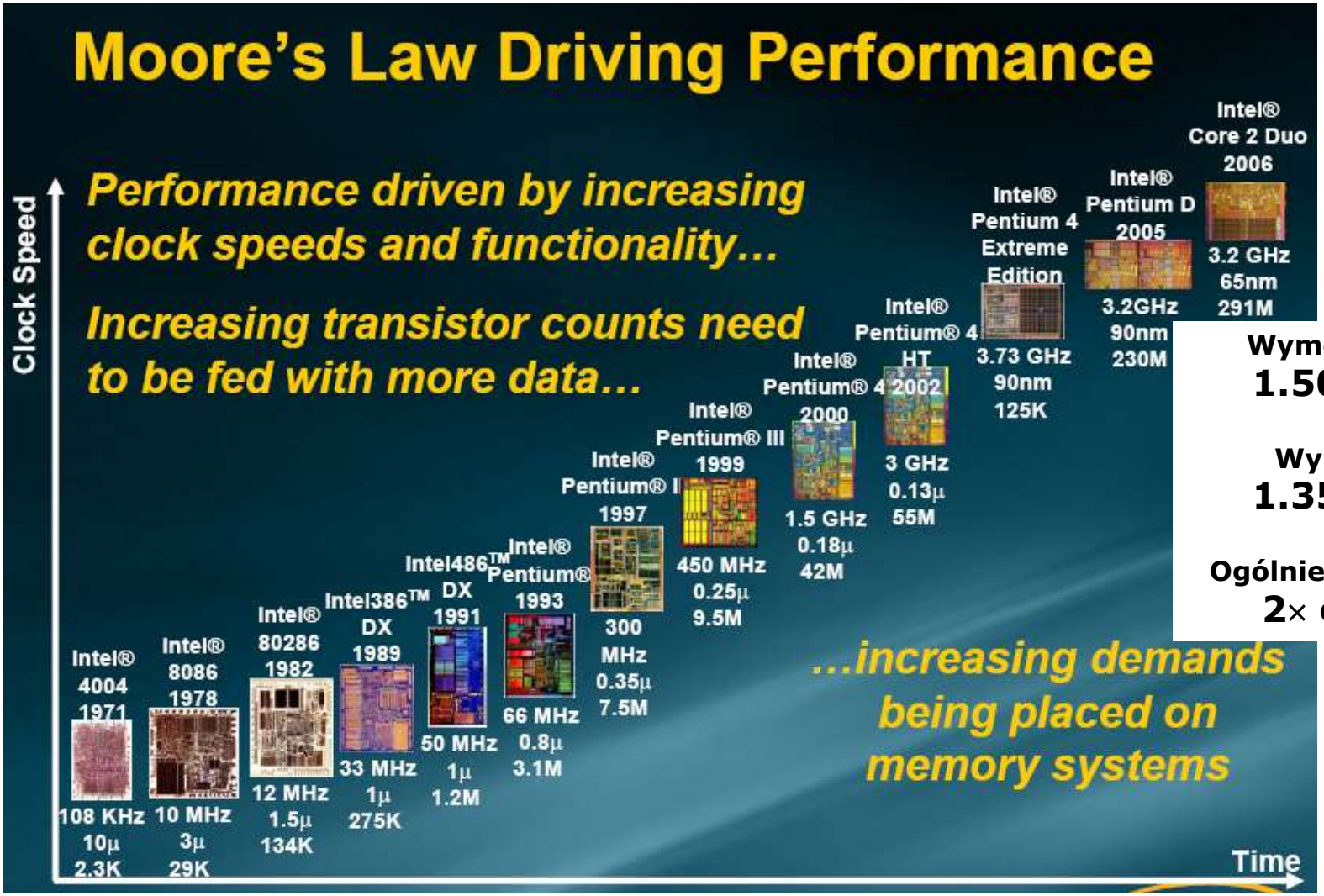


Brain usage:
both hemispheres

Algorithm



Złożoność projektów



Wymogi pamięci:
1.50× na rok

Wymogi CPU:
1.35× na rok

Ogólnie prawo Moora:
2× co 2 lata

Złożoność projektów

Potrzeba narzędzia:

• 1970 - INTEL 4004	4 projektantów	1 tys tranzystorów
• 1982 - INTEL 80286	20 projektantów	100 tys tranzystorów
• 1992 - INTEL PENTIUM	100 projektantów	3 mln tranzystorów
• 2003 - ???	1000 projektantów	150 mln tranzystorów

Współczesne wymagania:

- *hardware-software codesign*
- *design reuse*

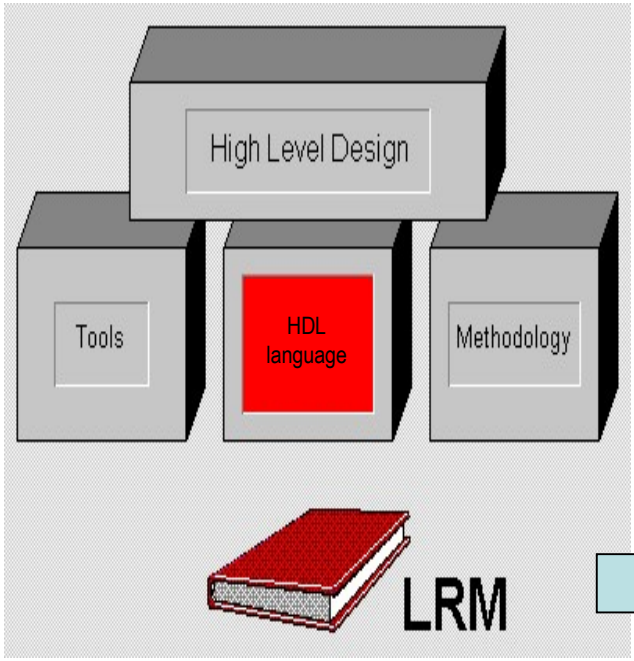
Rozwiązanie:
wykorzystanie języka HDL (VHDL, Verilog, ...)

Języki Opisu Sprzętu

- **PALASM**
- **ABEL**
- **CUPL**

- **VHDL**
- **VERILOG, System VERILOG**
- **C, C++, Handel-C, System-C**
- **inne**

Definicja



It is "a formal notation intended for use in all phases of the creation of electronic systems. (...) it supports the development, verification, synthesis, and testing of hardware designs..."

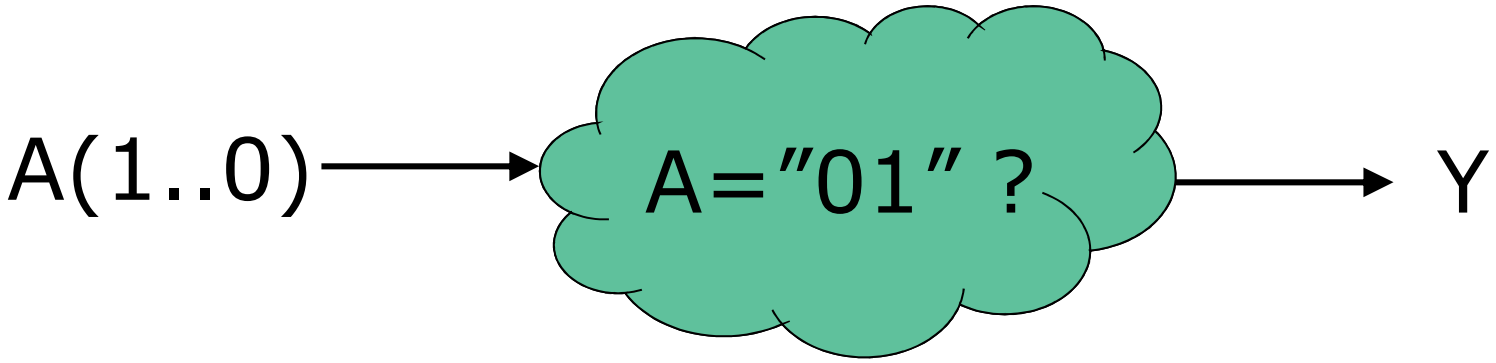
[*IEEE Standard VHDL Language Reference Manual*]
[*IEEE Standard Verilog Language Reference Manual*]

VHDL - VHSIC Hardware Description Language

↳ **Very High Speed Integrated Circuit**

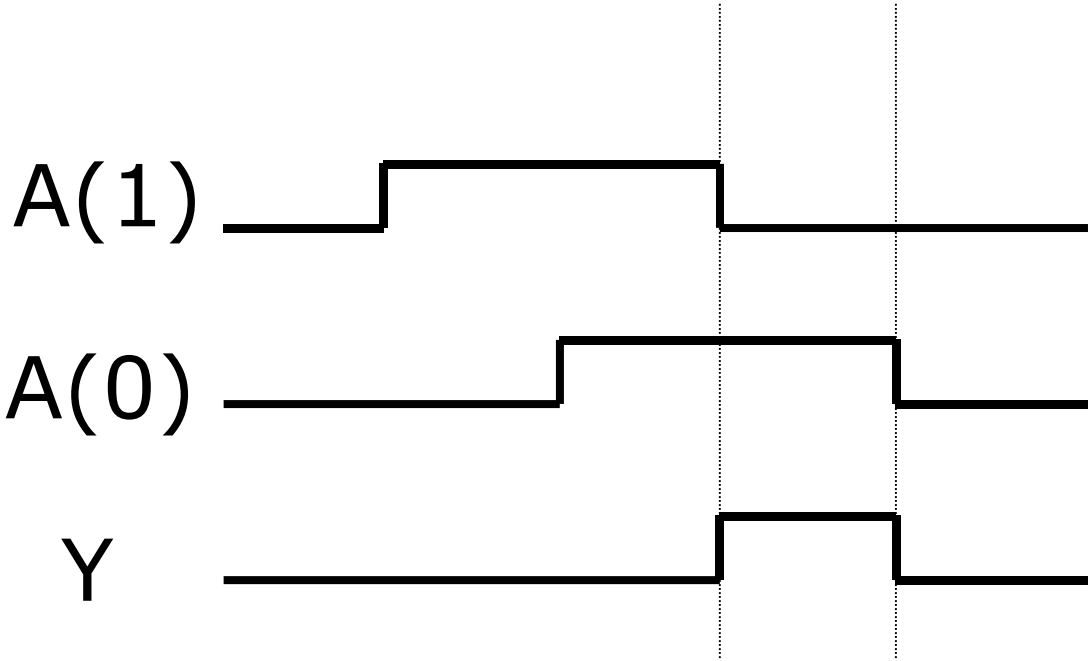
Verilog - Verification Logic

Modelowanie




```
assign Y = (A == 2'b10) ? 1'b1 : 1'b0;
```

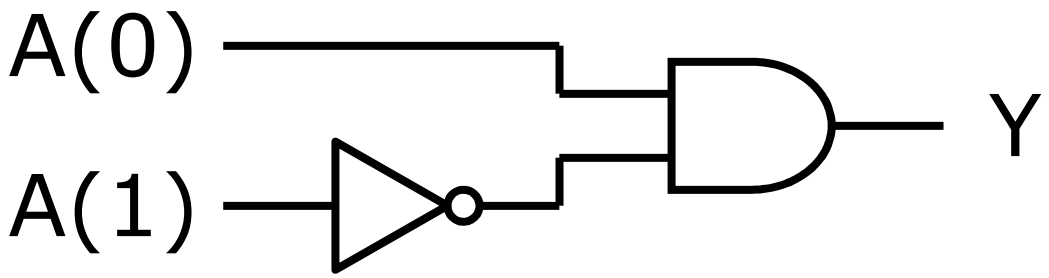
Symulacja



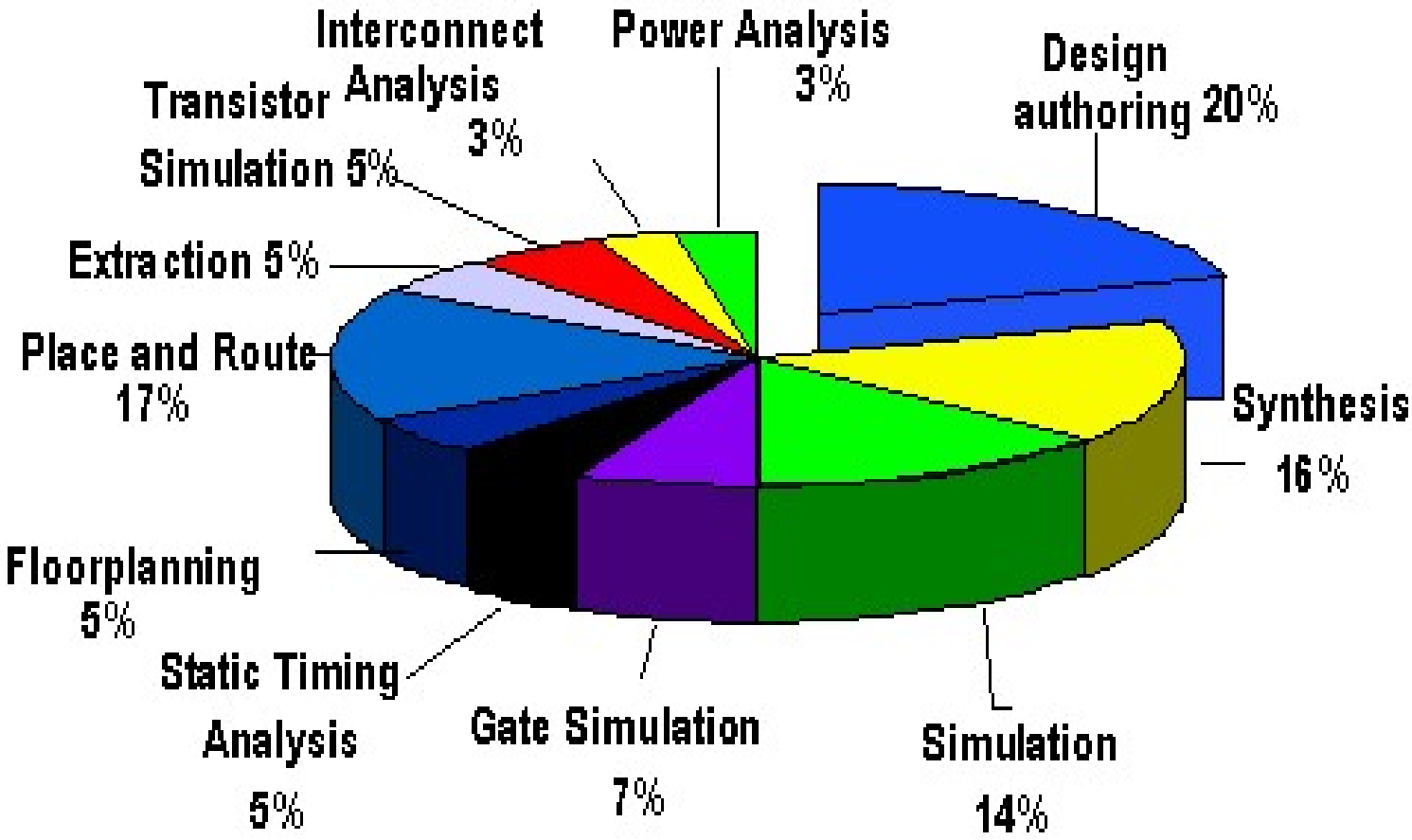

```
    A = 2'b10;
#5  A = 2'b11;
#5  A = 2'b01;
...
```

Synteza

Synteza – (automatyczna) translacja opisu w języku HDL na strukturę w postaci listy połączeń elementarnych bloków funkcyjnych docelowej platformy sprzętowej (bramek, przerzutników, pamięci i innych).

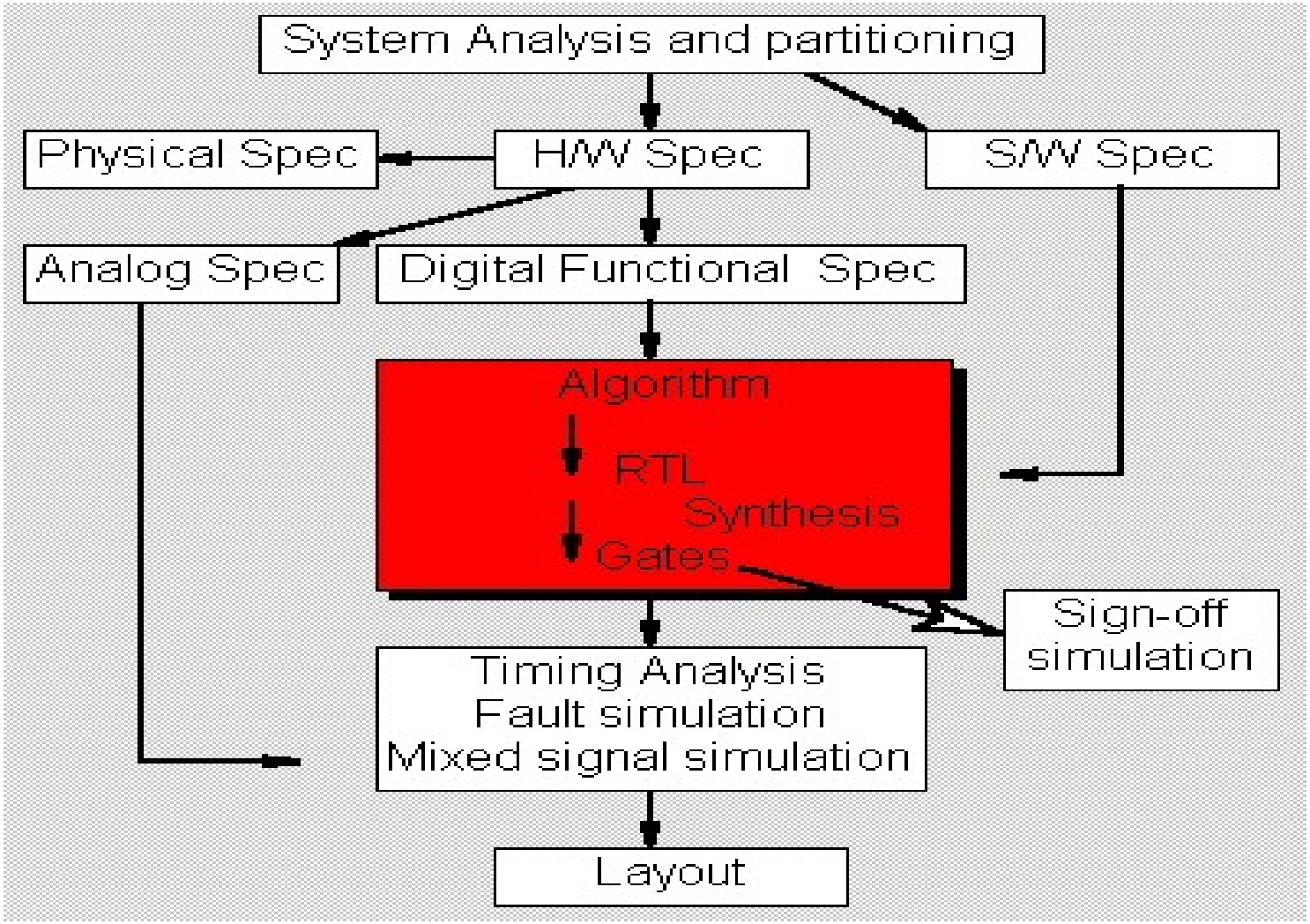


Etapy projektu

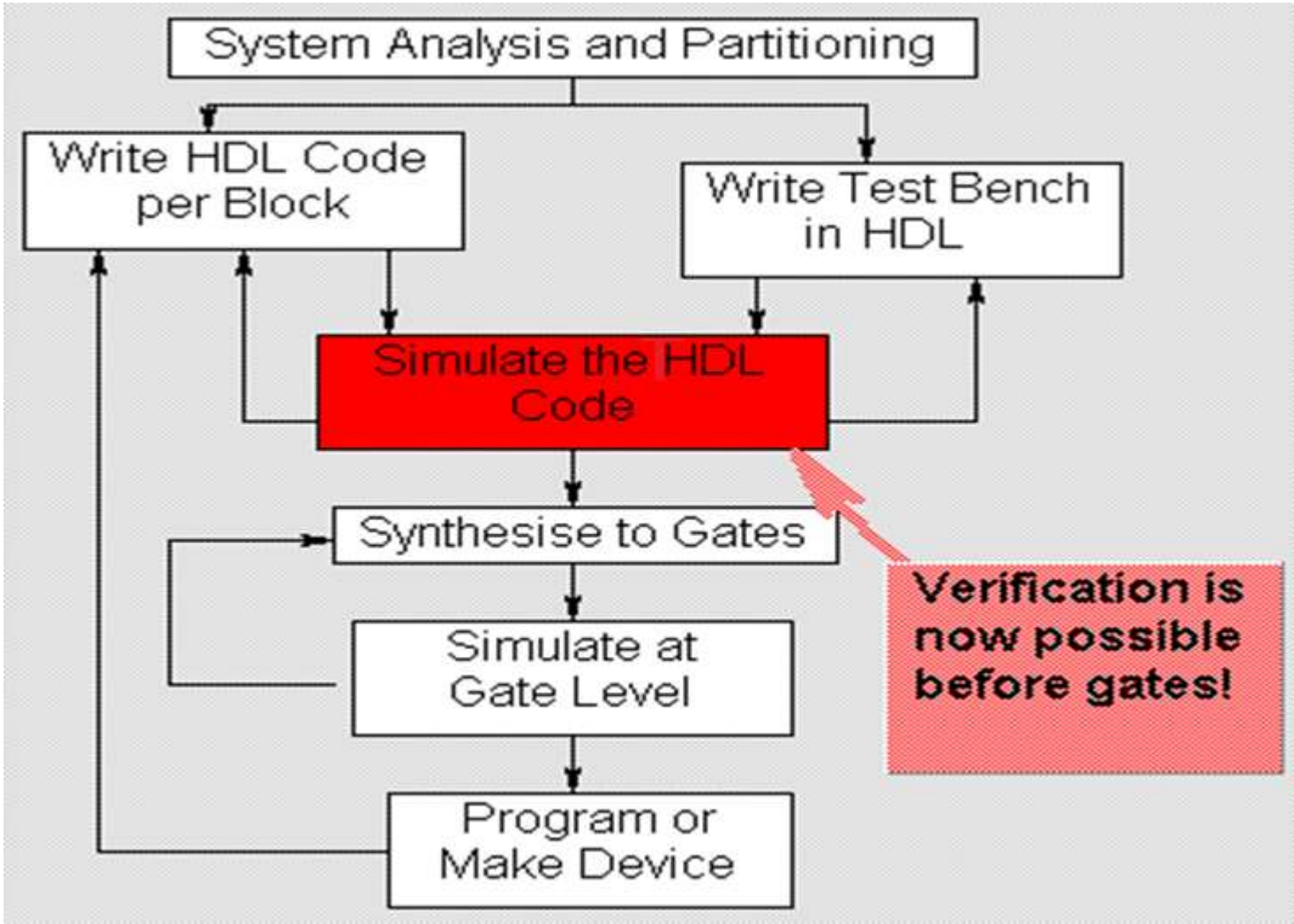


Average iterations between design and layout = 20

Projektowanie systemu



Proces projektowania



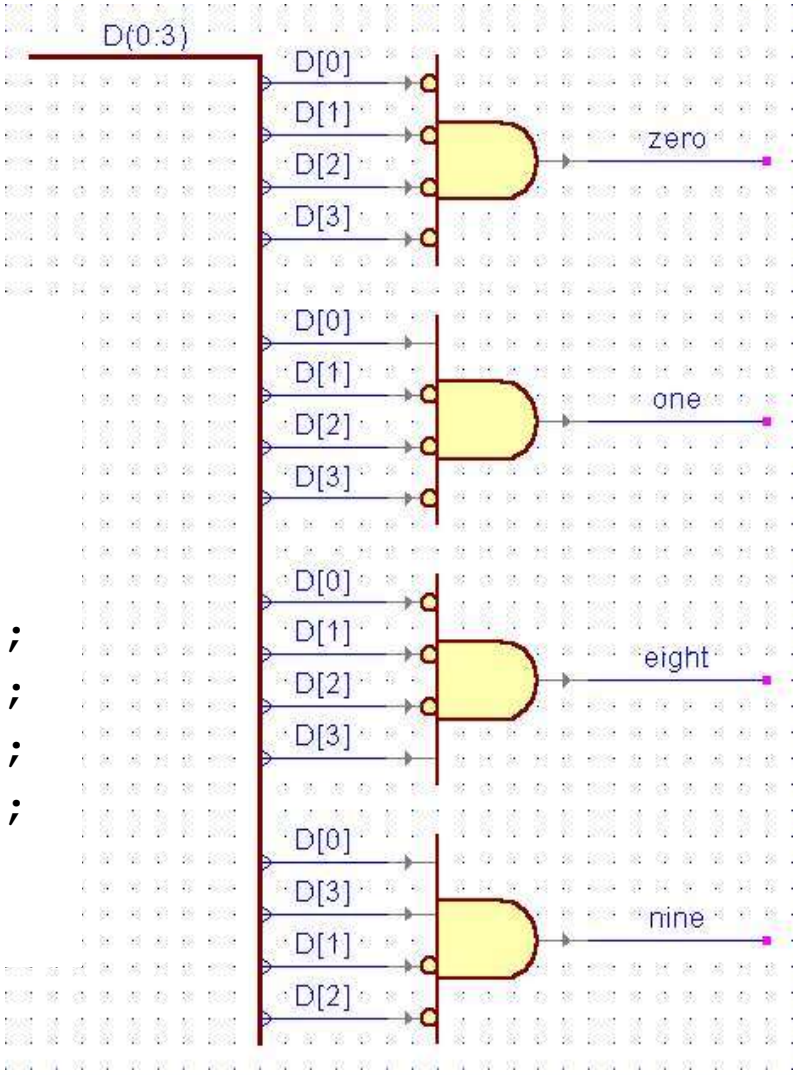
Jednostka projektowa

module

```
module COMPARE
//description of ports
   (A, B, C);
   input A, B;
   output C;

   reg C;

//description of functionality / structure
   always @(A or B)
     begin
       if (A == B)
          C = 1'b1;
        else
          C = 1'b0;
     end
endmodule
```

Dekoder

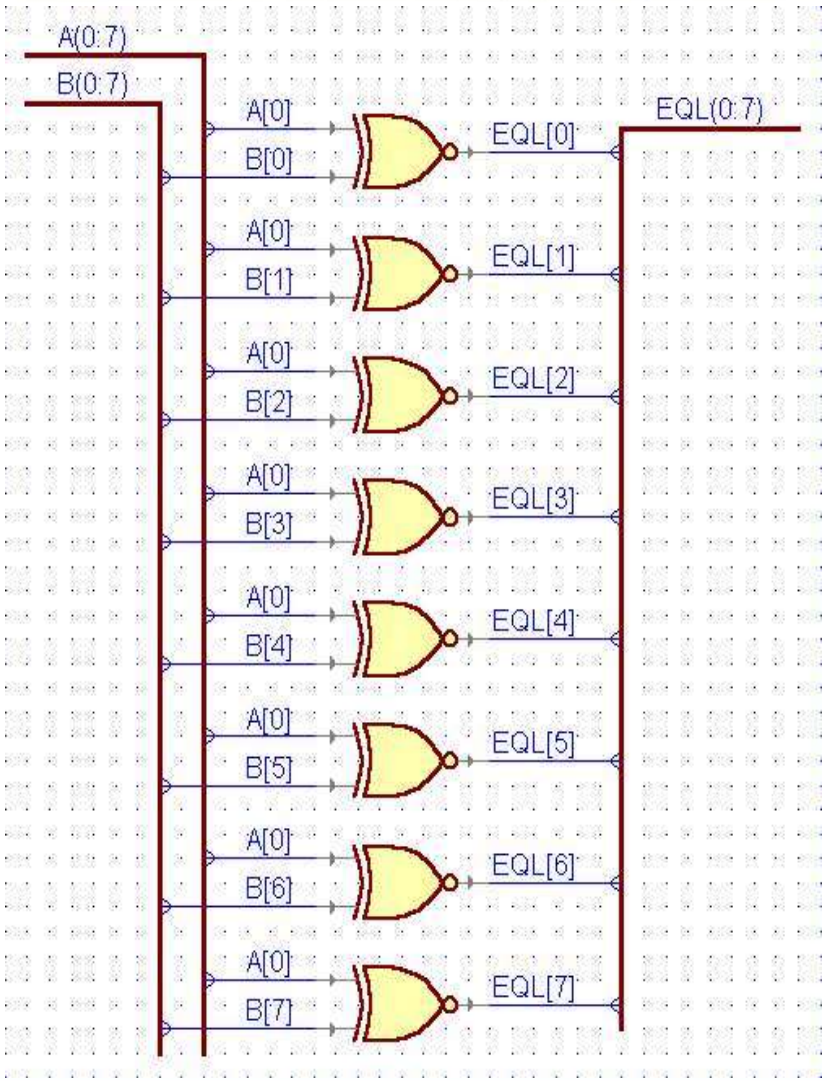
```
module DECODER
   (D, ZERO, ONE, EIGHT, NINE);
   input [3:0] D;
   output ZERO, ONE, EIGHT, NINE;

   assign ZERO  = (D==4'd0)?1'b1:1'b0;
   assign ONE   = (D==4'd1)?1'b1:1'b0;
   assign EIGHT = (D==4'd8)?1'b1:1'b0;
   assign NINE  = (D==4'd9)?1'b1:1'b0;

endmodule
```



Komparator

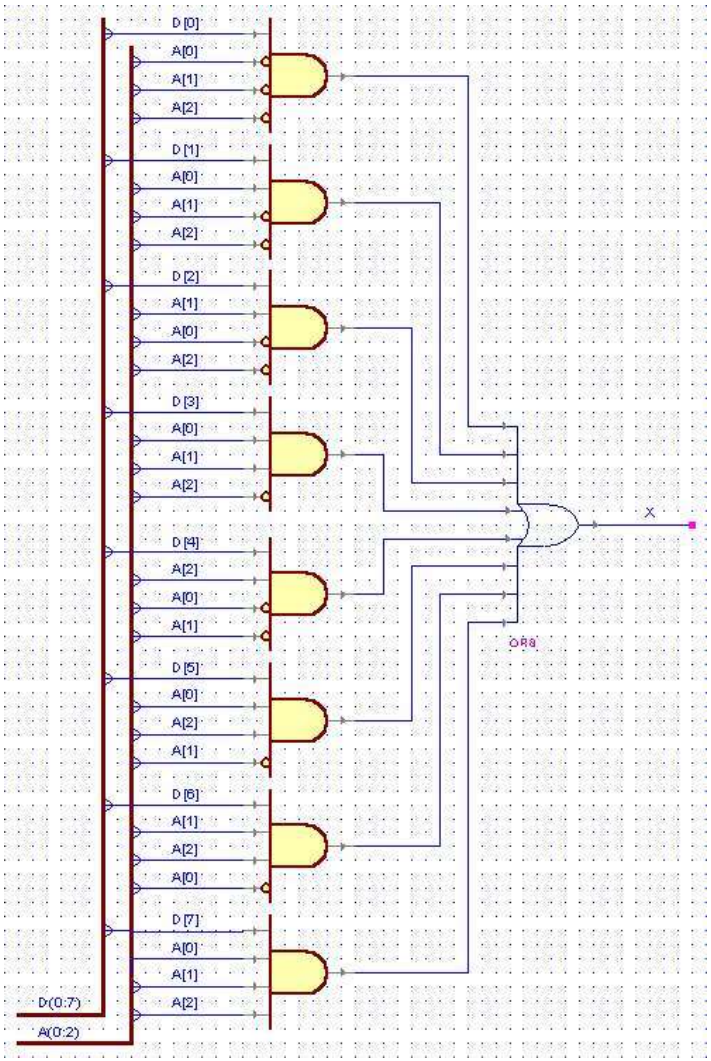
```
module COMPARE
   (A, B, EQL);
   input  [7:0] A, B;
   output [7:0] EQL;

   assign EQL = ~(A | B);

endmodule
```

Błąd !



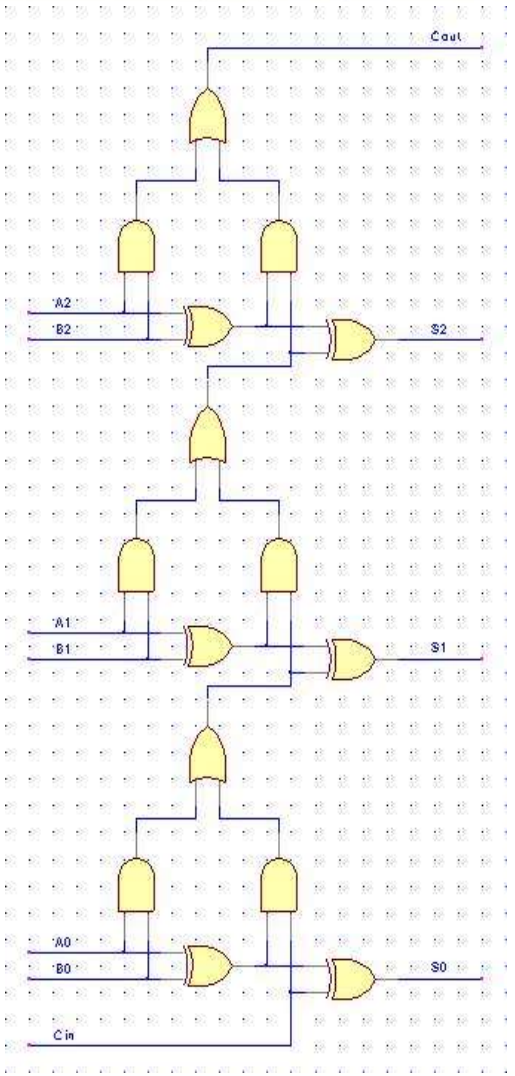
Multiplekser

```
module MPLEXER
   (D, A, X);
   input [7:0] D;
   input A;
   output X;

   int A;

   assign X = D[A];

endmodule
```



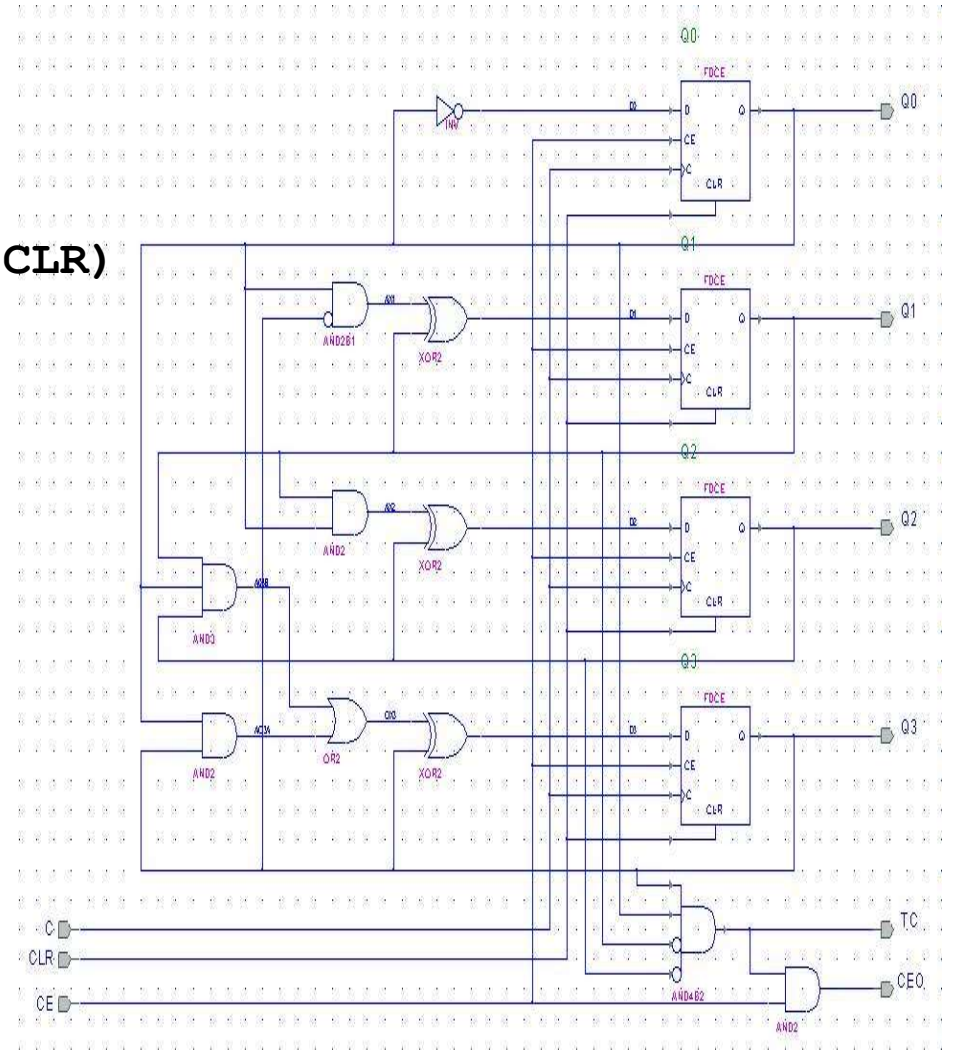
Sumator

```
module SUM
   (A, B, S, Cin, Cout);
   input  [2:0] A, B;
   output [2:0] S;
   input  Cin;
   output Cout;

   wire [3:0] V;

   assign V = A + B + Cin;
   assign S = V[2:0];
   assign Cout = V[3];

endmodule
```



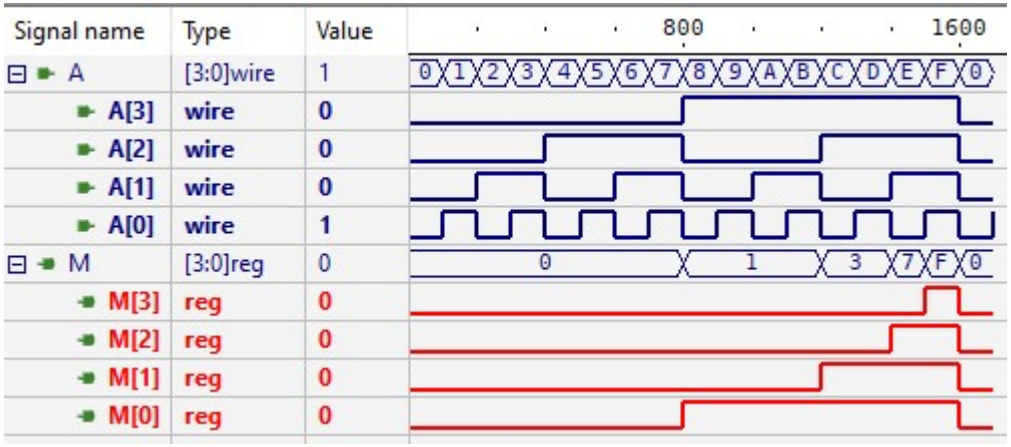
Licznik

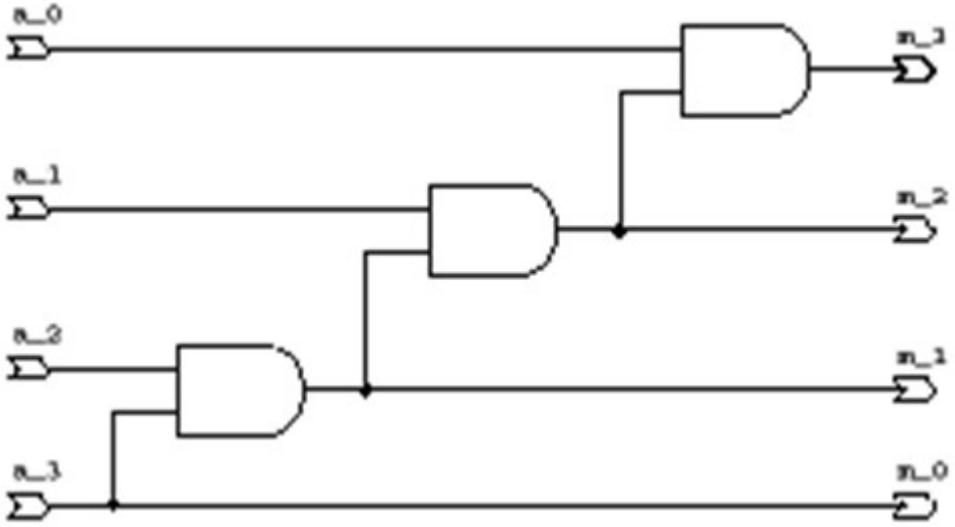
```
module COUNT
   (C, CE, CLR, Q, TC, CEO);
   (…)

   always @(posedge C or negedge CLR)
     if (!CLR)
       Q = 'd0;
     else
       if (CE)
         if (Q < 'd9)
           Q = Q + 'd1;
         else
           Q = 'd0;

   assign TC = (Q=='d9)?'b1:'b0;
   assign CEO = TC & CE;

endmodule
```



Powielanie logiki

```
module LOOP
   (A, M);
   input  [3:0] A;
   output [3:0] M;
   reg    [3:0] M;

   integer i;
   reg B;

   always @ (A)
   begin
     B = 'b1;
     for (i=0; i<=3; i=i+1)
     begin
        B = A[3-i] & B;
        M[i] = B;
     end
   end
endmodule
```





Zalety

- Specyfikacja projektu niezależna od technologii
  - możliwość współpracy z wieloma producentami
  - uniknięcie problemów z wycofywanymi technologiami
  - łatwe ulepszenia i poprawy
- Automatyzacja projektowania niskiego poziomu
  - krótszy czas projektowania
  - redukcja kosztów
  - eliminacja błędów niskiego poziomu
- Poprawa jakości projektu
  - łatwe sprawdzanie opcjonalnych technologii
  - weryfikacja działania na wysokim poziomie
  - łatwa weryfikacja implementacji
  - modularność projektu – łatwa wymiana do innych celów

Porównianie

VHDL	VERILOG
Uważa się go za trudniejszy do nauczenia. Złożona składnia, bazuje na językach Ada i Pascal. Rzadziej daje mniej mylne wrażenie co do istoty projektu.	Uważa się go za łatwiejszy do nauczenia. Prosta składnia, bazuje na języku C. Przez to może dawać mylne wrażenie co do istoty projektu.
Kod jest nieco bardziej rozwlekły, jednak lepiej się samo-tłumaczy - wiele elementów składni jest słowami.	Kod jest nieco bardziej zwięzły, jednak gorzej się samo-tłumaczy - wiele elementów składni jest symbolami.
Dla projektów *reliability* (lotnictwo, kosmos) używany wyłącznie.	Szeroko używany w przemyśle *consumer*.
Wymaga więcej wysiłku dla uruchomienia wstępnej weryfikacji.	Pozwala na szybką i łatwą wstępną weryfikację.
Oferuje liczne predefiniowane typy danych i restryktywną politykę weryfikacji. Błędy związane z typami danych rzadko mogą nie zostać wykryte na wstępnych etapach obróbki kodu.	Oferuje nieliczne predefiniowane typy danych i liberalną politykę weryfikacji. Błędy związane z typami danych często mogą nie zostać wykryte na wstępnych etapach obróbki kodu.

Porównianie

VHDL	VERILOG
Ma bogatsze wsparcie dla abstrakcji (np. wielowymiarowe tablice, deklaracje nowych typów), przez co obróbka złożonych systemów jest łatwiejsza.	Ma ograniczone wsparcie dla abstrakcji (np. jednowymiarowe tablice, brak deklaracji nowych typów), przez co obróbka złożonych systemów jest trudniejsza.
Nie wspiera projektowania niskopoziomowego, na poziomie kluczy - nadaje się bardziej dla platform FPGA.	Wspiera projektowanie niskopoziomowego, na poziomie kluczy - nadaje się bardziej dla platform ASIC.
Silne wsparcie dla weryfikacji formalnej.	Brak silnego wsparcia dla weryfikacji formalnej.
Silne wsparcie dla struktur współbieżnych, jednak złożoność języka ogranicza możliwości osiągnięcia maksymalnej współbieżności.	Brak silnego wsparcia dla struktur współbieżnych.
Silne wsparcie dla kodowania obiektowego, co ułatwia projektowanie złożonych systemów.	Brak silnego wsparcia dla kodowania obiektowego, co utrudnia projektowanie złożonych systemów.

HDL – wczoraj

ROK	VHDL	Verilog
1980	DoD USA – program rozwijania układów VHSIC	
1981	konferencja na temat założeń przyszłego standardu HDL	
1982	DoD ustala założenia VHDL kontrakt otrzymują: Intermetrics, TI i IBM	
1983	gotowa wersja VHDL 6.0	
1984	zwolnienie z restrykcji ITAR VHDL 7.2 przekazany do standaryzacji IEEE	zaprojektowany przez Gateway Design Automation
1985	**wydany opis IEEE Std 1076**	
1990		przejęcie przez Cadence, założenie OVI
1983	nowelizacja IEEE Std 1076-1993	
1995		**wydany opis IEEE Std 1364**
2000	errata IEEE Std 1076a-1993	
2001		nowelizacja IEEE Std 1364-2001
2003	nowelizacja IEEE Std 1076-2003	
2005	przejęcie inicjatywy przez Accelera	nowelizacja IEEE Std 1364-2005
2006	nowelizacja IEEE Std 1076-2006	
2008	nowelizacja IEEE Std 1076-2008	
2009		włączenie do SystemVerilog IEEE Std 1800-2009
2012		nowelizacja IEEE Std 1800-2012
2017		nowelizacja IEEE Std 1800-2017
2019	nowelizacja IEEE Std 1076-2019	
2023		nowelizacja IEEE Std 1800-2023

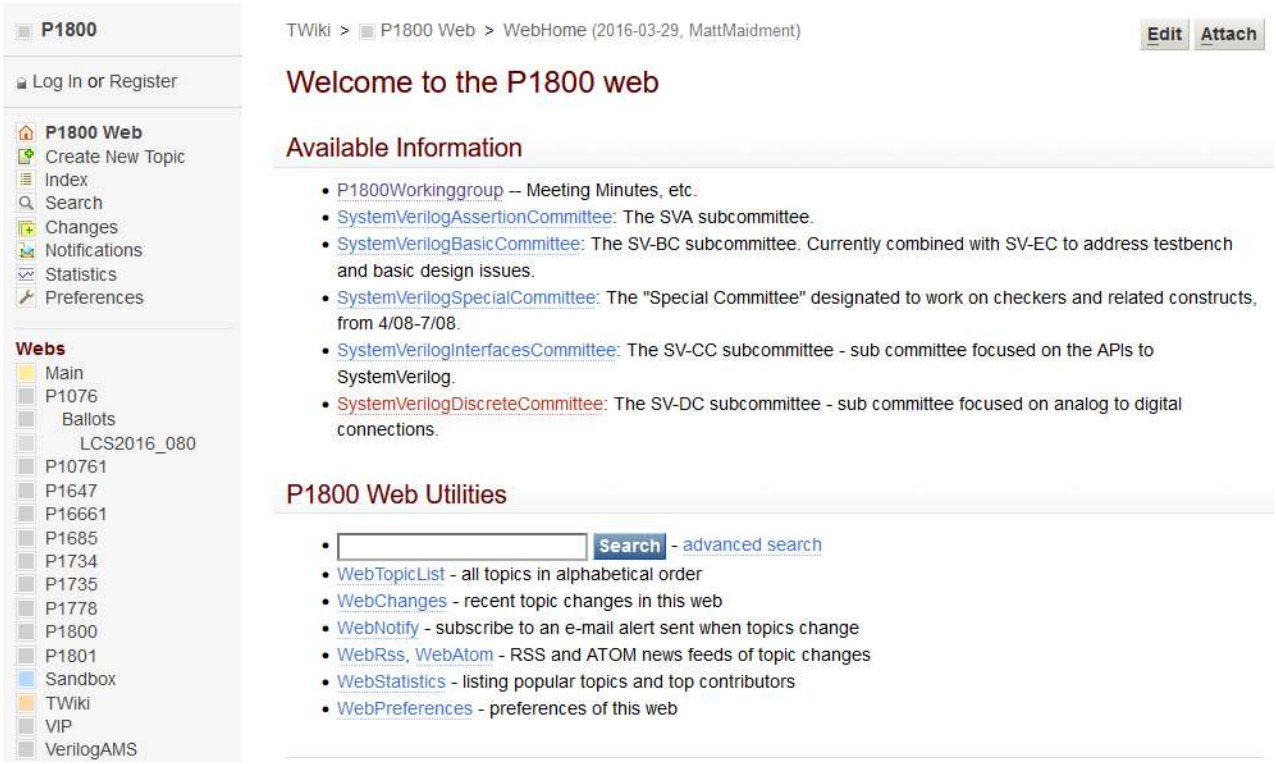
VHDL – dziś

IEEE P1076 Working Group
VHDL Analysis and Standarization Group (VASG)
http://www.eda-twiki.org/cgi-bin/view.cgi/P1076/WebHome

Verilog – dziś

IEEE P1800 Working Group

http://www.eda-twiki.org/cgi-bin/view.cgi/P1800/WebHome



HDL – jutro

- Czy „sprzętowiec” jest jeszcze potrzebny?
  - zbyt mała wydajność architektury Von Neumanna
  - konieczność kompensacji „niewydajności” oprogramowania przy pomocy dedykowanego sprzętu
- Język: ogólny czy szczególny ?
  - ogólny (RTL)
    - blisko implementacji sprzętowej
    - w projekcie są zawarte szczegóły architektury
  - szczególny
    - blisko aplikacji (np. DSP: Matlab)
    - automatyczna generacja zrównoleglonego sprzętu
- Język: dwa podejścia
  - nowy język, dedykowany do potrzeb ⇨ adopcja przez użytkowników
  - adaptacja (do nowego kontekstu) języka już istniejącego

HDL – standardy

- IEEE Std 1076/INT-1991, IEEE Standards Interpretations: IEEE Std 1076-1987 IEEE Standard VHDL Language Reference Manual
- IEEE Std 1076-1993, IEEE Standard VHDL Language Reference Manual
- IEEE Std 1076a-2000, Amendment to IEEE Std 1076-1993
- IEEE Std 1029.1-1998, IEEE Standard for VHDL Waveform and Vector Exchange (WAVES) to Support Design and Test Verification
- IEEE Std 1076.1-1999, IEEE Standard VHDL Analog and Mixed-Signal Extensions (VHDL-AMS)
- IEEE Std 1076.2-1996, IEEE Standard VHDL Mathematical Packages
- IEEE Std 1076.3-1997, IEEE Standard VHDL Synthesis Packages
- IEEE Std 1076.4-1995, IEEE Standard for VITAL Application-Specific Integrated Circuit (ASIC) Modeling Specification
- IEEE Std 1076.6-1999, IEEE Standard for VHDL Register Transfer Level Synthesis
- IEEE Std 1164-1993, IEEE Standard Multivalue Logic System for VHDL Model Interoperability

---

- IEEE 1364.1-2002, Standard for Verilog Register Transfer Level Synthesis
- IEEE 1800.2-2017, Standard for Universal Verification Methodology Language Reference Manual

Verilog – literatura

Biblioteka Główna AGH

- *A Verilog HDL Primer, J. Bhasker*
- *Complete Verilog Book, V. Sagdeo*
- *Design through Verilog HDL, T.R. Padmanabhan*
- *Digital System Designs and Practices: Using Verilog HDL amd FPGAs, M.B. Lin*
- *Real world FPGA design with Verilog, K. Coffman*
- *Verilog Coding for Logic Synthesis, W.F. Lee*
- *Verilog Digital System Design, Z. Navabi*
- *Verilog Hardware Description Language, D.E. Thomas, P.R. Moorby*
- *Verilog HDL: A Guide to Digital Design and Synthesis, S. Palnitkar*
- *Verilog HDL: digital design and modeling, J.J.F. Cavanagh*
- *Verilog HDL synthesis: a practical primer, J. Bhasker*

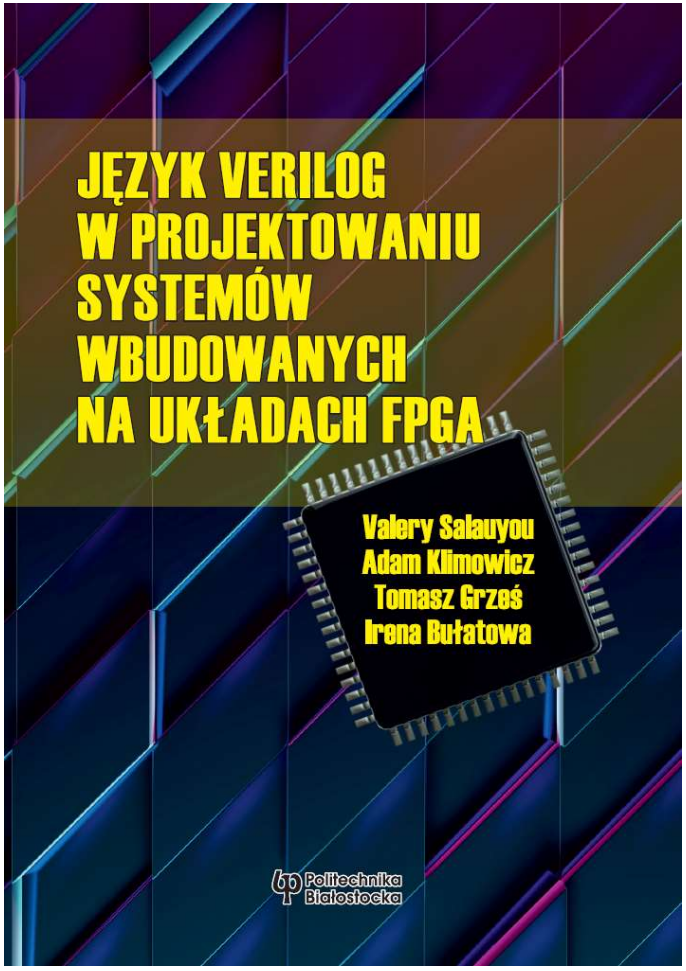
Verilog – literatura

- *Verilog Quickstart - Practical Guide to Simulation & Synthesis in Verilog, J.M. Lee*
- *Verilog Styles for Synthesis of Digital Systems, D.R. Smith, P.D. Franzon*
- *Wprowadzenie do języka VERILOG, Z. Hajduk*
- *Digital Design Techniques and Exercises, V. Taraate*

Inne źródła

- *Język Verilog w projektowaniu systemów wbudowanych na układach FPGA, Politechnika Białostocka*
- *FPGA Prototyping by Verilog Examples, P.P. Chu*
- *Digital Design - An Embedded Systems Approach Using Verilog, P.P. Ashenden*
- *Verilog Language and Simulation - Lecture Manual, Cadence*
- *Verilog Golden Reference Guide, Doulos*

Verilog – literatura



Verilog – zasoby w Internecie

Verilog development

- Accellera: http://www.accellera.org/
- IEEE P1800 Working Group
  http://www.eda-twiki.org/cgi-bin/view.cgi/P1800/WebHome

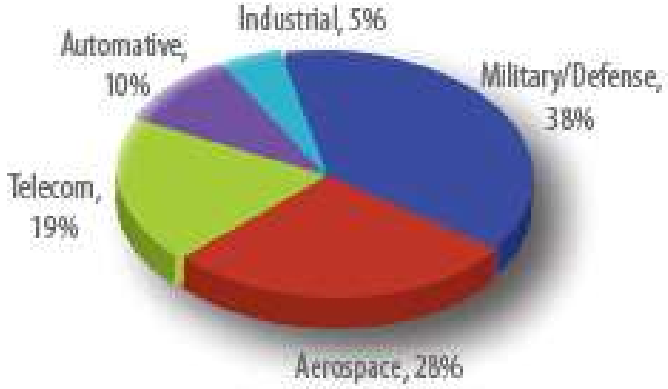
Verilog tutorials

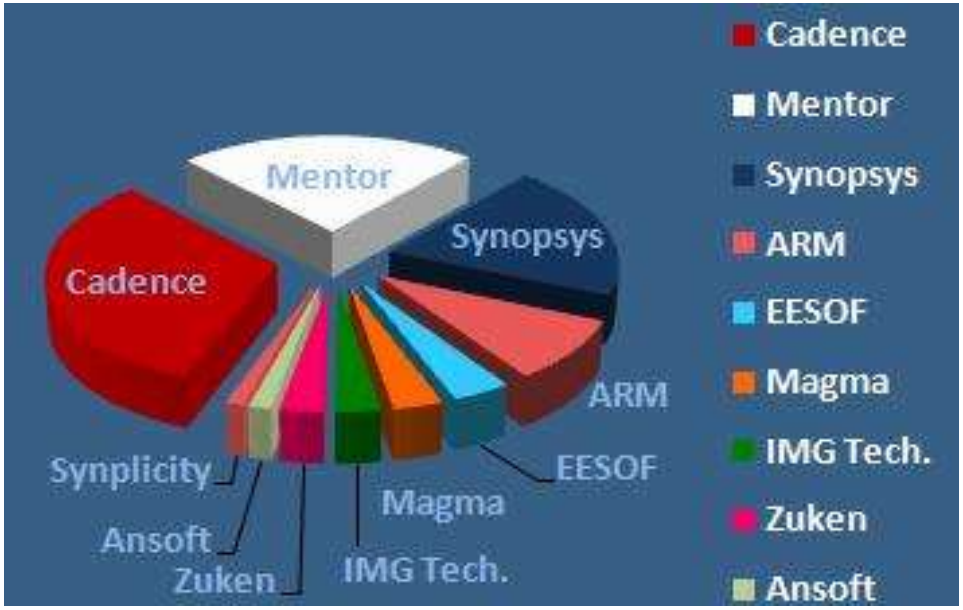
- ChipVerify: Verilog Tutorial
  https://www.chipverify.com/tutorials/verilog
- Doulos: The Designer's Guide to Verilog
  https://www.doulos.com/knowhow/verilog/
- Aldec: EVITA Interactive Verilog Tutorial
  http://www.aldec.com/products/tutorials/
- NAND LAND: Verilog Tutorials and Examples
  https://nandland.com/learn-verilog/
- Free Learning Platform For Better Future
  https://www.javatpoint.com/verilog

Verilog – *free IP cores*

- OpenIP home page: http://www.opencores.org/
- Design & Reuse: https://www.design-reuse.com/
- ALSE: https://www.alse-fr.com/Free-IPs.html
- RISC-V: https://riscv.org/exchange/
- Micron Technology, Inc. (memories): http://www.micron.com/
- 860 VHDL/Verilog Free IP Cores: https://github.com/fabriziotappero/ip-cores
- FREE IP Cores https://www.asics.ws/fip_sub.html
- FREE IP Cores https://github.com/freecores
- .....

HDL – firmy (produkty)

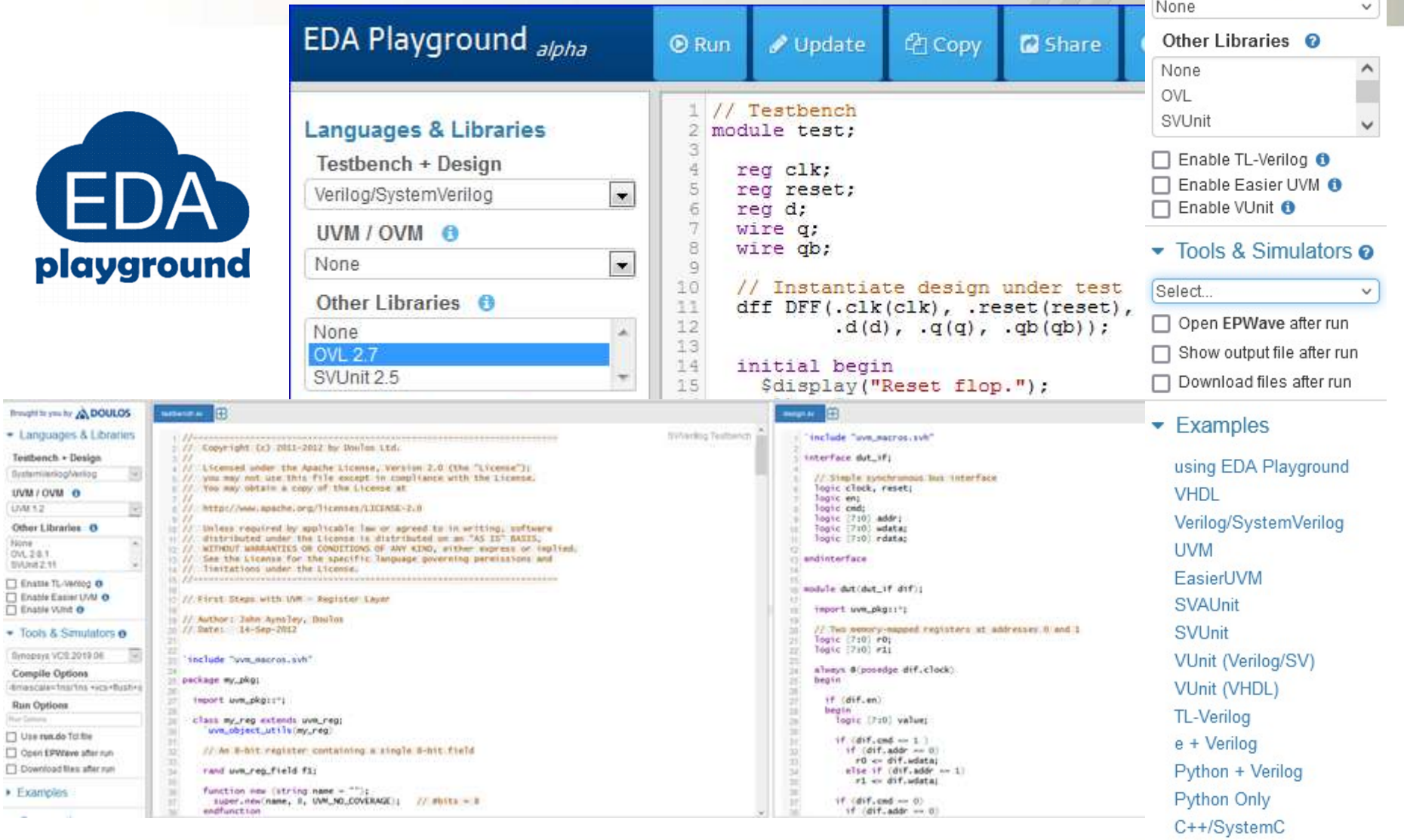




- **Aldec (*Active-HDL, Riviera*)**
- **Altium (*Altium Designer*)**
- **Cadence Design Systems**
- **Mentor Graphics (*ModelSim*) => Siemens**
- **Synopsys (VCS)**
- **Xilinx *(Vivado)* => AMD**
- **Altera (Quartus) => Intel**

- **Green Mountain Computing Systems *(Direct VHDL – low cost)***

HDL – EDA playground



HDL – Aldec Inc.

- **Produkty**
  - ***Software* (ActiveCAD, ActiveHDL)**
  - ***Hardware-based Simulation Accelerators***
  - ***IP Cores***

- **Donacja dla laboratorium:**
  - **Komputery – 11×PC**
  - **Oprogramowanie (ActiveHDL)**
    - ***20 sites license (Professional Edition)***
    - ***Student Edition***
  - **Karty HES**

- **<u>Oddział Kraków!</u>**

HDL – Aldec Inc.

- **Praktyki studenckie**
- **Prace dyplomowe *(IP Cores)***
  - **μP/μC: PIC17C4x, PIC16C5x, 8051-SoC**
  - **peryferia: 8251-SIO, 8255-PIO, 8257-DMA, 8259-INT**
  - **interfejsy: CAN, UART/IrDA, USB, I2S, PS/2, VGA, I2C, SD, …**
  - **telekom: Reed-Solomon, Viterbi, Utopia, DTMF, MP3, ADPCM, LDPC, Kasumi, …**
- **Praca zawodowa**

HDL – dyplomy

Embedded Systems Group

OPERATING SYSTEMS FPGA SYSTEMS DSP SYSTEMS

DYPLOMY

Inżynierskie
Magisterskie
Zalecenia redakcyjne

OSTATNIE WPISY

Nagroda za dyplom
Aldec Praktyki 2020
Astronarium: CTA
Narzędzia mroku
Generator z NCO
Radar
Wyświetlacz OLED
Kryptonim "Virushaus"
Termometr RGB LED
Otto Hahn

INTERNAL

Bryk
Wtyczki
Nowa

INŻYNIERSKIE

Dla każdej pracy dyplomowej wyszczególniono: temat, opiekuna oraz zamawiającego (firma, projekt badawczy itp.)
Kolory oznaczają:

- zielony — temat wolny
- czerwony — temat zajęty (realizacja w toku)
- szary — temat wstrzymany
- czarny — temat już zrealizowany

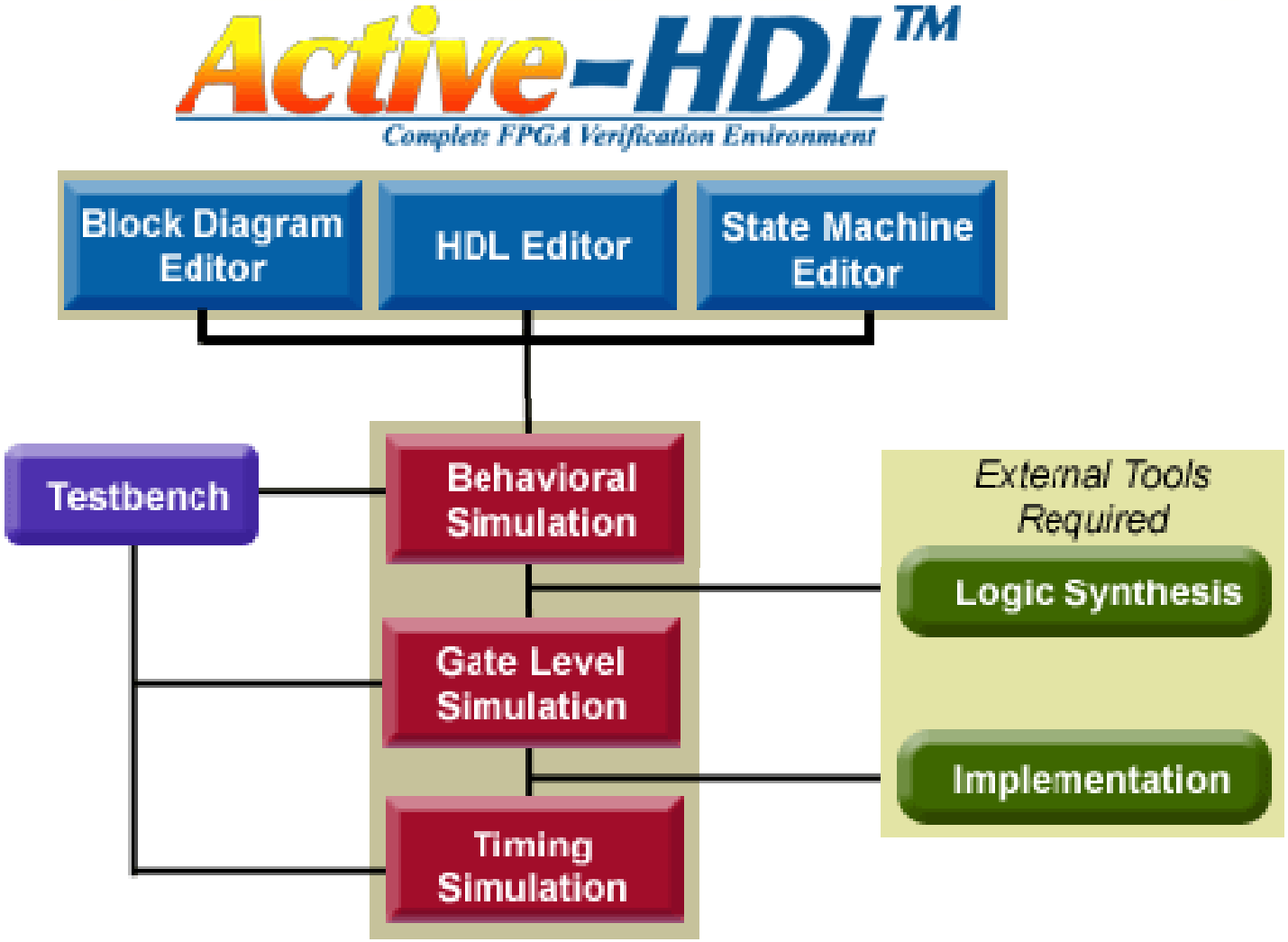
2021

46. Implementacja sprzętowa oscylatora sterowanego cyfrowo, współpraca z Aldec
45. Sprzętowa implementacja filtru różniczkującego na potrzeby radia programowalnego, współpraca z Aldec

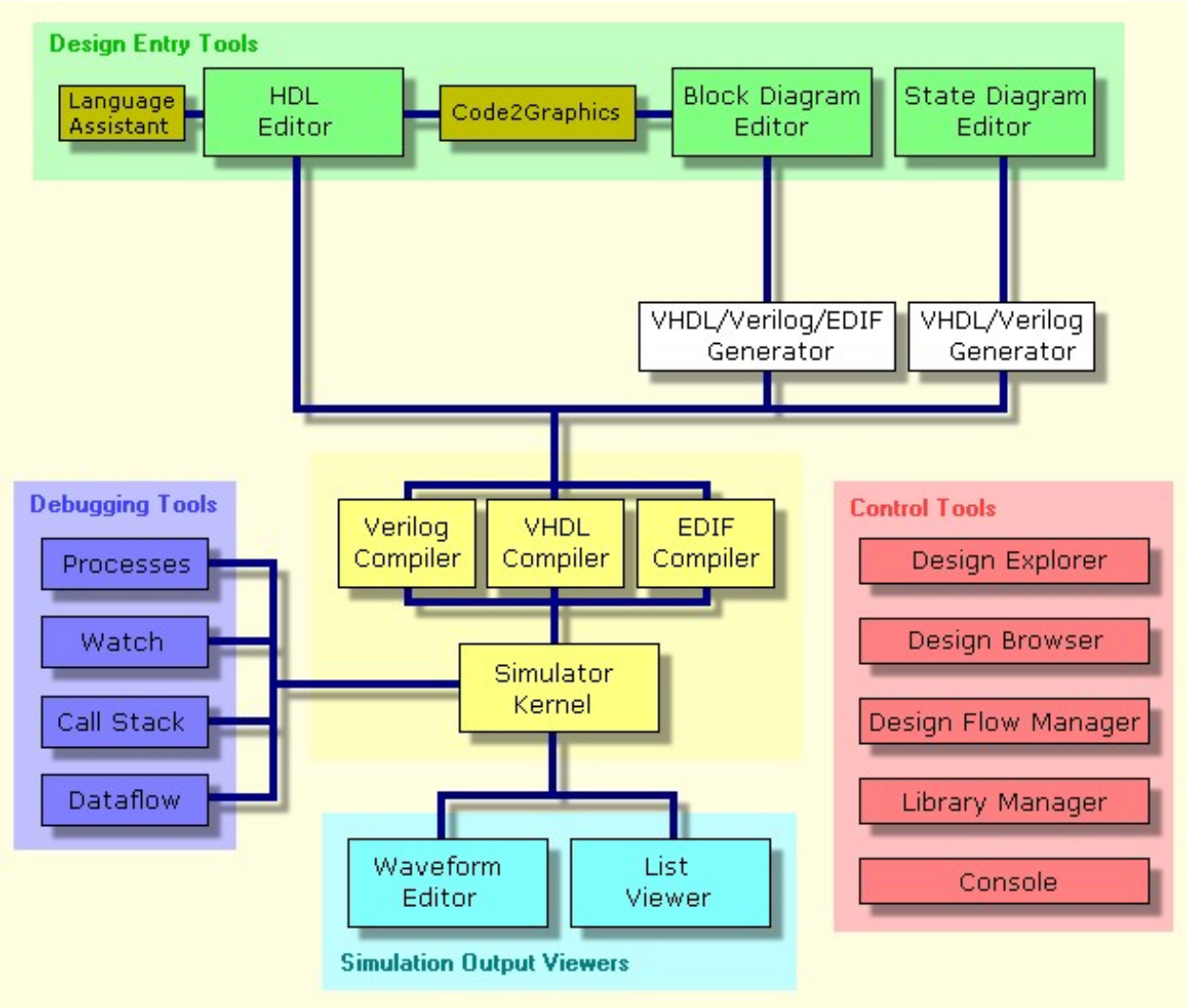
2020

44. Opracowanie i uruchomienie gry typu "Tetris" w układzie FPGA, współpraca z BTC Korporacja
43. Opracowanie i uruchomienie gry typu "Pong " w układzie FPGA, współpraca z BTC Korporacja
42. Podsystem obsługi wirtualnego interfejsu JTAG w układzie FPGA, współpraca z BTC Korporacja
41. Podsystem obsługi wyświetlacza dotykowego LCD w układzie FPGA, współpraca z BTC Korporacja
40. Podsystem tekstowego monitora HDMI w układzie FPGA, współpraca z BTC Korporacja

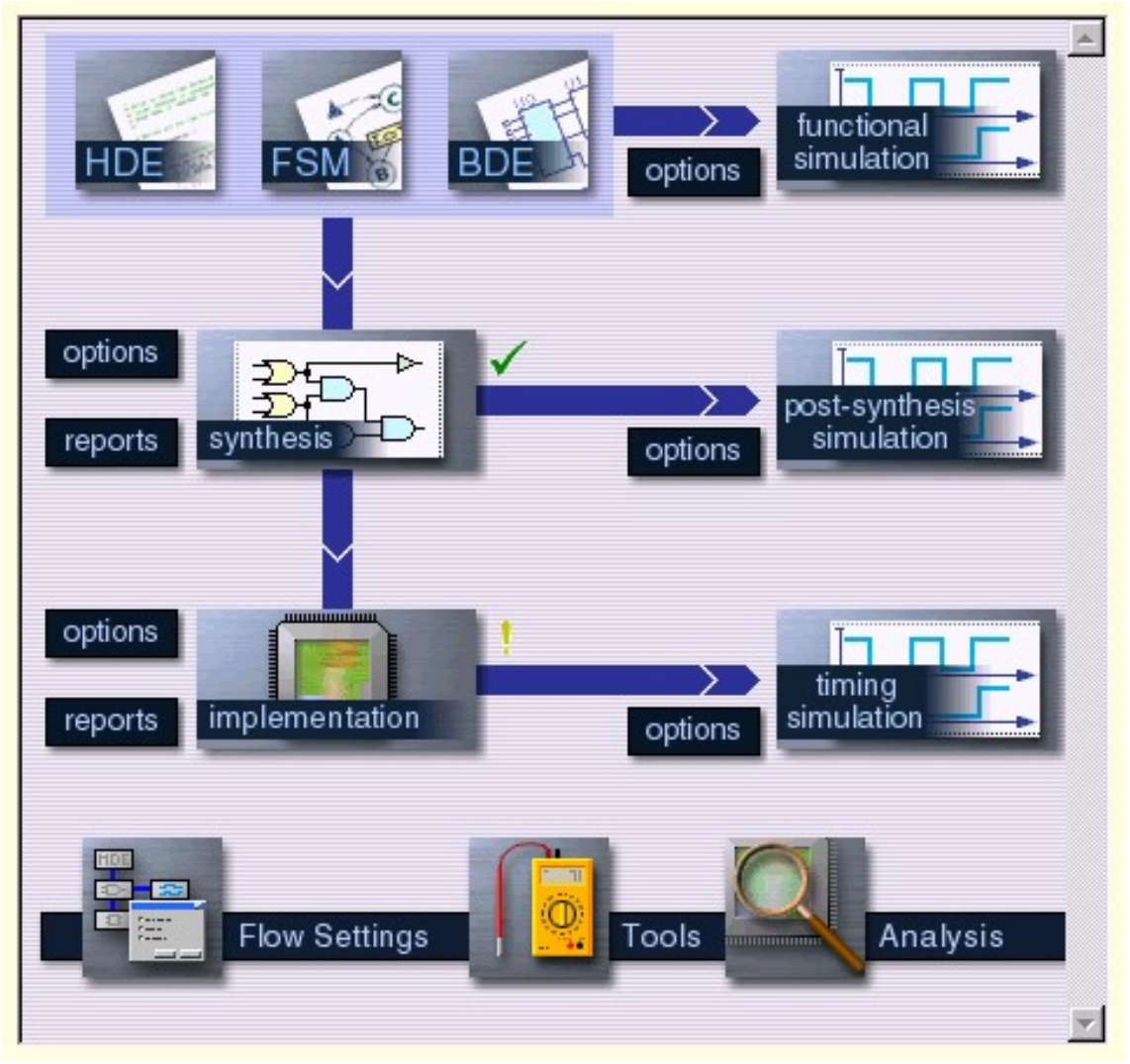
ActiveHDL – moduły



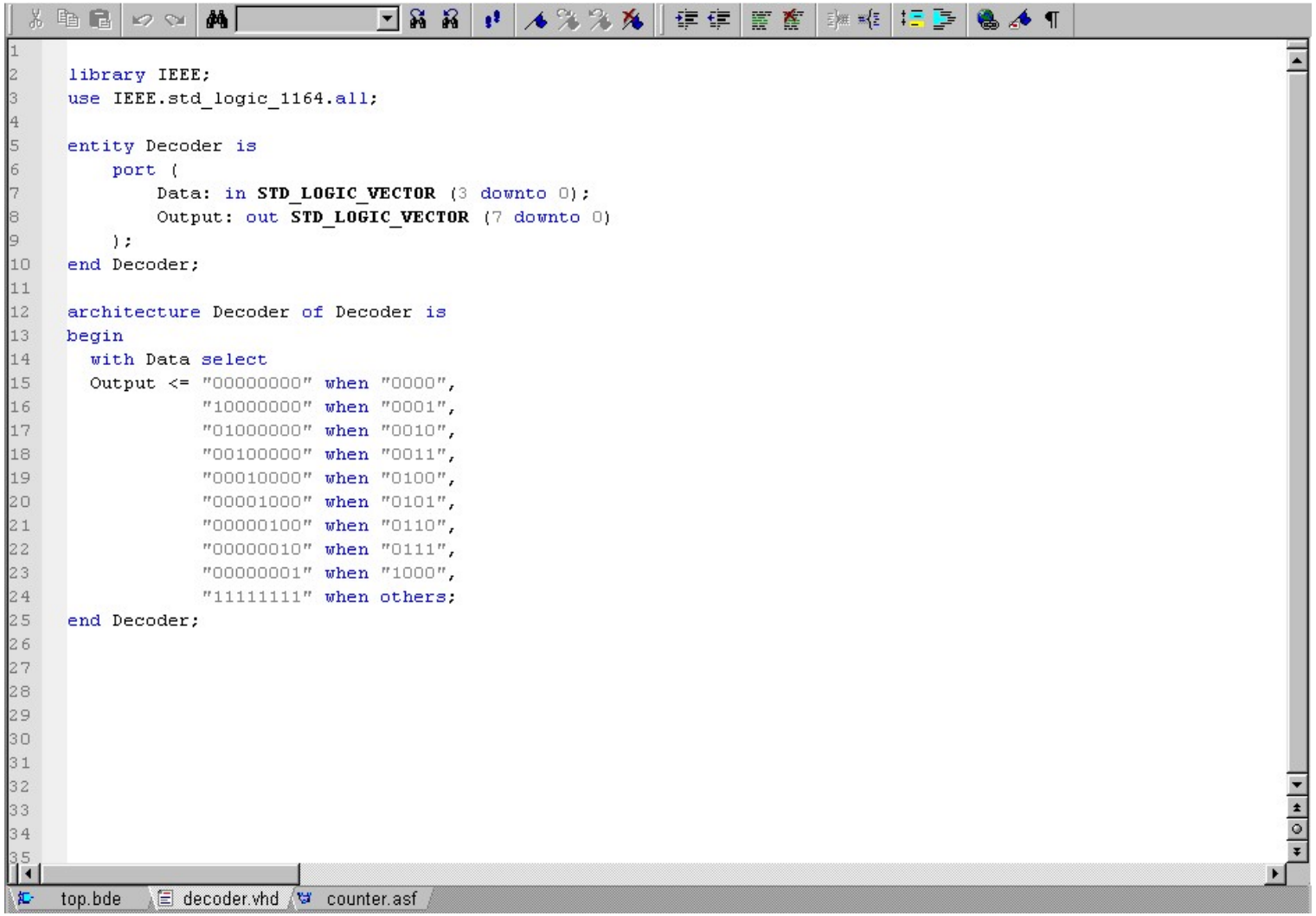
ActiveHDL – moduły



ActiveHDL – *Design Flow*



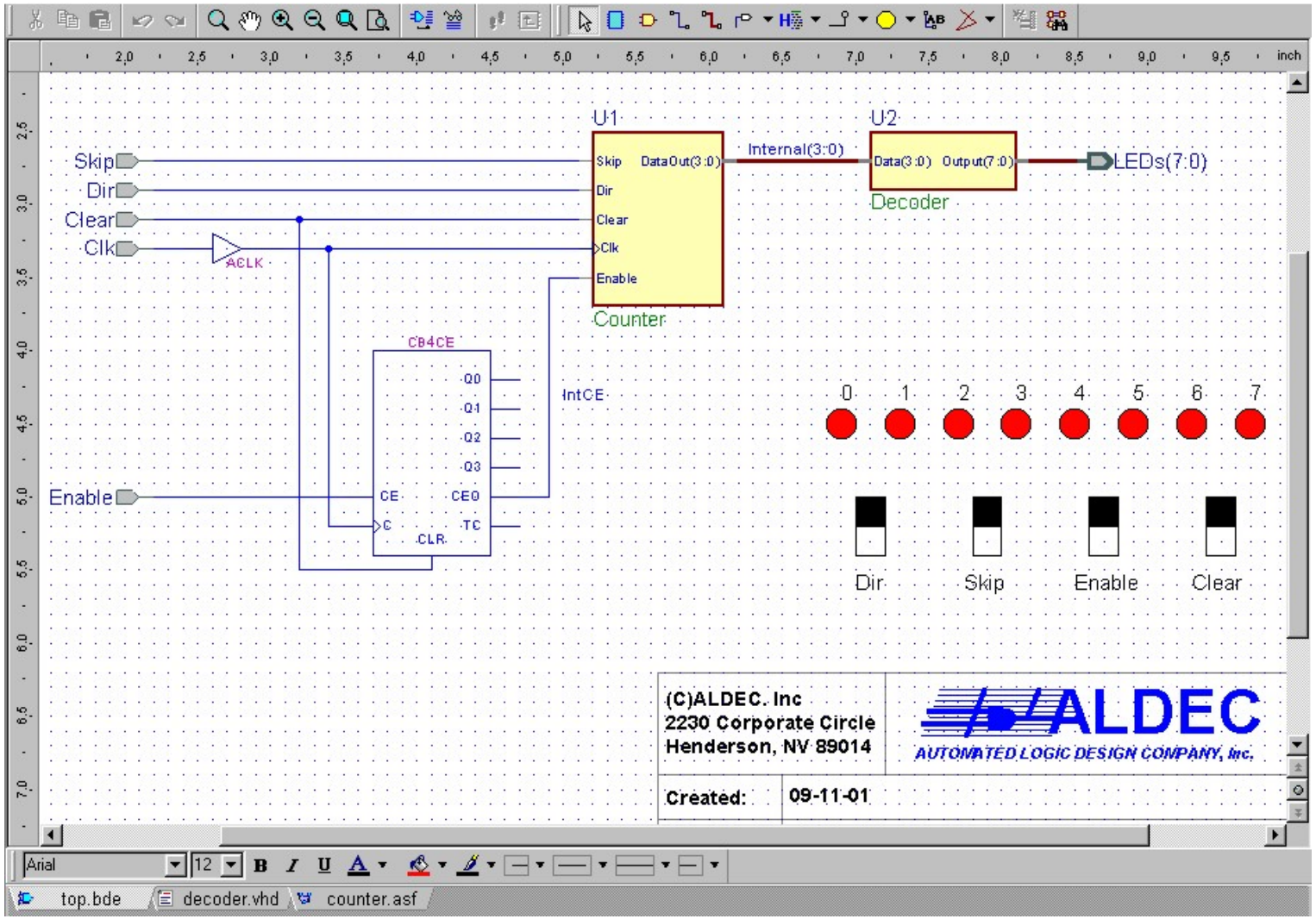
ActiveHDL – HDL



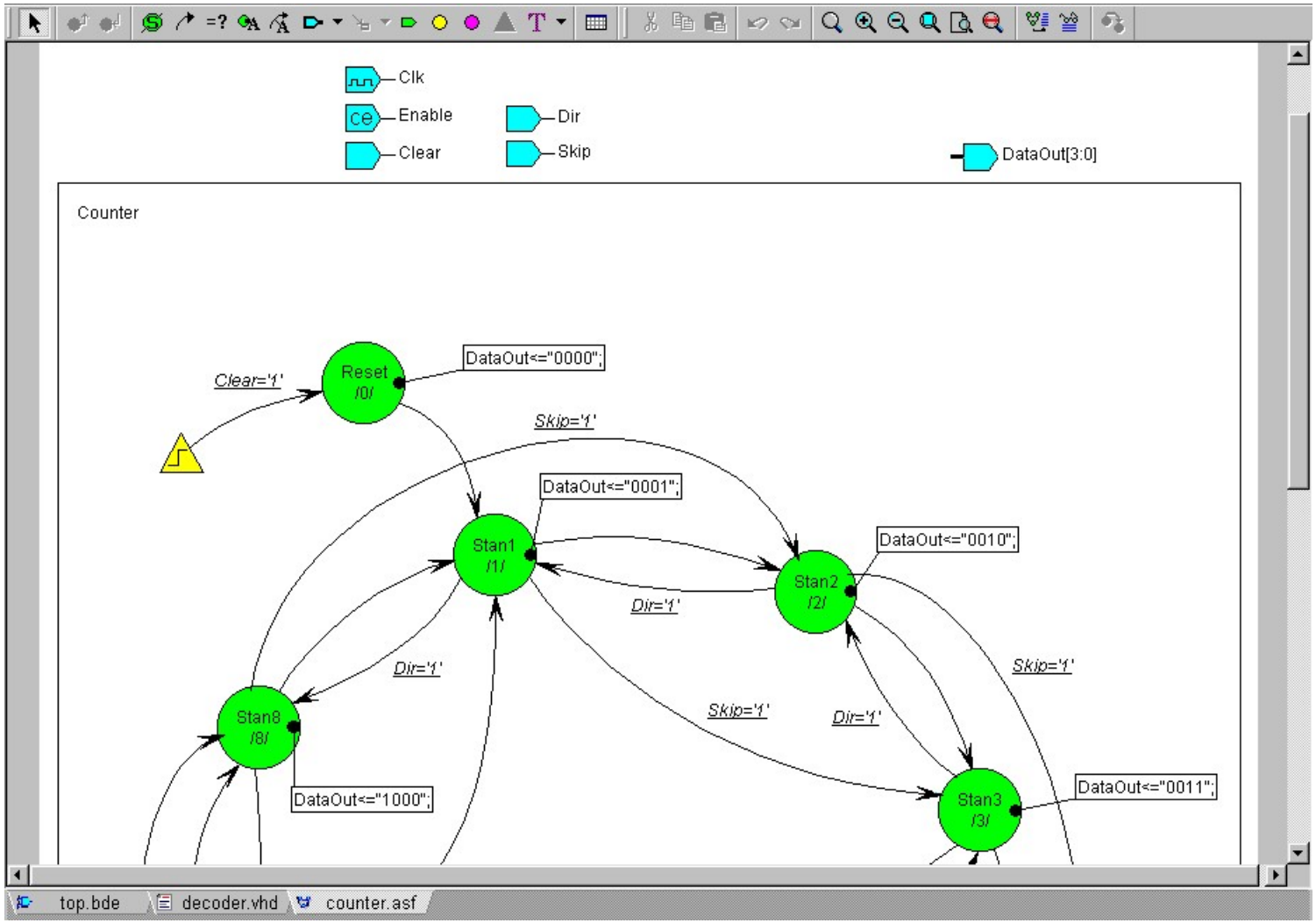
```
1
2   library IEEE;
3   use IEEE.std_logic_1164.all;
4
5   entity Decoder is
6       port (
7           Data: in STD_LOGIC_VECTOR (3 downto 0);
8           Output: out STD_LOGIC_VECTOR (7 downto 0)
9       );
10  end Decoder;
11
12  architecture Decoder of Decoder is
13  begin
14    with Data select
15    Output <= "00000000" when "0000",
16              "10000000" when "0001",
17              "01000000" when "0010",
18              "00100000" when "0011",
19              "00010000" when "0100",
20              "00001000" when "0101",
21              "00000100" when "0110",
22              "00000010" when "0111",
23              "00000001" when "1000",
24              "11111111" when others;
25  end Decoder;
26
27
28
29
30
31
32
33
34
35
```

top.bde | decoder.vhd | counter.asf

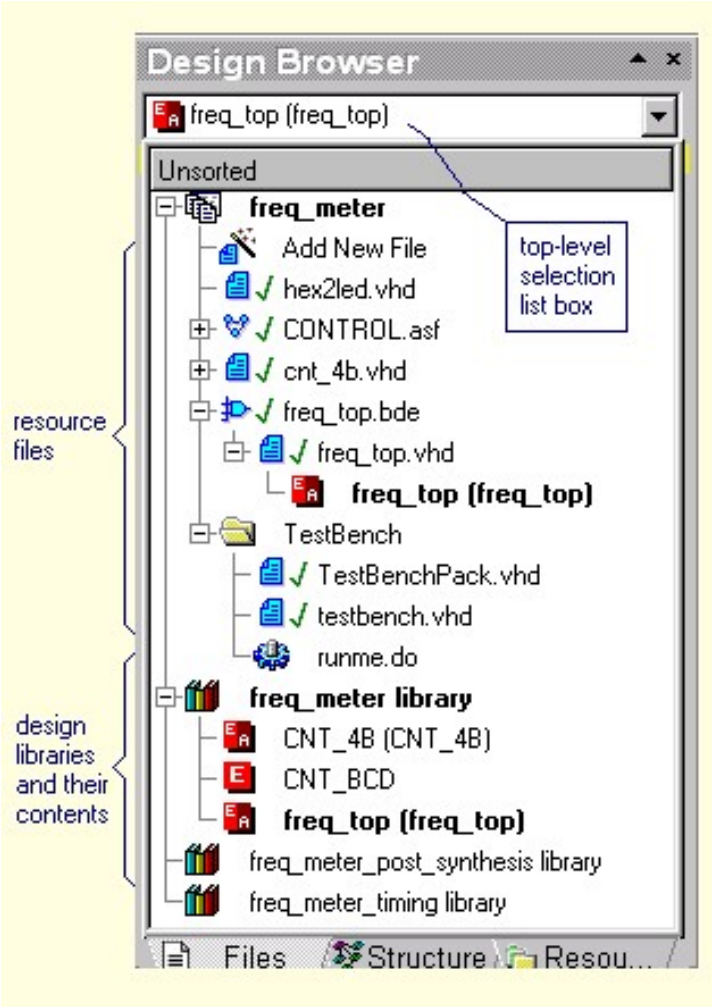
ActiveHDL – BDE

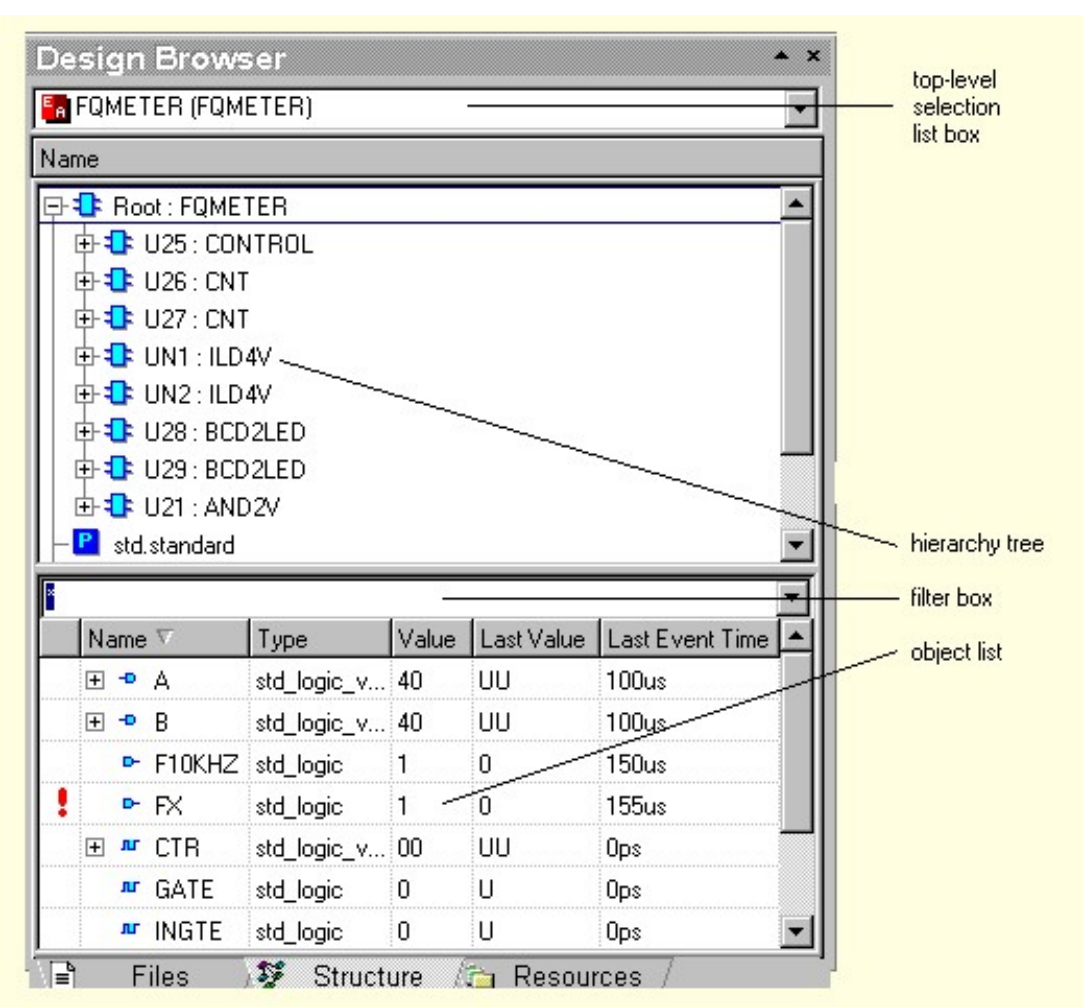


ActiveHDL – FSM

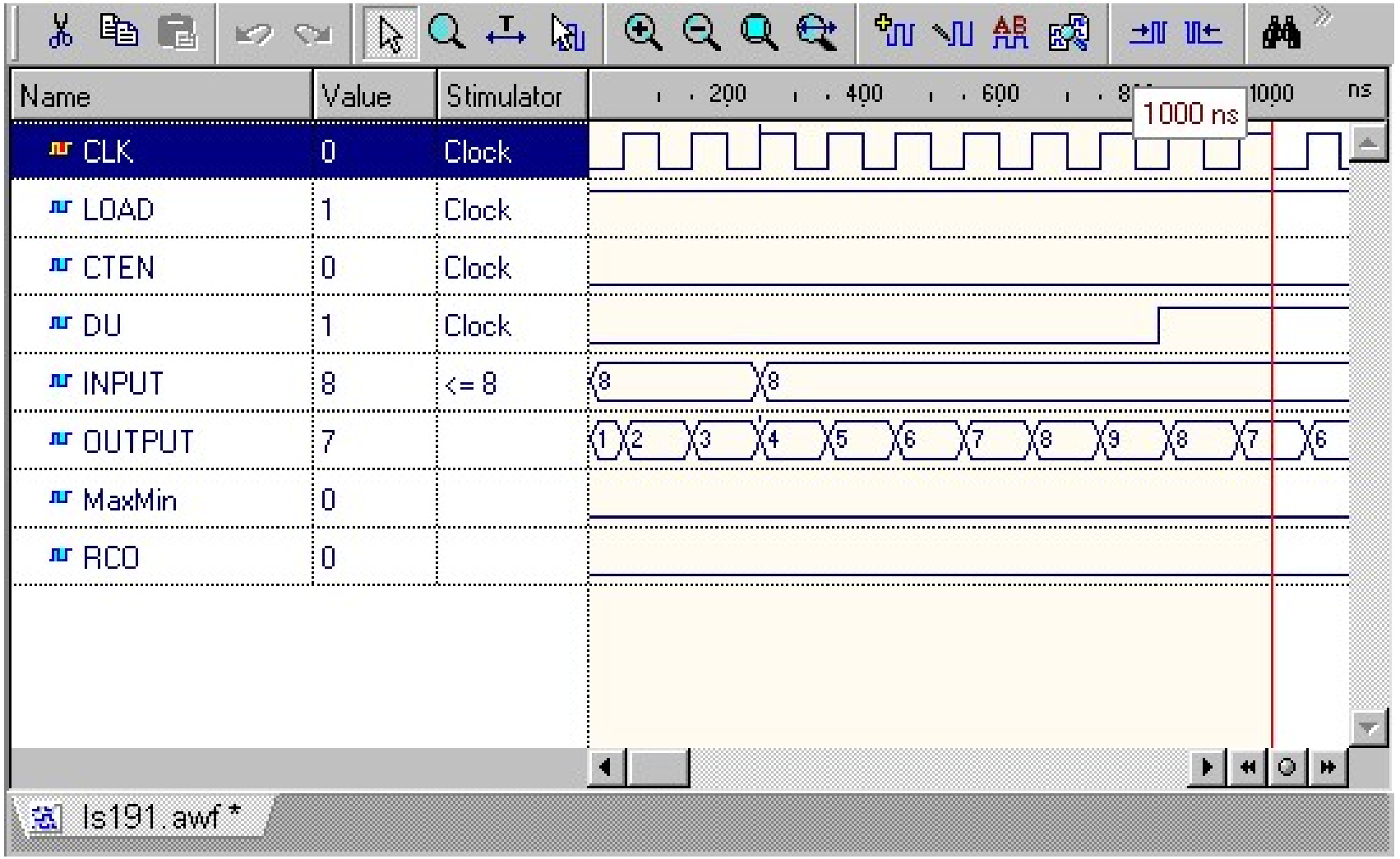


ActiveHDL – *Design Browser*

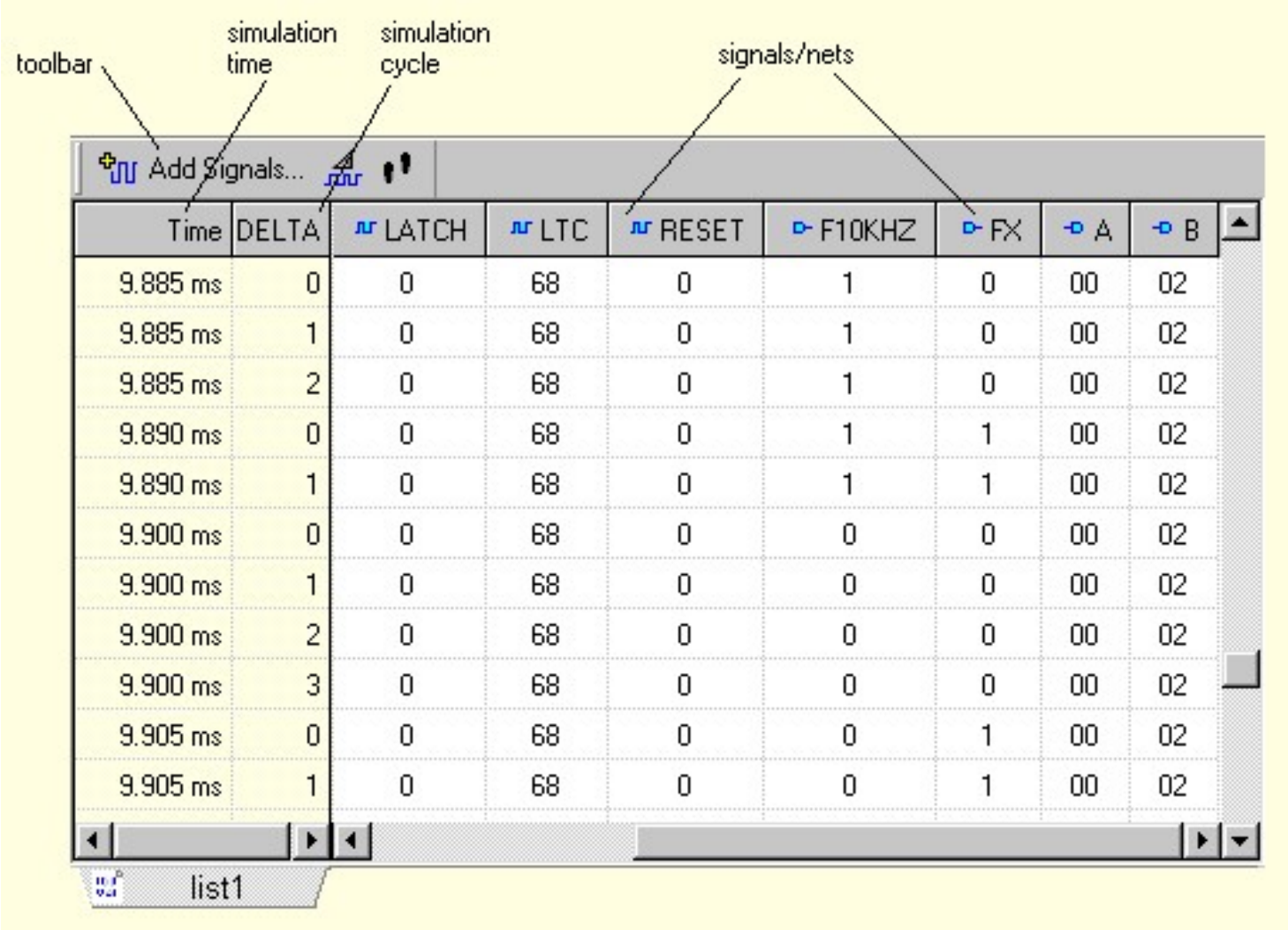




ActiveHDL – *Waveform Viewer*

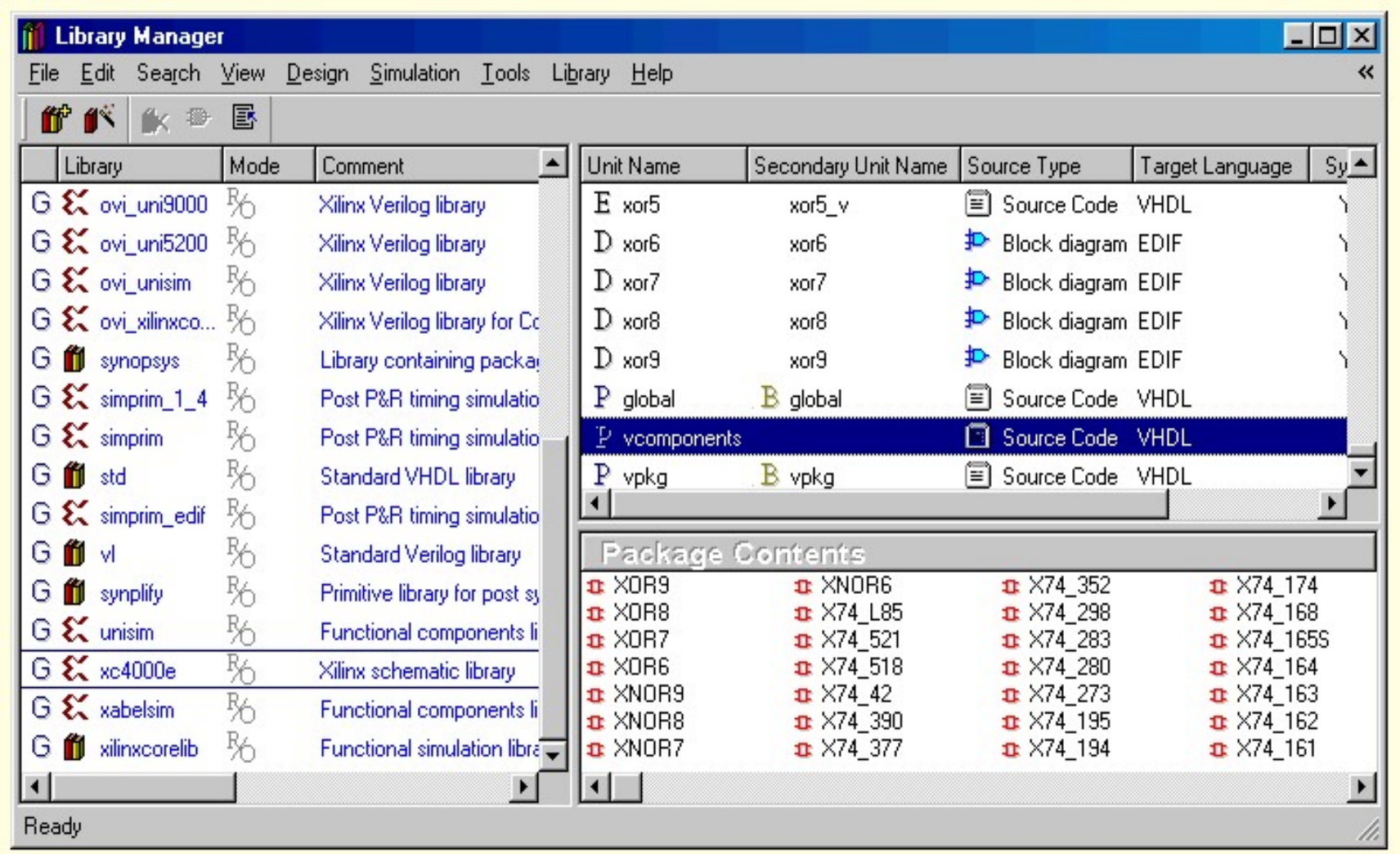


ActiveHDL – *List Viewer*



Time	DELTA	LATCH	LTC	RESET	F10KHZ	FX	A	B
9.885 ms	0	0	68	0	1	0	00	02
9.885 ms	1	0	68	0	1	0	00	02
9.885 ms	2	0	68	0	1	0	00	02
9.890 ms	0	0	68	0	1	1	00	02
9.890 ms	1	0	68	0	1	1	00	02
9.900 ms	0	0	68	0	0	0	00	02
9.900 ms	1	0	68	0	0	0	00	02
9.900 ms	2	0	68	0	0	0	00	02
9.900 ms	3	0	68	0	0	0	00	02
9.905 ms	0	0	68	0	0	1	00	02
9.905 ms	1	0	68	0	0	1	00	02

ActiveHDL – *Library Manager*



Ciąg dalszy nastąpi…