



# Wprowadzenie

# Kontakt z prowadzącym

dr inż. Paweł J. Rajda [pjrajda@agh.edu.pl](mailto:pjrajda@agh.edu.pl)

<http://www.embedded.agh.edu.pl>

C-3, p.502, tel. (12) 617 3980



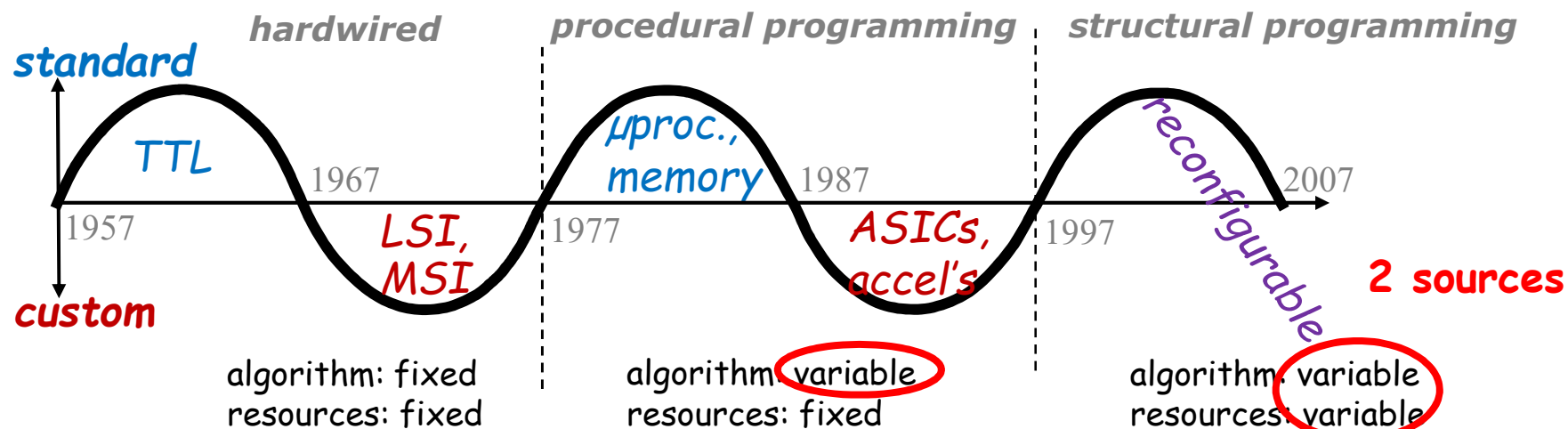
- 1. HDL – po co?**
- 2. HDL – co to jest?**
- 3. HDL – jak, gdzie, kiedy?**
- 4. HDL – i co z tego?**
- 5. HDL – wczoraj, dziś i jutro**
- 6. HDL – standardy**
- 7. HDL – literatura**
- 8. HDL – źródła**
- 9. HDL – firmy**
- 10. HDL – Aldec: Active-HDL**



# Makimoto's Wave Technologiczne wahadło



Reiner Hartenstein  
Model systemu



Single chip  $\mu$ C circa 1974

TI TMS 1000  
Fairchild F8  
Intel 8048  
Mostek 3870  
etc.

40 pins  
10,000 gates  
10,000 RAM bits  
1MHz clock

Single chip FPGA circa 2004

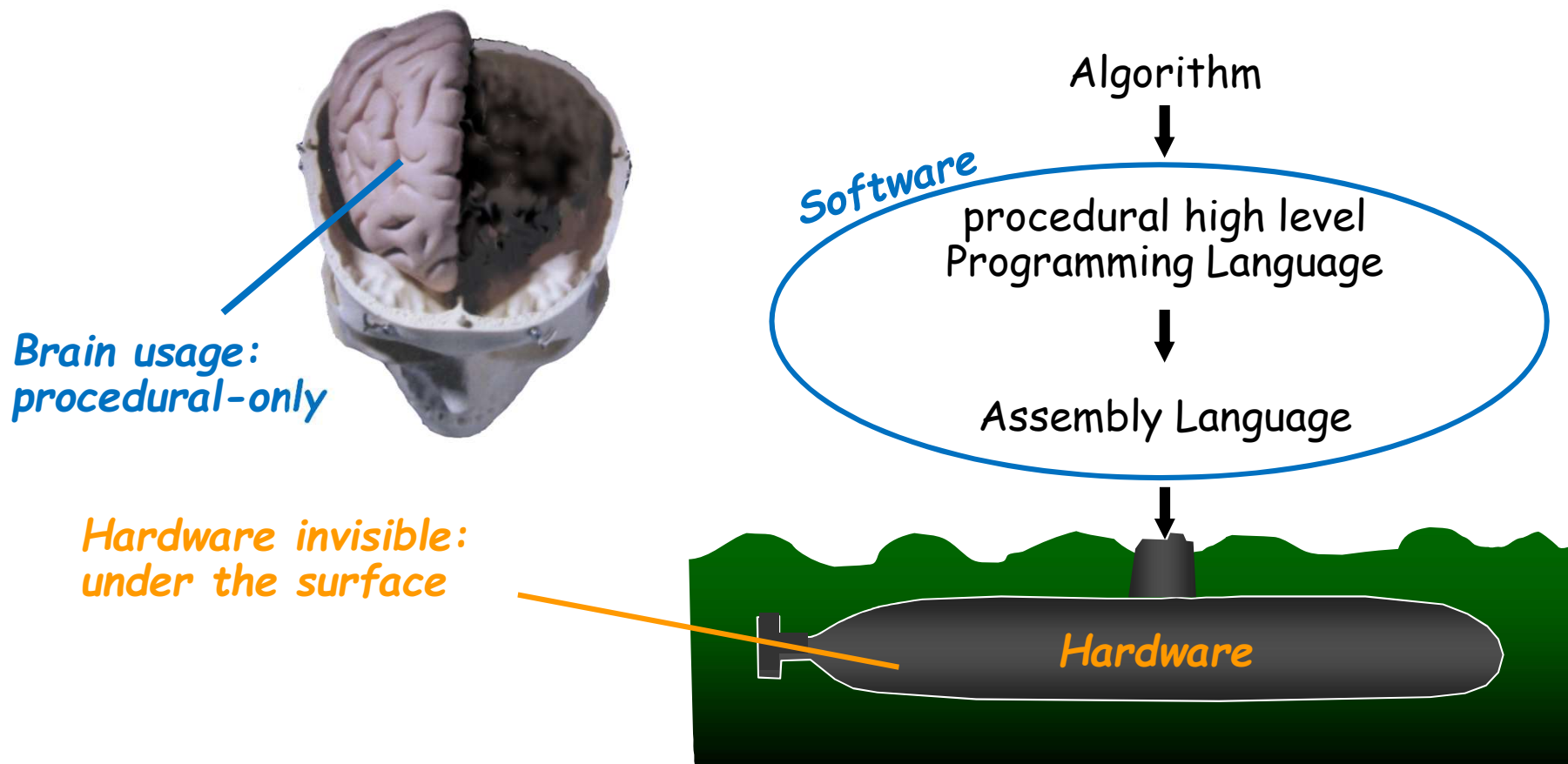
Xilinx  
Altera  
Actel  
etc.

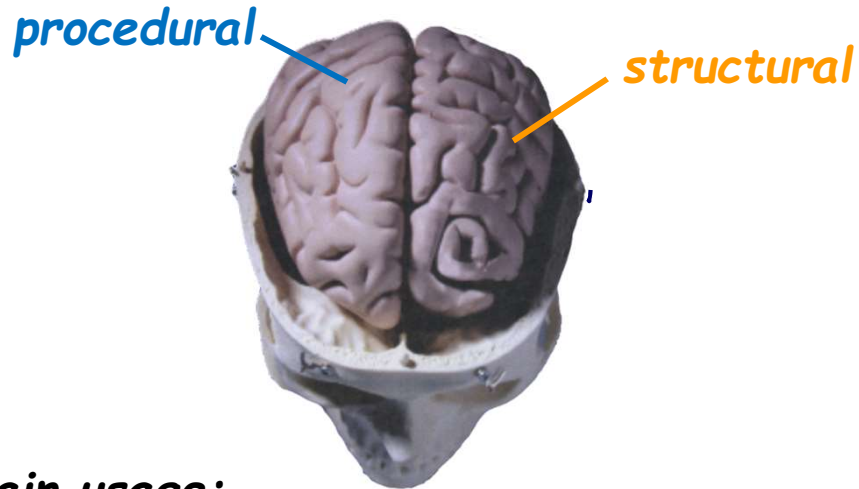
200+ pins  
10,000,000 gates  
10,000,000 RAM bits  
100MHz clock

vN machine paradigm

new machine paradigm needed

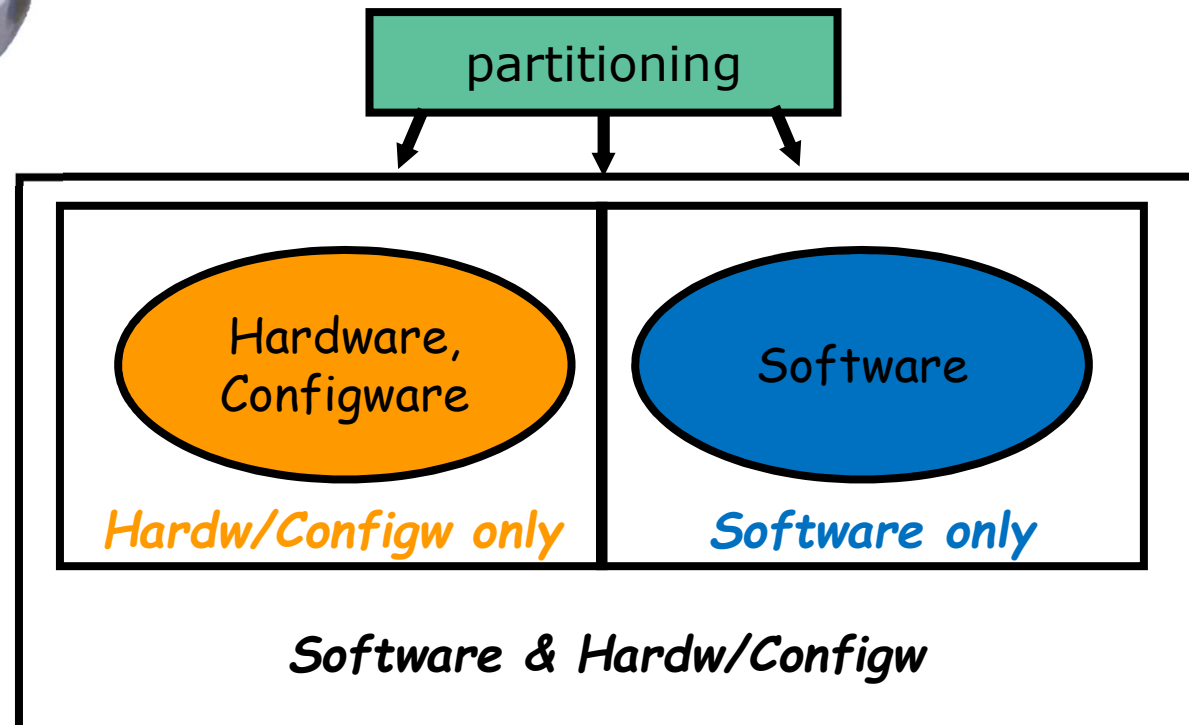
## The Programmable System-on-a-Chip IS THE NEXT WAVE



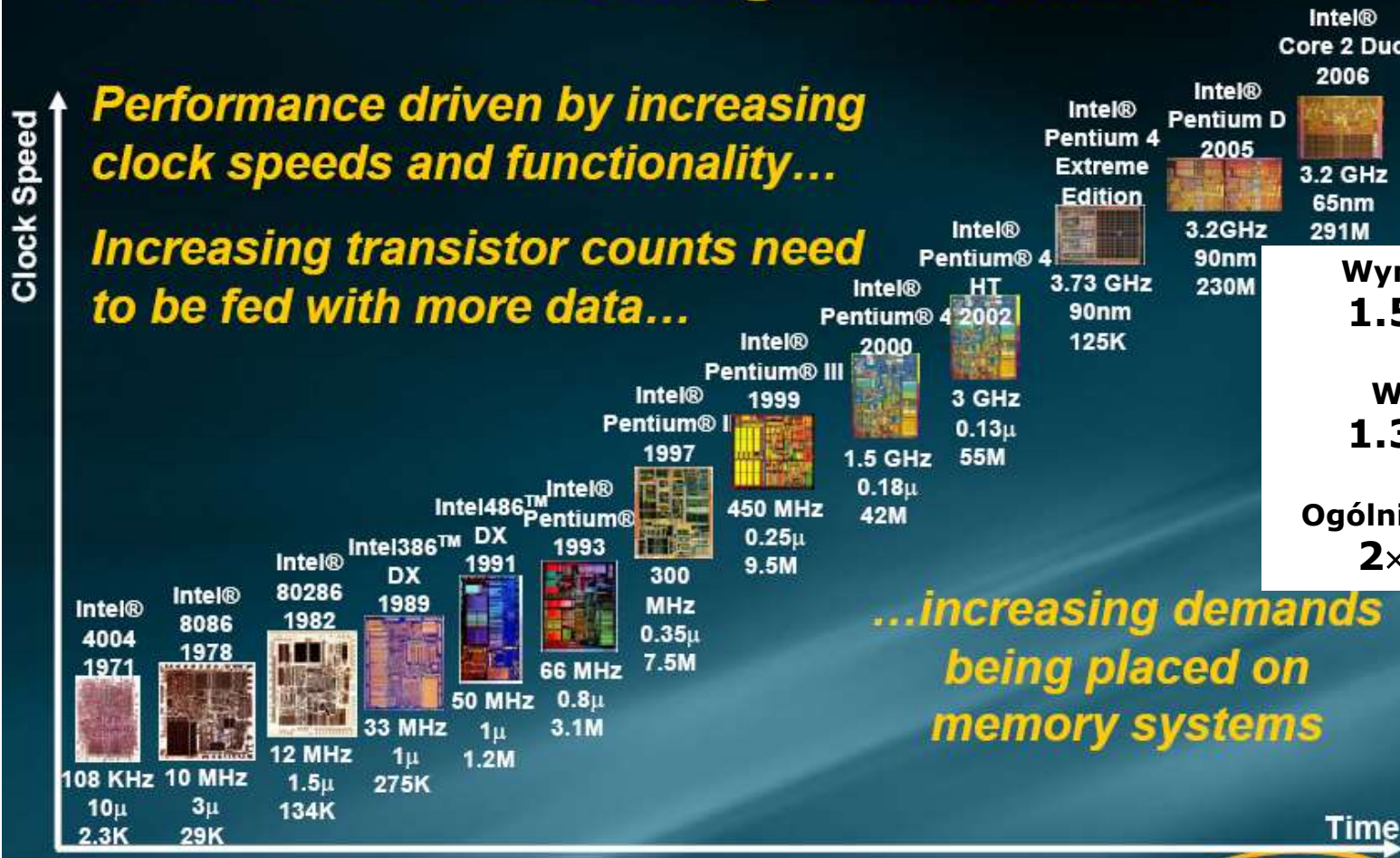


*Brain usage:  
both hemispheres*

# Algorithm



# Moore's Law Driving Performance



Wymogi pamięci:  
**1.50× na rok**

Wymogi CPU:  
**1.35× na rok**

Ogólnie prawo Moora:  
**2× co 2 lata**

## 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*

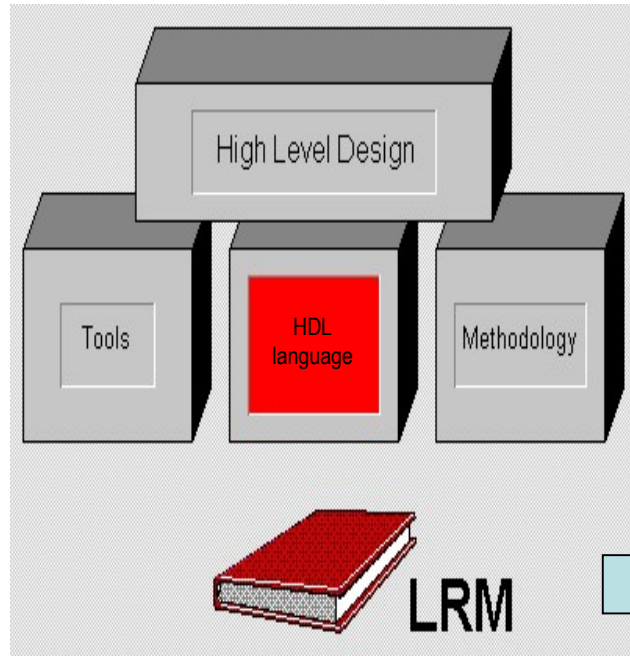


Rozwiązanie:  
wykorzystanie języka HDL  
(VHDL, Verilog, ...)



- **PALASM**
- **ABEL**
- **CUPL**
  
- **VHDL**
- **VERILOG, System VERILOG**
- **C, C++, Handel-C, System-C**
- **inne**

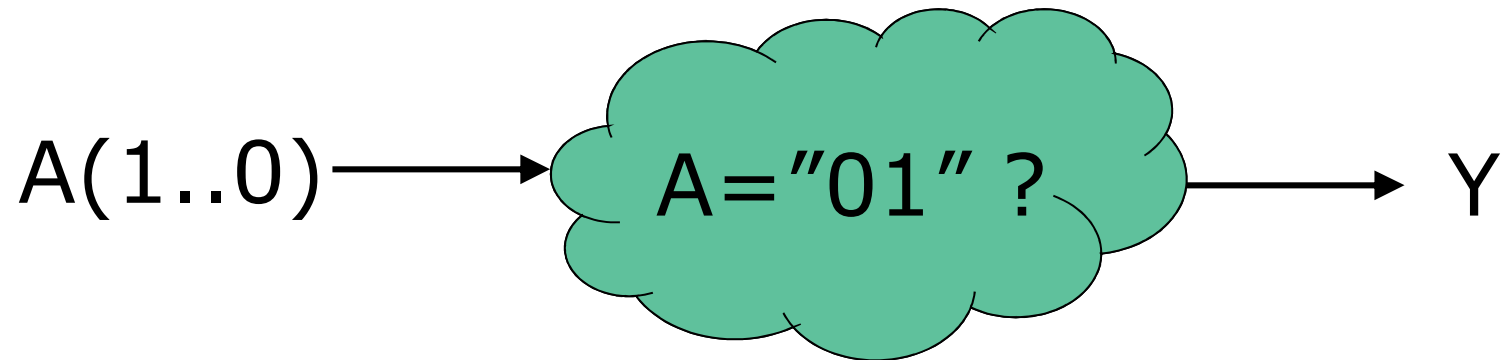
## HDL – co to jest? Definicja



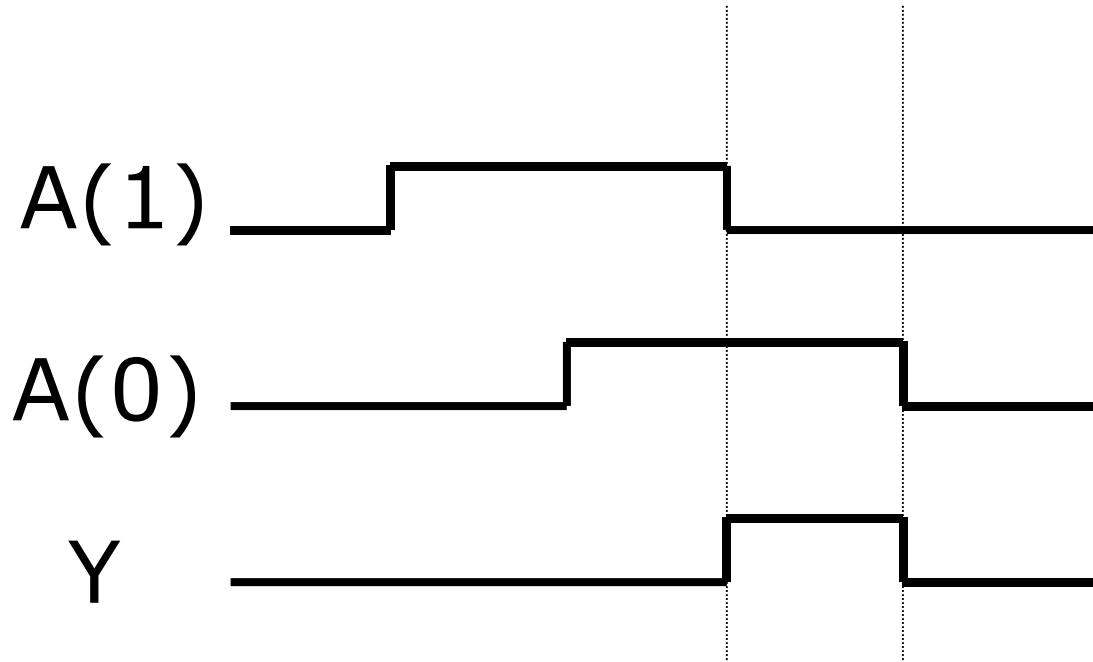
*It is "a formal notation intended for use in all phases of the creation of electronic systems. (...) it supports the development, verification, synthesis, and testing of hardware designs..."*

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

**VHDL** - **V**HSIC **H**ardware **D**escription **L**anguage  
↳ **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit  
**Verilog** - **V**erification **L**ogic

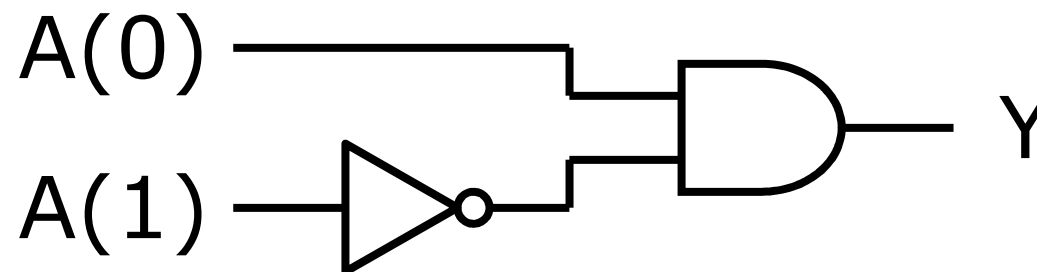


```
assign Y = (A == 2'b10) ? 1'b1 : 1'b0;
```



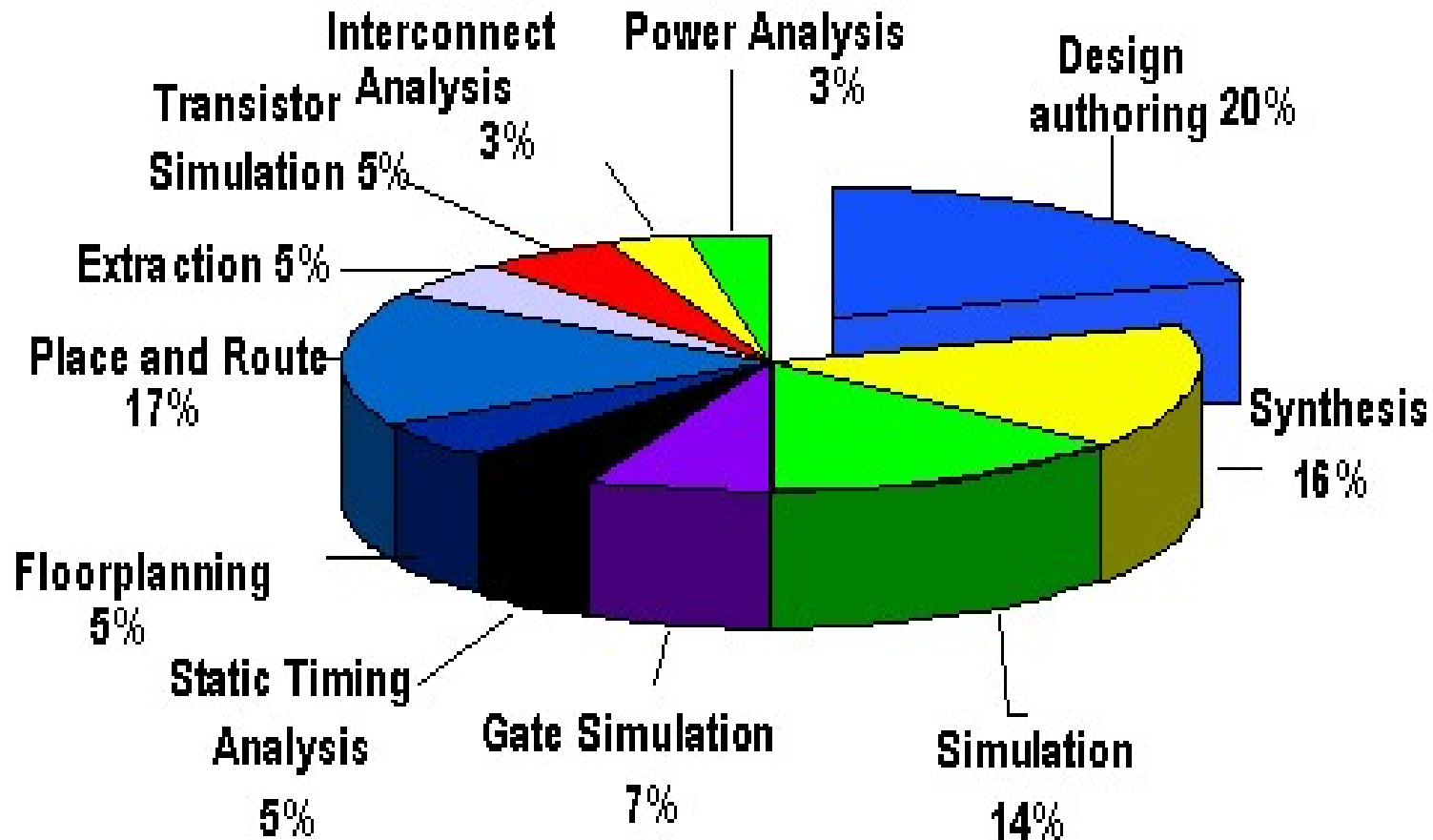
```
A = 2'b10;  
#5 A = 2'b11;  
#5 A = 2'b01;  
...
```

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



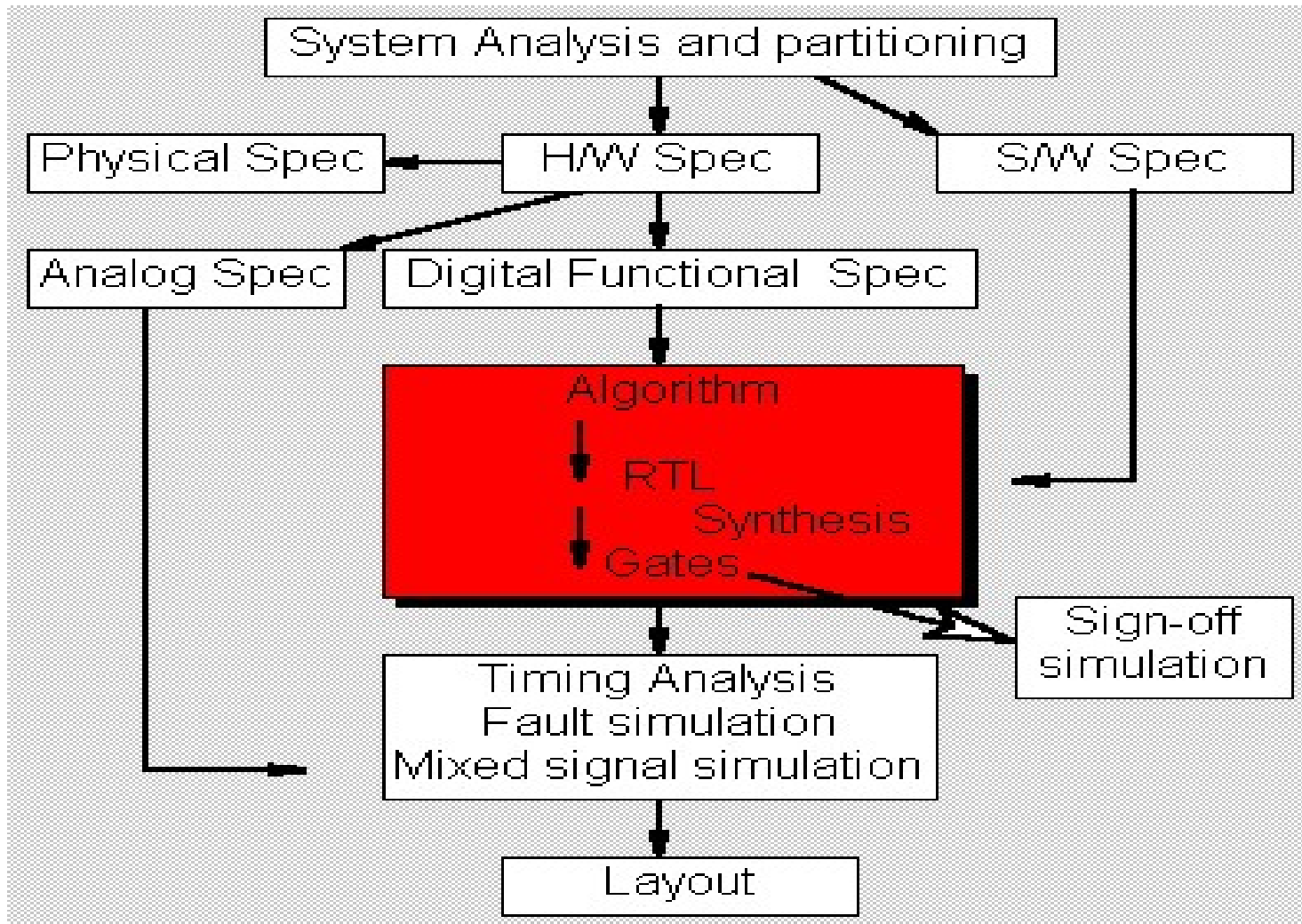
# HDL – jak, gdzie, kiedy?

## Etapy projektu

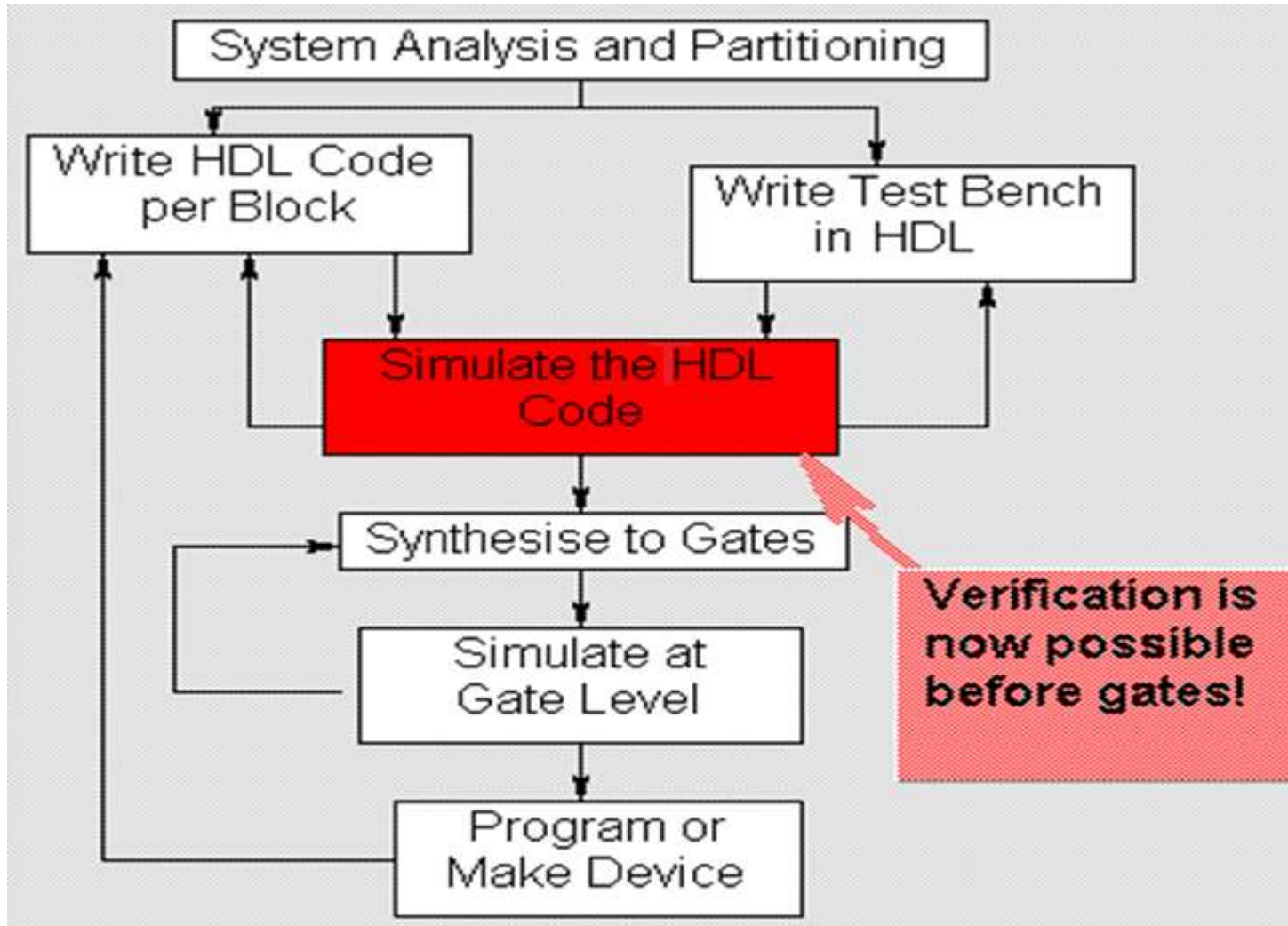


Average iterations between design and layout = 20

# HDL – jak, gdzie, kiedy? Projektowanie systemu



# HDL – jak, gdzie, kiedy? Proces projektowania





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
```

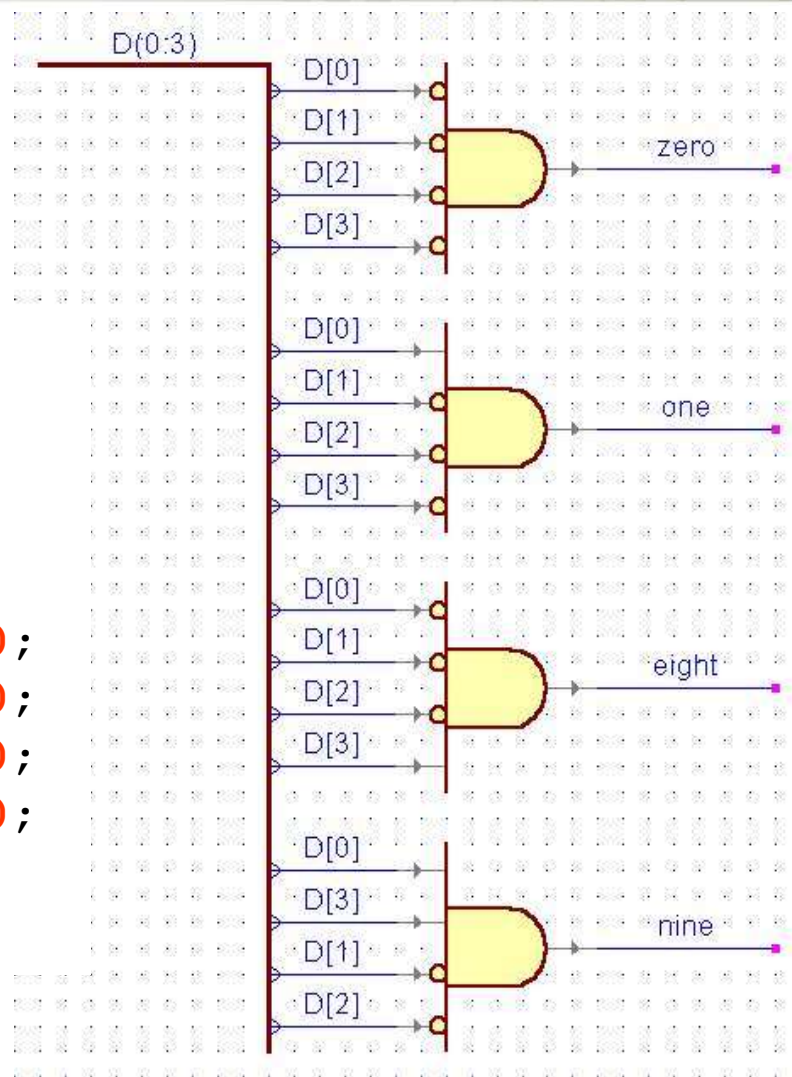
```

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

```



```

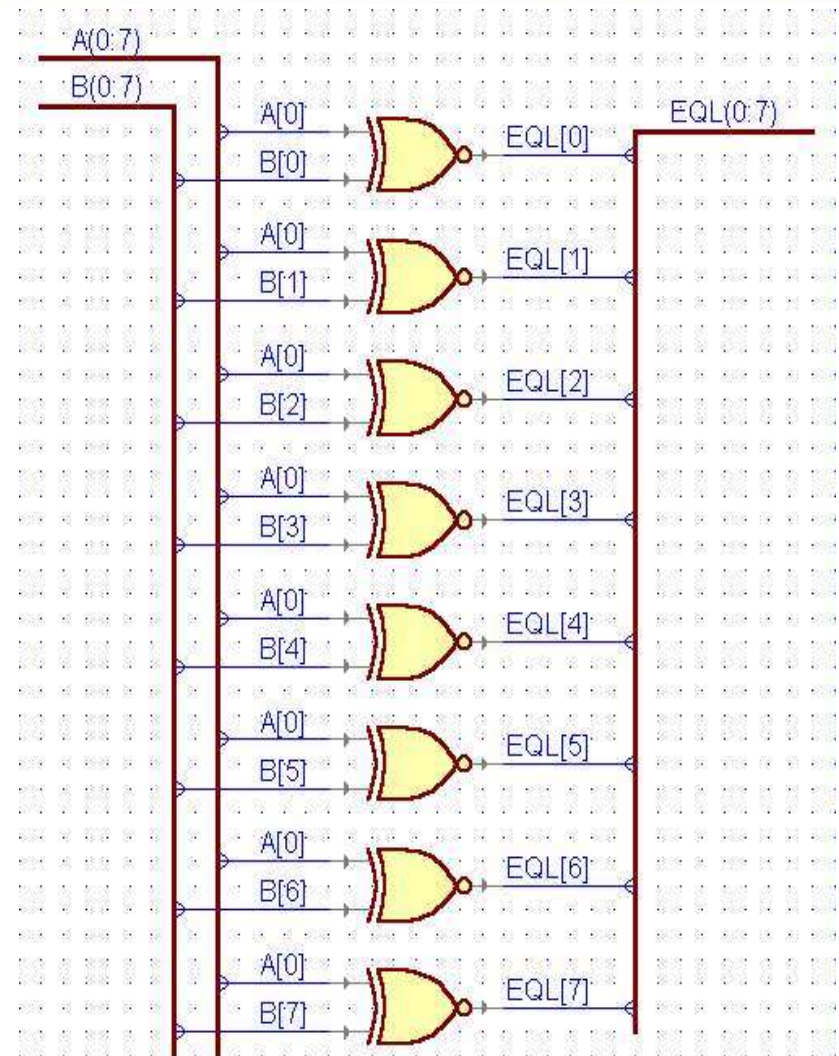
module COMPARE
  (A, B, EQL);
  input  [7:0] A, B;
  output [7:0] EQL;

  assign EQL = ~(A | B);

endmodule

```

**Błąd ! →**



```

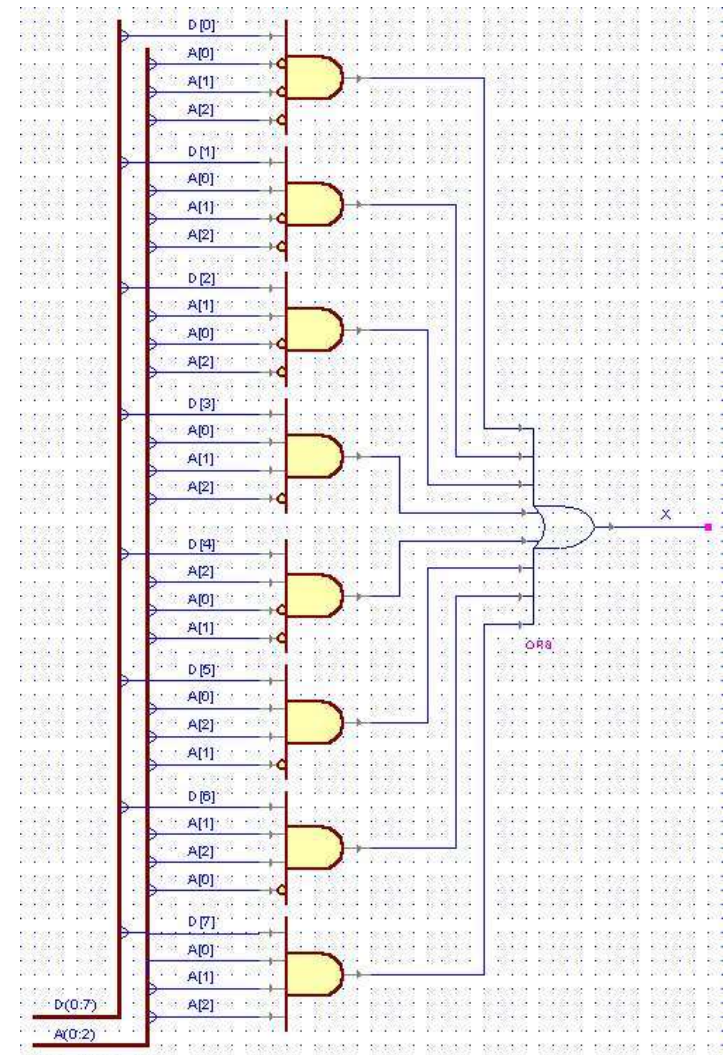
module MPLEXER
  (D, A, X);
  input [7:0] D;
  input A;
  output X;

  int A;

  assign X = D[A];

endmodule

```



```

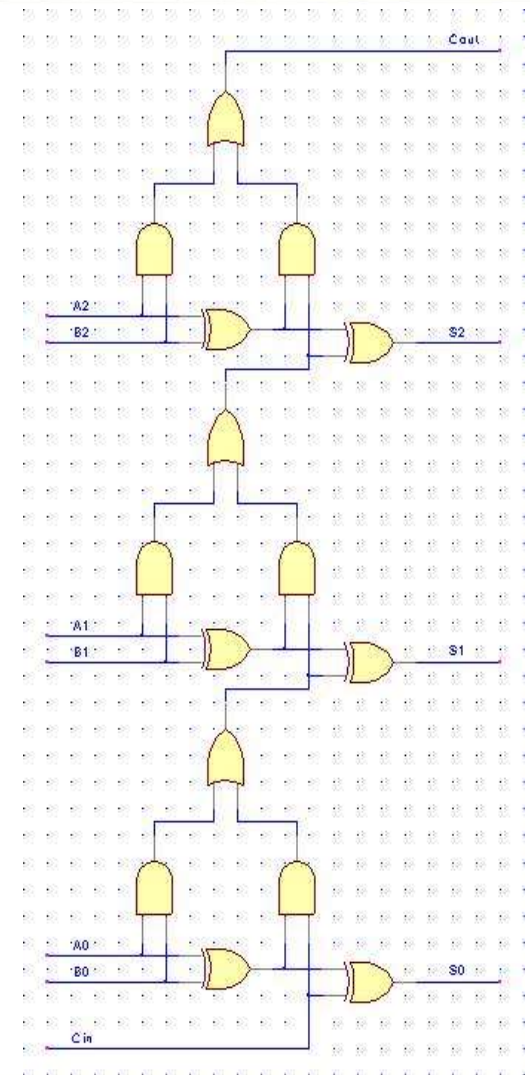
module SUM
  (A, B, S, Cin, Cout);
  input  [2:0] A, B;
  output [2:0] S;
  input  Cin;
  output Cout;

  wire [3:0] V;

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

endmodule

```



```

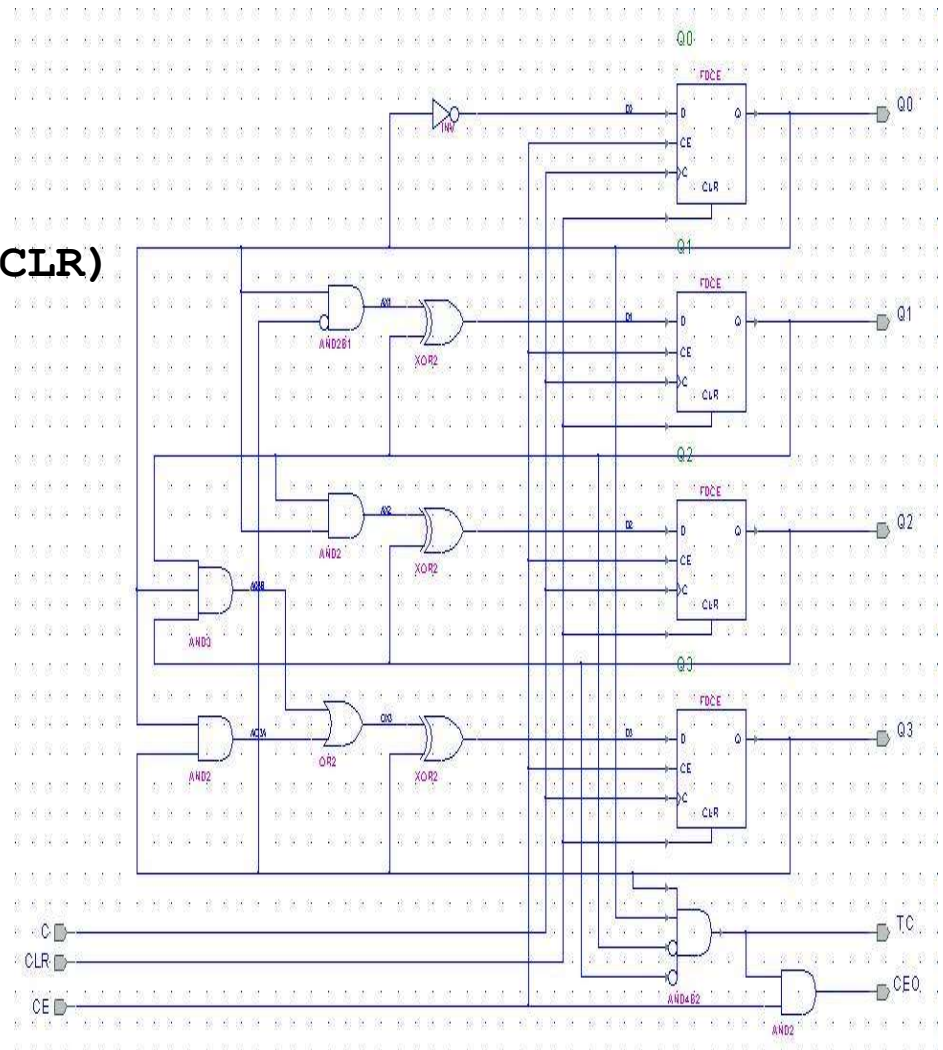
module COUNT
  (C, CE, CLR, Q, TC, CEO);
  (...)

  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

```



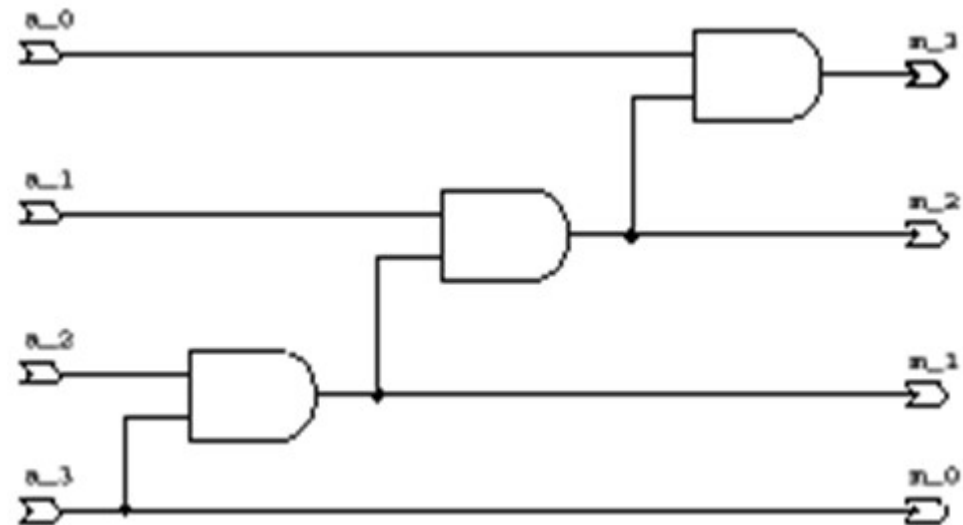
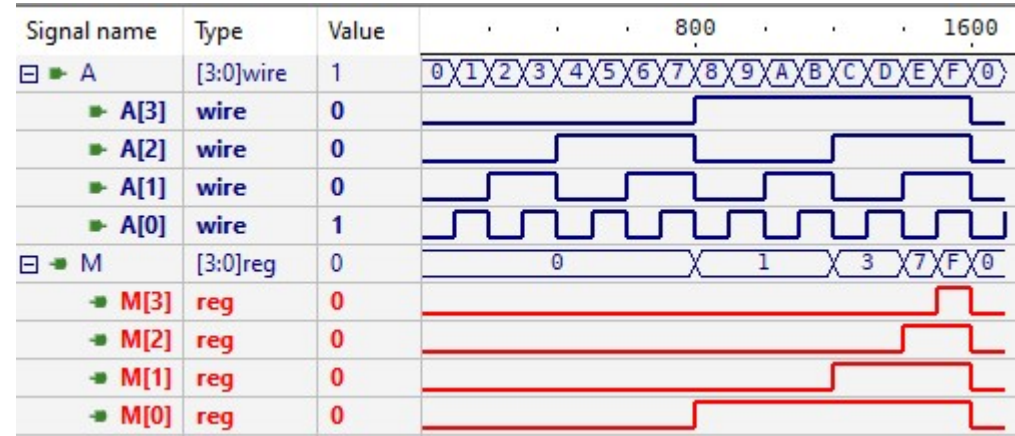


# VERILOG – przykłady Powielanie logiki

```
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
```



- **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**





## HDL – i co z tego? Porównanie

### VHDL

Uważa się go za trudniejszy do nauczenia. Złożona składnia, bazuje na językach Ada i Pascal. Rzadziej daje mniej mylne wrażenie co do istoty projektu.

Kod jest nieco bardziej rozwlekły, jednak lepiej się samo-tłumaczy - wiele elementów składni jest słowami.

Dla projektów *reliability* (lotnictwo, kosmos) używany wyłącznie.

Wymaga więcej wysiłku dla uruchomienia wstępnej weryfikacji.

Oferuje liczne predefiniowane typy danych i restryktywną politykę weryfikacji. Błędy związane z typami danych rzadko mogą nie zostać wykryte na wstępnych etapach obróbki kodu.

### VERILOG

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 zwięzły, jednak gorzej się samo-tłumaczy - wiele elementów składni jest symbolami.

Szeroko używany w przemyśle *consumer*.

Pozwala na szybką i łatwą wstępną weryfikację.

Oferuje nieliczne predefiniowane typy danych i liberalną politykę weryfikacji. Błędy związane z typami danych często mogą nie zostać wykryte na wstępnych etapach obróbki kodu.



## HDL – i co z tego? Porównanie

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	<b>wydany opis IEEE Std 1076</b>	
1990		przejęcie przez Cadence, założenie OVI
1983	nowelizacja IEEE Std 1076-1993	
1995		<b>wydany opis IEEE Std 1364</b>
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 Standardization Group (VASG)

<http://www.eda-twiki.org/cgi-bin/view.cgi/P1076/WebHome>



P1076

Log In or Register

P1076 Web

- Create New Topic
- Index
- Search
- Changes
- Notifications
- RSS Feed
- Statistics
- Preferences

Webs

- Main
- P1076
- P10761
- P1647
- P16661
- P1685
- P1734
- P1735
- P1778
- P1800
- P1801
- Sandbox
- TWiki
- VIP
- VerilogAMS

TWiki > P1076 Web > WebHome (2016-01-28, StanKrolkoski) Edit Attach

### IEEE P1076 Working Group

#### VHDL Analysis and Standardization Group (VASG)

#### Mission

VASG is responsible for maintaining and extending the VHDL standard (IEEE 1076).

#### Status

VASG is actively working on proposals for P1076 201X. For more information see [Working Group Status](#)

#### Participating

If you are an experienced VHDL user, digital designer, or verification engineer, then this is your working group. Here you can contribute to the future of VHDL.

P1076 is an individual based standard. The only requirement for membership is to show up and participate. Work is done via on this TWIKI site, and the email reflector, in our phone meetings. Opportunities to participate are available at many different levels of commitment and your participation will help. Join us.

- To participate in TWIKI, send email to TWIKI admin: [Jim Lewis](#)
- Email Reflector: \* [View](#) \* [subscribe](#) \* [unsubscribe](#) \* [Send Email \(subscribe first\)](#)

#### Meetings & Work Areas



# Verilog – dziś

## IEEE P1800 Working Group

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



**P1800** TWiki > P1800 Web > WebHome (2016-03-29, MattMaidment) [Edit](#) [Attach](#)

[Log In or Register](#)

**P1800 Web**

- [Create New Topic](#)
- [Index](#)
- [Search](#)
- [Changes](#)
- [Notifications](#)
- [Statistics](#)
- [Preferences](#)

**Webs**

- [Main](#)
- [P1076](#)
- [Ballots](#)
  - [LCS2016\\_080](#)
- [P10761](#)
- [P1647](#)
- [P16661](#)
- [P1685](#)
- [P1734](#)
- [P1735](#)
- [P1778](#)
- [P1800](#)
- [P1801](#)
- [Sandbox](#)
- [TWiki](#)
- [VIP](#)
- [VerilogAMS](#)

### Welcome to the P1800 web

#### Available Information

- [P1800Workinggroup](#) – Meeting Minutes, etc.
- [SystemVerilogAssertionCommittee](#): The SVA subcommittee.
- [SystemVerilogBasicCommittee](#): The SV-BC subcommittee. Currently combined with SV-EC to address testbench and basic design issues.
- [SystemVerilogSpecialCommittee](#): The "Special Committee" designated to work on checkers and related constructs, from 4/08-7/08.
- [SystemVerilogInterfacesCommittee](#): The SV-CC subcommittee - sub committee focused on the APIs to SystemVerilog.
- [SystemVerilogDiscreteCommittee](#): The SV-DC subcommittee - sub committee focused on analog to digital connections.

#### P1800 Web Utilities

- [Search](#) - [advanced search](#)
- [WebTopicList](#) - all topics in alphabetical order
- [WebChanges](#) - recent topic changes in this web
- [WebNotify](#) - subscribe to an e-mail alert sent when topics change
- [WebRss](#), [WebAtom](#) - RSS and ATOM news feeds of topic changes
- [WebStatistics](#) - listing popular topics and top contributors
- [WebPreferences](#) - preferences of this web

- 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 Multivalued 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 and 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*



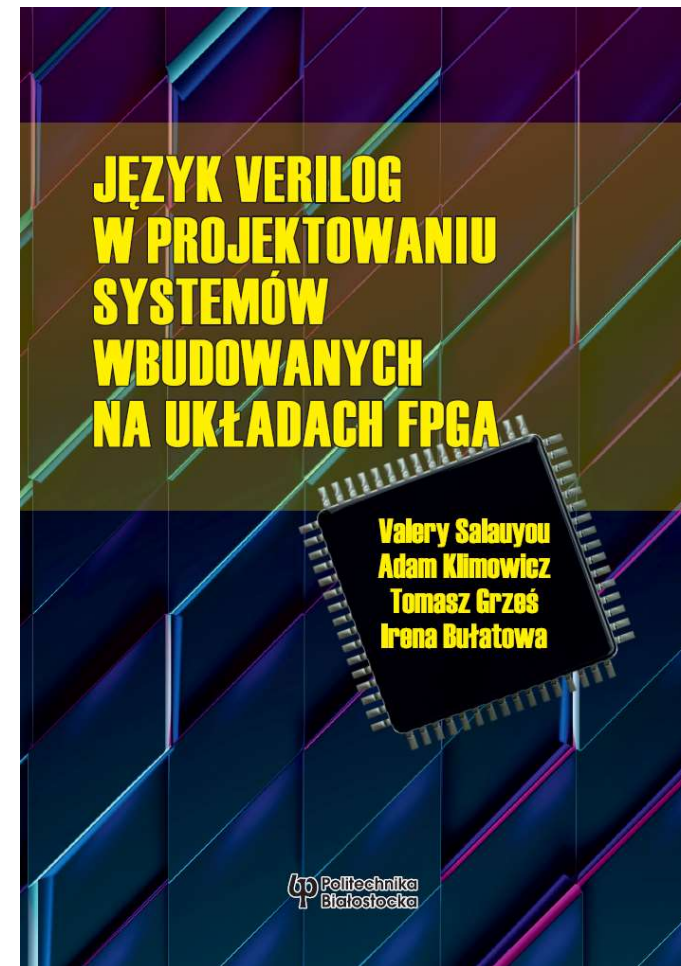


## Verilog – literatura

- *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 – 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

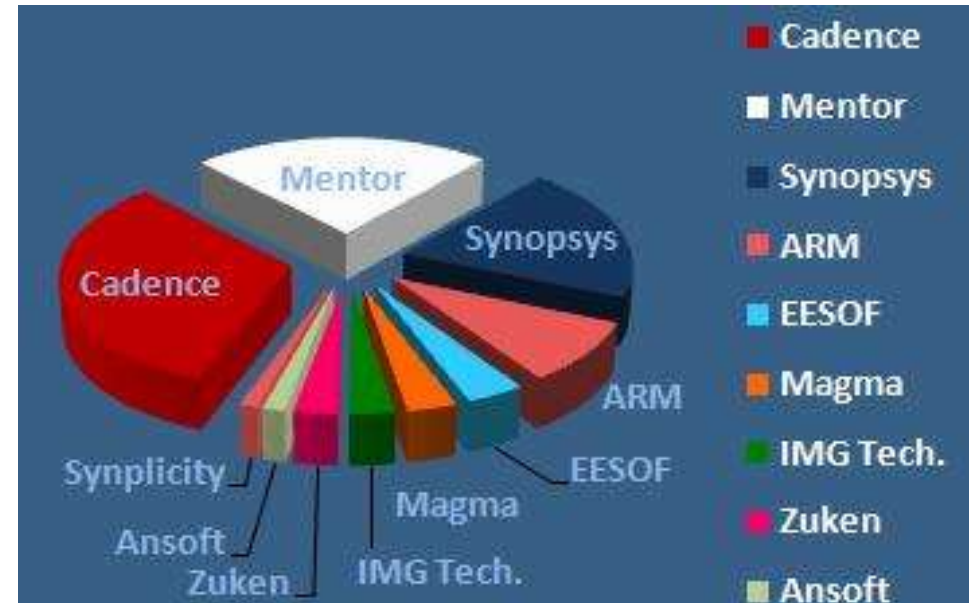
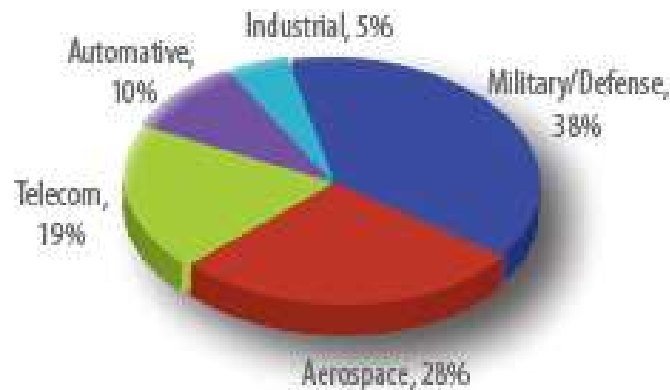
- ChipVerify: Verilog Tutorial  
<https://www.chipverify.com/tutorials/verilog>
- Doulos: The Designer's Guide to Verilog  
<https://www.doulos.com/knowhow/verilog/>
- Aldec: EVITA Interactive Verilog Tutorial  
<http://www.aldec.com/products/tutorials/>
- NAND LAND: Verilog Tutorials and Examples  
<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](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



EDA Playground *alpha* Run Update Copy Share

**Languages & Libraries**

Testbench + Design  
Verilog/SystemVerilog

UVM / OVM  
None

Other Libraries  
None  
OVL 2.7  
SVUnit 2.5

```
1 // Testbench
2 module test;
3
4     reg clk;
5     reg reset;
6     reg d;
7     wire q;
8     wire qb;
9
10 // Instantiate design under test
11 dff DFF(.clk(clk), .reset(reset),
12         .d(d), .q(q), .qb(qb));
13
14 initial begin
15     $display("Reset flop.");
```

▼ Languages & Libraries

Testbench + Design  
SystemVerilog/Verilog

UVM / OVM  
None

Other Libraries  
None  
OVL  
SVUnit

Enable TL-Verilog  
 Enable Easier UVM  
 Enable VUnit

▼ Tools & Simulators  
Select...

Open EPWave after run  
 Show output file after run  
 Download files after run

Brought to you by DOULOS

▼ Languages & Libraries

Testbench + Design  
SystemVerilog/Verilog

UVM / OVM  
UVM 1.2

Other Libraries  
None  
OVL 2.8.1  
SVUnit 2.11

Enable TL-Verilog  
 Enable Easier UVM  
 Enable VUnit

▼ Tools & Simulators  
Synopsys VCS 2019.06

Compile Options  
-timescale=1ns/rts -csc=burst

Run Options  
Run Settings  
 Use run.do file  
 Open EPWave after run  
 Download files after run

▼ Examples

```
1 // Copyright (C) 2011-2012 by Doulos Ltd.
2
3 Licensed under the Apache License, Version 2.0 (the "License");
4 you may not use this file except in compliance with the License.
5 You may obtain a copy of the License at
6
7 http://www.apache.org/licenses/LICENSE-2.0
8
9 Unless required by applicable law or agreed to in writing, software
10 distributed under the license is distributed on an "AS IS" BASIS,
11 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 See the License for the specific language governing permissions and
13 limitations under the License.
14
15 =====
16 // First Steps with UVM - Register Layer
17
18 Author: John Aynley, Doulos
19 Date: 14-Sep-2012
20
21
22 *include "uvm_macros.svh"
23
24 package my_pkg;
25
26 import uvm_pkg::*;
27
28 class my_reg extends uvm_reg;
29     uvm_object_utils(my_reg);
30
31 // An 8-bit register containing a single 8-bit field
32 rand uvm_reg_field f1;
33
34 function new (string name = "");
35     super.new(name, 8, UVM_NO_COVERAGE); // #bits = 8
36 endfunction
37
38 *include "uvm_macros.svh"
39
40 interface dut_if;
41
42 // Simple synchronous bus interface
43 logic clock, reset;
44 logic en;
45 logic cmd;
46 logic [7:0] addr;
47 logic [7:0] wdata;
48 logic [7:0] rdata;
49
50 andinterface
51
52 module dut(dut_if dut);
53
54 import my_pkg::*;
55
56 // Two memory-mapped registers at addresses 0 and 1
57 logic [7:0] r0;
58 logic [7:0] r1;
59
60 always @(posedge dut.clock)
61 begin
62     if (dut.en)
63     begin
64         logic [7:0] value;
65
66         if (dut.cmd == 1)
67             if (dut.addr == 0)
68                 r0 <- dut.wdata;
69             else if (dut.addr == 1)
70                 r1 <- dut.wdata;
71
72         if (dut.cmd == 0)
73             if (dut.addr == 0)
```

- ▼ Examples
- using EDA Playground
  - VHDL
  - Verilog/SystemVerilog
  - UVM
  - EasierUVM
  - SVAUnit
  - SVUnit
  - VUnit (Verilog/SV)
  - VUnit (VHDL)
  - TL-Verilog
  - e + Verilog
  - Python + Verilog
  - Python Only
  - C++/SystemC



## 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**
- **Oddział Kraków!**



**HDL – Aldec Inc.**



- **Praktyki studenckie**
- **Prace dyplomowe (*IP Cores*)**
  - **$\mu$ P/ $\mu$ C: PIC17C4x, PIC16C5x, 8051-SoC**
  - **periferia: 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**



### 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 filtra 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

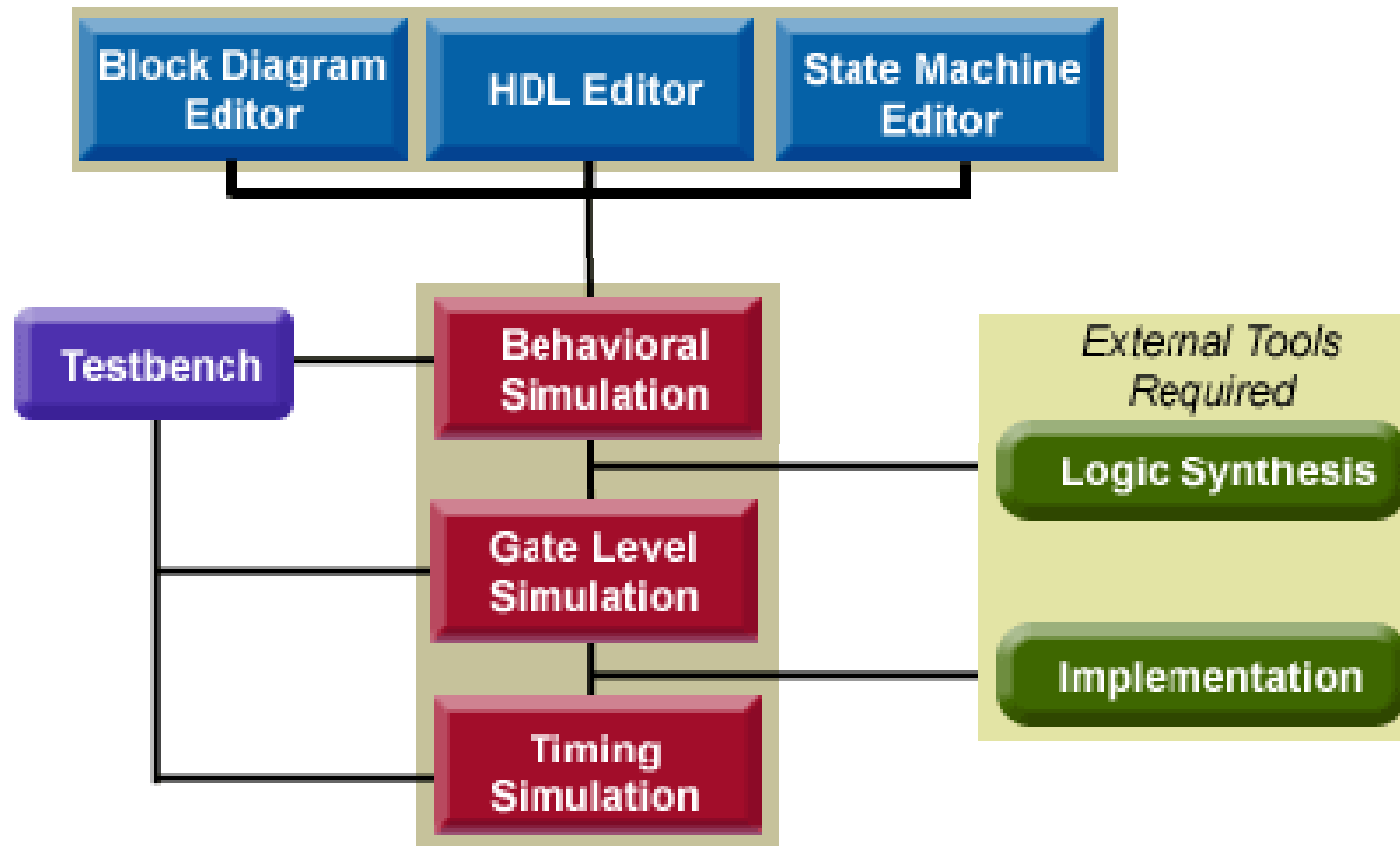
41. Podsystem obsługi wyświetlacza dotykowego LCD w układzie FPGA, współpraca z BTC Korporacja

40. Podsystem tekstowego monitora HDMI w układzie FPGA, współpraca z BTC Korporacja

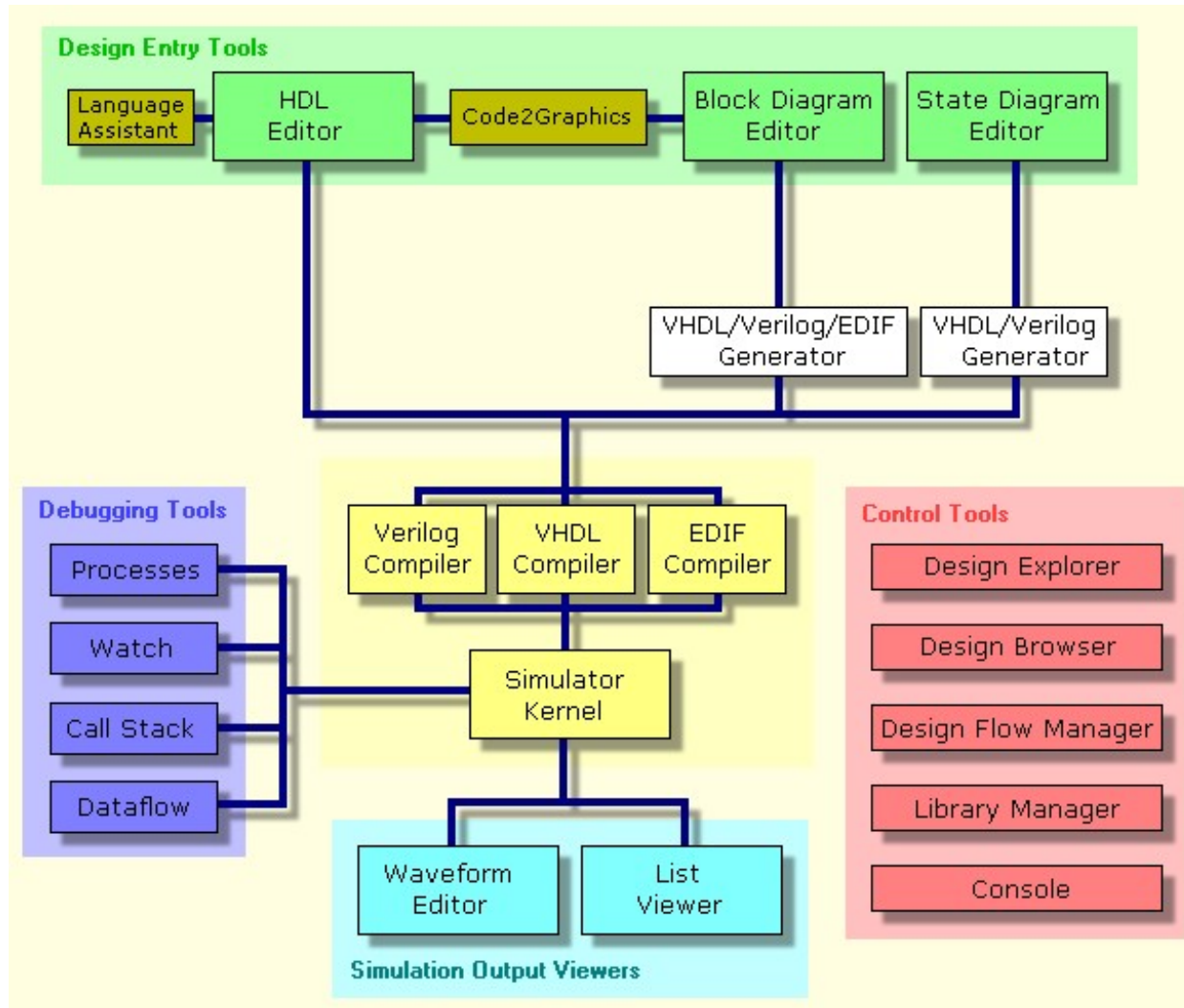
## ActiveHDL – moduły

# Active-HDL™

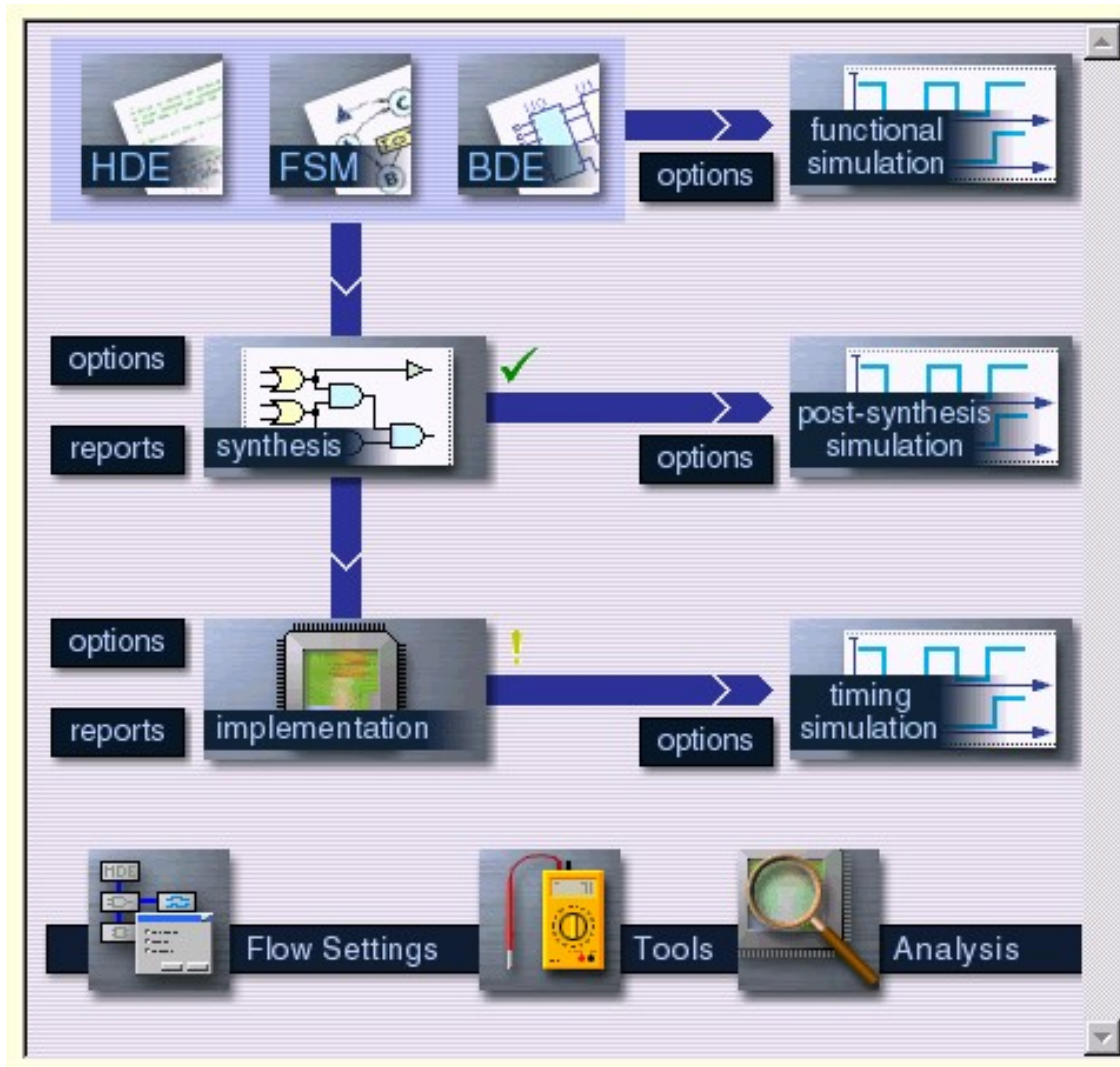
Complete FPGA Verification Environment

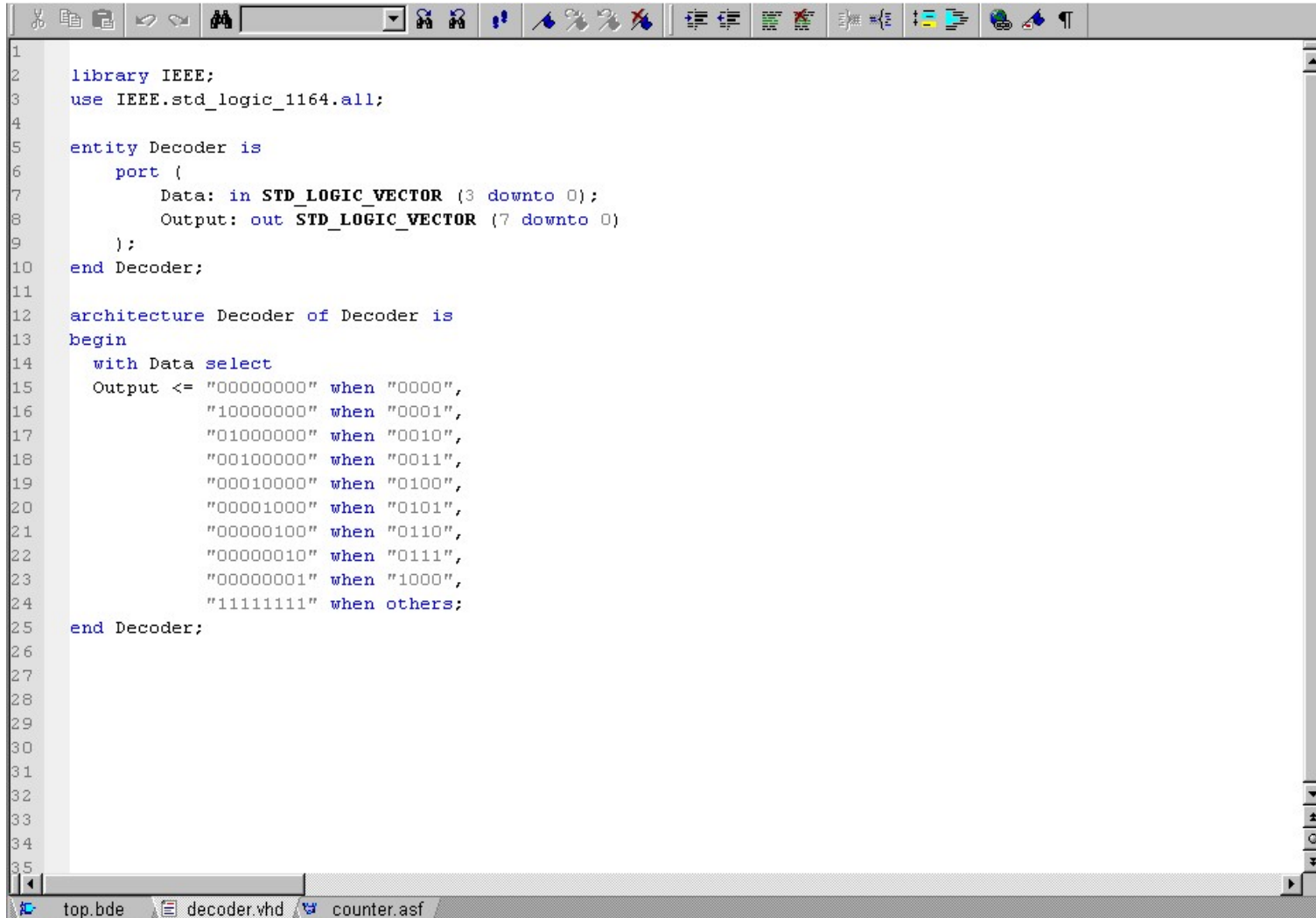


# ActiveHDL – moduły



# ActiveHDL – Design Flow



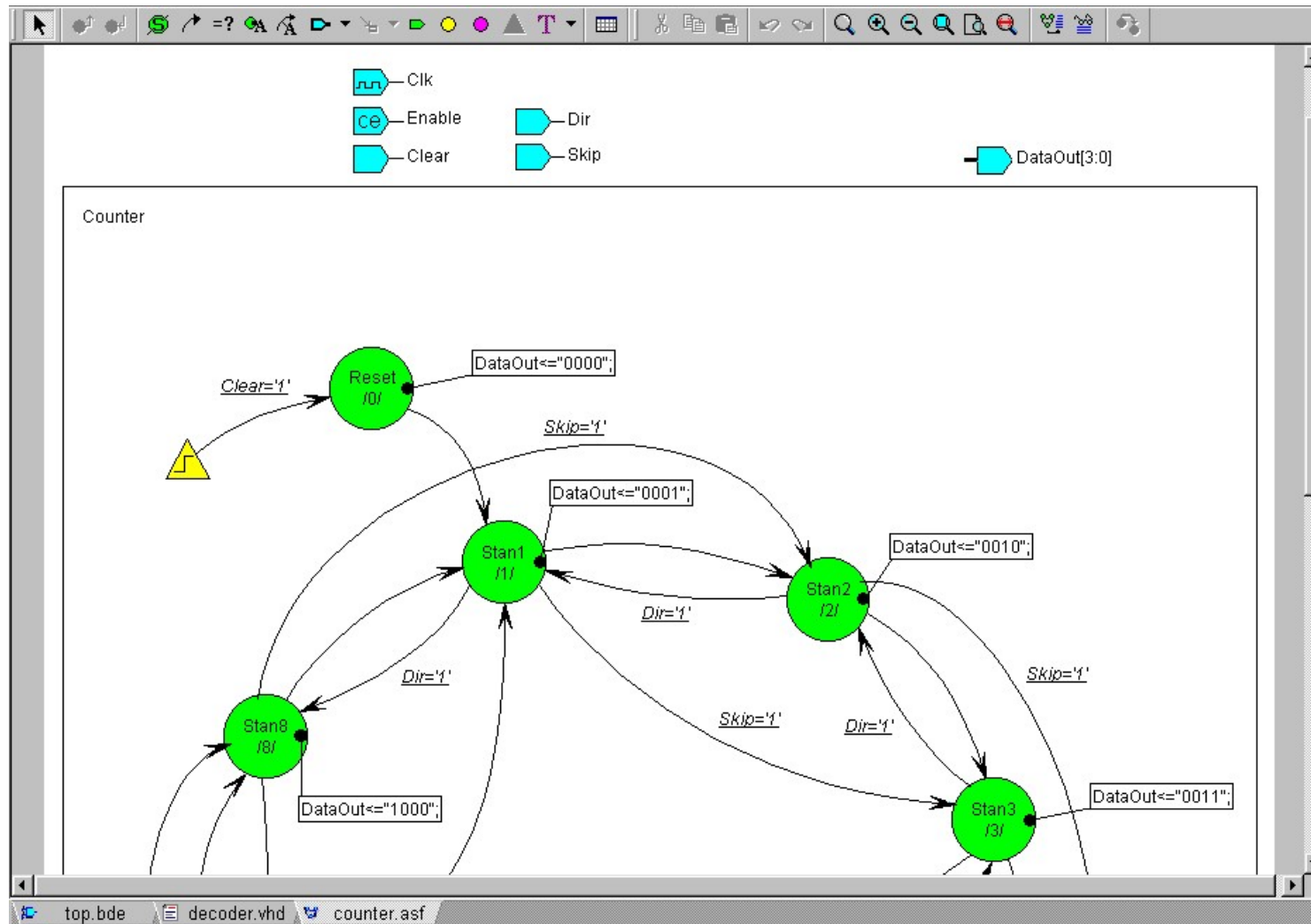


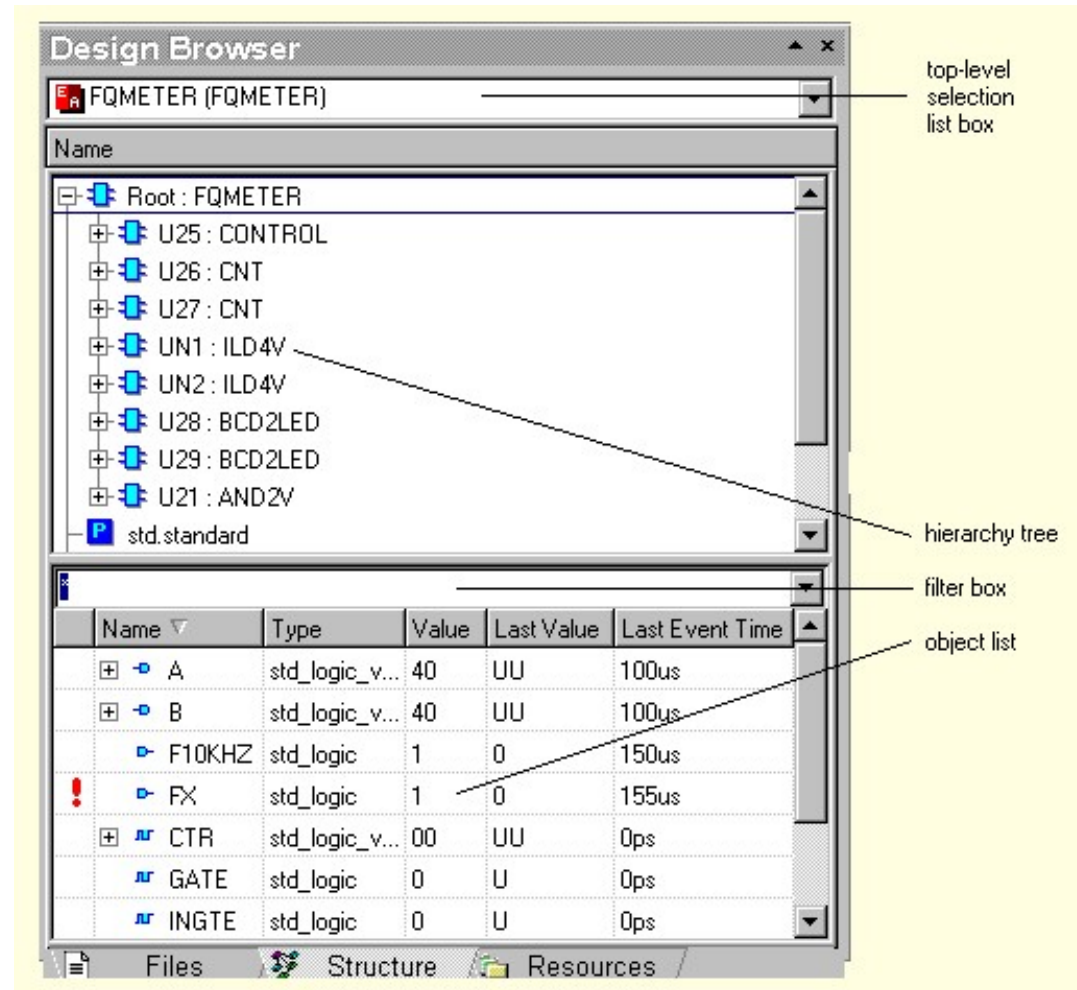
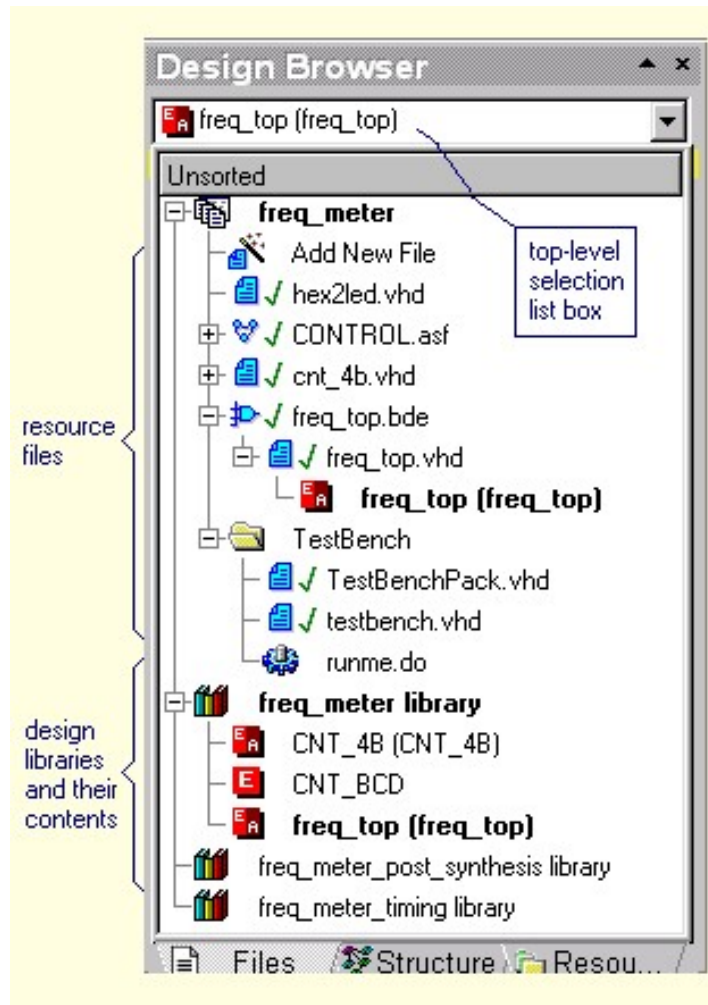
```
1
2 library IEEE;
3 use IEEE.std_logic_1164.all;
4
5 entity Decoder is
6     port (
7         Data: in STD_LOGIC_VECTOR (3 downto 0);
8         Output: out STD_LOGIC_VECTOR (7 downto 0)
9     );
10 end Decoder;
11
12 architecture Decoder of Decoder is
13 begin
14     with Data select
15     Output <= "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

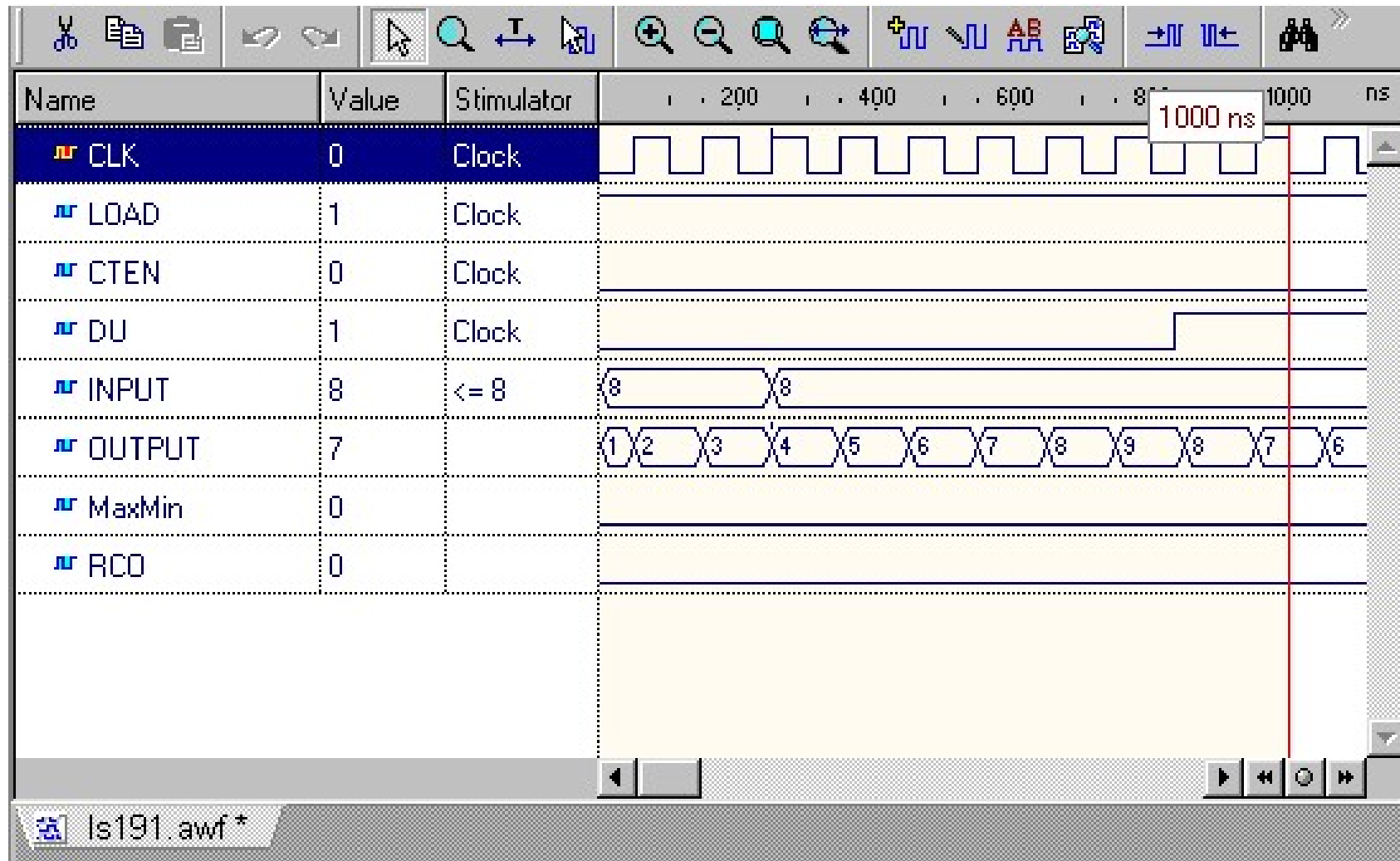


# ActiveHDL – FSM









# ActiveHDL - List Viewer

simulation time      simulation cycle      signals/nets

toolbar

+ Add Signals...

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

list1



# ActiveHDL – Library Manager

The screenshot shows the 'Library Manager' window in ActiveHDL. The window title is 'Library Manager' and it has a menu bar with 'File', 'Edit', 'Search', 'View', 'Design', 'Simulation', 'Tools', 'Library', and 'Help'. Below the menu bar is a toolbar with icons for adding, deleting, and refreshing libraries. The main area is divided into two panes. The left pane shows a list of libraries with columns for 'Library', 'Mode', and 'Comment'. The right pane shows a detailed view of the selected 'vcomponents' library, with columns for 'Unit Name', 'Secondary Unit Name', 'Source Type', and 'Target Language'. Below the detailed view is a 'Package Contents' section showing a grid of component names.

Library	Mode	Comment	Unit Name	Secondary Unit Name	Source Type	Target Language
ovi_uni9000	...	Xilinx Verilog library	xor5	xor5_v	Source Code	VHDL
ovi_uni5200	...	Xilinx Verilog library	xor6	xor6	Block diagram	EDIF
ovi_unisim	...	Xilinx Verilog library	xor7	xor7	Block diagram	EDIF
ovi_xilinxco...	...	Xilinx Verilog library for Co	xor8	xor8	Block diagram	EDIF
synopsys	...	Library containing packa	xor9	xor9	Block diagram	EDIF
simprim_1_4	...	Post P&R timing simulatio	global	global	Source Code	VHDL
simprim	...	Post P&R timing simulatio	vcomponents		Source Code	VHDL
std	...	Standard VHDL library	vpkg	vpkg	Source Code	VHDL
simprim_edif	...	Post P&R timing simulatio	Package Contents			
vi	...	Standard Verilog library	XOR9	XNOR6	X74_352	X74_174
synplify	...	Primitive library for post sy	XOR8	X74_L85	X74_298	X74_168
unisim	...	Functional components li	XOR7	X74_521	X74_283	X74_165S
xc4000e	...	Xilinx schematic library	XOR6	X74_518	X74_280	X74_164
xabelsim	...	Functional components li	XNOR9	X74_42	X74_273	X74_163
xilinxcorelib	...	Functional simulation libra	XNOR8	X74_390	X74_195	X74_162
			XNOR7	X74_377	X74_194	X74_161

Ciąg dalszy  
nastąpi...

