Wybrane zagadnienia projektowania i testowania PLD

- Firmowe systemy projektowe
  - MAX+Plus II (Altera)
  - WebPack (Xilinx)
  - Simili (Sonata EDA)

- Testowanie funkcjonalne projektu

- Własność intelektualna (IP) i rdzenie projektowe

- Testowanie krawędziowe złożonych cyfrowych układów elektronicznych; standard IEEE 1491.1

Wykorzystano materiały firm:

**Altera, Actel, Cypress, Lattice, Xilinx**
World’s Lowest Cost FPGAs

- Platform Design Capability
- Up to 5 Million System Gates
- Lowest Cost Per Gate and Per Pin
- World’s First 90nm FPGA

Xilinx - rodzina FPGA: Spartan-3
Xilinx - Spartan-3

The Spartan-3 Platform FPGA

- Embedded XtremeDSP Functionality
- Advanced FPGA Logic
- High Performance Sync Dual-Port™ RAM
- SelectIO™ Ultra Technology
- Digital Clock Management
- Staggered Pad Technology
- XCITE Digitally Controlled Impedance
Xilinx - Spartan-3: główne właściwości

• Revolutionary 90-nanometer process technology
• Very low cost, high-performance logic solution
  - Densities as high as 74,880 logic cells
  - 326 MHz system clock rate
  - Three separate power supplies for the core (1.2V), I/Os (1.2V to 3.3V), and special functions (2.5V)
    - Up to 784 I/O pins
    - 622 Mb/s data transfer rate per I/O
    - Seventeen single-ended signal standards
    - Six differential signal standards including LVDS
    - Termination by Digitally Controlled Impedance
    - Signal swing ranging from 1.14V to 3.45V
    - Double Data Rate (DDR) support

Logic resources
  - Abundant, flexible logic cells with registers
  - Wide multiplexers
  - Fast look-ahead carry logic
  - Dedicated 18 x 18 multipliers
  - JTAG logic compatible with IEEE 1149.1/1532

Standards
  - Up to 1,872 Kbits of total block RAM
  - Up to 520 Kbits of total distributed RAM

• Digital Clock Manager (up to four DCMs)
  - Clock skew elimination
  - Frequency synthesis
  - High resolution phase shifting
Table 1: Summary of Spartan-3 FPGA Attributes

<table>
<thead>
<tr>
<th>Device</th>
<th>System Gates</th>
<th>Logic Cells</th>
<th>CLB Array (One CLB = Four Slices)</th>
<th>Distributed RAM (bits(^1))</th>
<th>Block RAM (bits(^1))</th>
<th>Dedicated Multipliers</th>
<th>DCMs</th>
<th>Maximum User I/O</th>
<th>Maximum Differential I/O Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC3S50</td>
<td>50K</td>
<td>1,728</td>
<td>16  12  192</td>
<td>12K</td>
<td>72K</td>
<td>4</td>
<td>2</td>
<td>124</td>
<td>56</td>
</tr>
<tr>
<td>XC3S200</td>
<td>200K</td>
<td>4,320</td>
<td>24  20  480</td>
<td>30K</td>
<td>216K</td>
<td>12</td>
<td>4</td>
<td>173</td>
<td>76</td>
</tr>
<tr>
<td>XC3S400</td>
<td>400K</td>
<td>8,064</td>
<td>32  28  896</td>
<td>56K</td>
<td>288K</td>
<td>16</td>
<td>4</td>
<td>264</td>
<td>116</td>
</tr>
<tr>
<td>XC3S1000</td>
<td>1M</td>
<td>17,280</td>
<td>48  40  1,920</td>
<td>120K</td>
<td>432K</td>
<td>24</td>
<td>4</td>
<td>391</td>
<td>175</td>
</tr>
<tr>
<td>XC3S1500</td>
<td>1.5M</td>
<td>29,952</td>
<td>64  52  3,328</td>
<td>208K</td>
<td>576K</td>
<td>32</td>
<td>4</td>
<td>487</td>
<td>221</td>
</tr>
<tr>
<td>XC3S2000</td>
<td>2M</td>
<td>46,080</td>
<td>80  64  5,120</td>
<td>320K</td>
<td>720K</td>
<td>40</td>
<td>4</td>
<td>565</td>
<td>270</td>
</tr>
<tr>
<td>XC3S4000</td>
<td>4M</td>
<td>62,208</td>
<td>96  72  6,912</td>
<td>432K</td>
<td>1,728K</td>
<td>96</td>
<td>4</td>
<td>712</td>
<td>312</td>
</tr>
<tr>
<td>XC3S5000</td>
<td>5M</td>
<td>74,880</td>
<td>104 80  8,320</td>
<td>520K</td>
<td>1,872K</td>
<td>104</td>
<td>4</td>
<td>784</td>
<td>344</td>
</tr>
</tbody>
</table>

Notes:
1. By convention, one Kb is equivalent to 1,024 bits.
# Xilinx - Spartan-3

## The Spartan-3 FPGA Family

<table>
<thead>
<tr>
<th>Device</th>
<th>XC3S50</th>
<th>XC3S200</th>
<th>XC3S400</th>
<th>XC3S1000</th>
<th>XC3S1500</th>
<th>XC3S2000</th>
<th>XC3S4000</th>
<th>XC3S5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Gates</td>
<td>50K</td>
<td>200K</td>
<td>400K</td>
<td>1000K</td>
<td>1500K</td>
<td>2000K</td>
<td>4000K</td>
<td>5000K</td>
</tr>
<tr>
<td>Logic Cells</td>
<td>1,728</td>
<td>4,320</td>
<td>8,064</td>
<td>17,280</td>
<td>29,952</td>
<td>46,080</td>
<td>62,208</td>
<td>74,880</td>
</tr>
<tr>
<td>18 x 18 Multipliers</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>96</td>
<td>104</td>
</tr>
<tr>
<td>Block RAM Blocks</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>96</td>
<td>104</td>
</tr>
<tr>
<td>Block RAM Bits</td>
<td>72K</td>
<td>216K</td>
<td>288K</td>
<td>432K</td>
<td>576K</td>
<td>720K</td>
<td>1,728K</td>
<td>1,872K</td>
</tr>
<tr>
<td>Distributed RAM Bits</td>
<td>12K</td>
<td>30K</td>
<td>56K</td>
<td>120K</td>
<td>208K</td>
<td>320K</td>
<td>432K</td>
<td>520K</td>
</tr>
<tr>
<td>DCMs</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>I/O Standards</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>Max Differential I/O Pairs</td>
<td>56</td>
<td>76</td>
<td>116</td>
<td>175</td>
<td>221</td>
<td>270</td>
<td>312</td>
<td>344</td>
</tr>
<tr>
<td>Max Single Ended I/O</td>
<td>124</td>
<td>173</td>
<td>264</td>
<td>391</td>
<td>487</td>
<td>565</td>
<td>712</td>
<td>784</td>
</tr>
</tbody>
</table>

## Package and I/O Offerings

<table>
<thead>
<tr>
<th>Package</th>
<th>XC3S50</th>
<th>XC3S200</th>
<th>XC3S400</th>
<th>XC3S1000</th>
<th>XC3S1500</th>
<th>XC3S2000</th>
<th>XC3S4000</th>
<th>XC3S5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQ100</td>
<td>14X14 mm</td>
<td>63</td>
<td>63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TQ144</td>
<td>20X20 mm</td>
<td>97</td>
<td>97</td>
<td>97</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ208</td>
<td>28X28 mm</td>
<td>124</td>
<td>141</td>
<td>141</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FT256</td>
<td>17X17 mm</td>
<td>173</td>
<td>173</td>
<td>173</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FG456</td>
<td>23X23 mm</td>
<td>264</td>
<td>333</td>
<td>333</td>
<td>489</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FG676</td>
<td>27X27 mm</td>
<td>391</td>
<td>487</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FG900</td>
<td>31X31 mm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>565</td>
</tr>
<tr>
<td>FG1156</td>
<td>35X35 mm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>712</td>
<td>784</td>
</tr>
</tbody>
</table>
The Spartan-3 family architecture consists of five fundamental programmable functional elements:

- **Configurable Logic Blocks (CLBs)** contain RAM-based Look-Up Tables (LUTs) to implement logic and storage elements that can be used as flip-flops or latches. CLBs can be programmed to perform a wide variety of logical functions as well as to store data.

- **Input/Output Blocks (IOBs)** control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. Twenty-three different signal standards, including six high-performance differential standards. Double Data-Rate (DDR) registers are included. The Digitally Controlled Impedance (DCI) feature provides automatic on-chip terminations, simplifying board designs.

- **Block RAM** provides data storage in the form of 18-Kbit dual-port blocks.

- **Multiplier blocks** accept two 18-bit binary numbers as inputs and calculate the product.

- **Digital Clock Manager (DCM)** blocks provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase shifting clock signals.
Xilinx - Spartan-3 : architektura
Xilinx - Spartan-3 : I/O (1)
Xilinx - Spartan-3 : I/O (2)

Note: All IOB signals communicating with the FPGA’s internal logic have the option of inverting polarity.

Figure 1: Simplified IOB Diagram
Xilinx - Spartan-3 : I/O (2)
Xilinx - Spartan-3 : "plasterki" w bloku CLB

Figure 5: Arrangement of Slices within the CLB
Xilinx - Spartan-3: pamięć dwuwrotna RAM

Figure 7: Block RAM Data Paths
Xilinx - Spartan-3 : mnożarki 18x18

(a) Asynchronous 18-bit Multiplier

(b) 18-bit Multiplier with Register at Outputs
Xilinx - Spartan-3: zarządzanie sygnałami zegarowymi
The

Xilinx - Spartan-3: rodzaje połączeń między blokami CLB

(a) Long Line

(b) Hex Line

(c) Double Line

(d) Direct Lines
Xilinx - Spartan-3

Configuration
Spartan-3 FPGAs are programmed by loading configuration data into robust static memory cells that collectively control all functional elements and routing resources. Before powering on the FPGA, configuration data is stored externally in a PROM or some other nonvolatile medium either on or off the board. After applying power, the configuration data is written to the FPGA using any of five different modes: Master Parallel, Slave Parallel, Master Serial, Slave Serial and Boundary Scan (JTAG). The Master and Slave Parallel modes use an 8-bit wide SelectMAP™ Port. The recommended memory for storing the configuration data is the low-cost Xilinx Platform Flash PROM family, which includes XCF00S PROMs for serial configuration and XCF00P PROMs for parallel configuration.
Xilinx - Spartan-3: tryby programowania

The **Configuration Modes**
Spartan-3 supports the following five configuration modes:
- **Slave Serial mode**
- **Master Serial mode**
- **Slave Parallel mode**
- **Master Parallel mode**
- **Boundary-Scan (JTA 1149.1)**
The **Configuration Modes**
Spartan-3 supports the following five modes:
- Slave Serial mode
- Master Serial mode
- **Slave Parallel mode**
- Master Parallel mode
- Boundary-Scan (JTAG) mode (IEEE 1149.1)
The **Configuration Modes**

Spartan-3 supports the following five configuration modes:

- Slave Serial mode
- Master Serial mode
- Slave Parallel mode
- **Master Parallel mode**
- Boundary-Scan (JTAG) mode (IE 1149.1)
Xilinx - schemat procesu projektowania z WebPack

FPGA:
1. **Translate** – interpretacja projektu, sprawdzenie reguł projektowych
2. **Map** – oblicza potrzebne zasoby i ich rozmieszczenie
3. **Place and Route** – Lokuje CLB w odpowiednich miejscach i wytycza ścieżki połączeń.
4. **Generate Programming File** – Tworzy ciąg bitów programujących

CPLD:
1. **Translate** – interpretacja projektu, sprawdzenie reguł projektowych
2. **Fit** – Rozmieszcza zasoby logiczne i połączenia.
3. **Generate Programming File** – Generuje "mapę przepaleń bezpieczników" , czyli plik *.jed dla zaprogramowania połączeń
Xilinx - WebPack - State Machine Tool
Xilinx - WebPack - State Machine Tool
Each state can have a transition which returns to it (loop back), as well as transitions going from it to the next state and previous state. The sample window shows the effects of your selections.

To place the state machine, click Finish. Move the cursor to the desired location and click the left mouse button.

Add Transitions | Set condition to
--- | ---
Loop back: | @ELSE
Next: | TIMER
Previous: | 

Help | < Back | Finish | Cancel

Figure 4-17: Setup Transitions Window
WEBPACK HDL BENCHER TOOL

The HDL Bencher tool generates the testbenches, allowing you to simulate the design under test.

The HDL Bencher tool reads the design under test. You simply enter signal transitions in a graphical timing diagram GUI.

You can also enter your expected simulation results, allowing the simulator to flag a warning if the simulation did not yield the expected results.
Xilinx - WebPack - Bench Test - PatternWizard

Choose Pattern

- Count Down
- Count Up

Description

Counts up from the initial value every nn cycles. The pattern repeats after the terminal value has been reached.

Radix

16 10 2

Do for # cycles:

16

Customize Pattern

Initial Value

0

Increment By

1

Terminal Value

1111

Count Every

1
Xilinx - WebPack - przykładowy temat projektu

**Rysunek 7.1: Kontroler krokomotora**
Xilinx - WebPack - implementacje \( \mu P \leftrightarrow \text{PLD} \)

**Design Implementation**

The stepper motor controller can be implemented using either a microcontroller or a Programmable Logic Device.

**Figure 7-2: Design Flow Comparison**
Intellectual Property (IP) is defined as very complex pre-tested system-level functions that are used in logic designs to dramatically shorten development time. The core benefits are:

- Faster Time-to-Market
- Simplifies the development process
- Minimal Design Risk
- Reduces software compile time
- Reduced verification time
- Predictable performance/functionality

Cores are similar to vendor-provided soft macros in that they simplify the design specification step by removing the designer from gate-level details of commonly used functions. Cores differ from soft macros in that they are generally much larger system-level functions such as, PCI bus interface, DSP filter, PCMCIA interface, etc. They are extensively tested (and hence rarely free of charge) to offload the designer from having to verify the core functions himself. The Xilinx website has a comprehensive data base of Xilinx (LogiCORE) and 3rd Party (AllianceCORE) verified & tested cores, these can be found by interrogating the on-line search facility called the ‘IP Center’.

www.xilinx.com/ipcenter

The CORE Generator tool form Xilinx delivers highly optimised cores that are compatible with standard design methodologies for Xilinx FPGAs. This easy-to-use tool generates flexible, high performance cores with a high degree of predictability and allows customers to download
In the 1980s, the Joint Test Action Group (JTAG) developed a specification for boundary-scan testing that was later standardized as the IEEE Std. 1149.1 specification. This boundary-scan test (BST) architecture offers the capability to efficiently test components on PCBs with tight lead spacing. This BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. Boundary-scan cells in a device can force signals onto pins, or capture data from pin or core logic signals. Forced test data is serially shifted into the boundary-scan cells. Captured data is serially shifted out and externally compared to expected results. Figure 1 illustrates the concept of boundary-scan testing.
IEEE 1532 Programming Standard

IEEE 1532 refers to an industry standardization effort aimed at simplifying the challenge of programming In-System Configurable (ISC) devices using the industry standard IEEE 1149.1 Boundary Scan Test Access Port. The IEEE 1532 effort is focused on the standardization of the programming algorithm at the component level (silicon) as well as the definition of the software required to describe the programming algorithm and associated programming data. The silicon portion of the standard establishes a common device behavior during programming via the IEEE 1149.1 state machine. Any device which claims compatibility or compliance with this standard will behave predictably and consistently when given an ISC instruction. The software portion of the IEEE 1532 standard defines a modified Boundary Scan Description Language file (ISC BSDL file) which has been expanded to cover the new ISC instructions. Additionally, there is a new data file (ISC data file) which contains all of the device and pattern-specific programming data.
Zalety realizacji ISP poprzez JTAG

In-system programming using a standard boundary scan test interface is necessary for compatibility with advanced board testing techniques. If a design incorporates 1149.1-ISP devices, then no separate programming interface is needed. All IEEE-1149.1 compatible or compliant devices (logic, interconnect and analog) can be used in the same scan chain. ISP devices make design jobs easier by simplifying device configuration. Designers have the option of soldering parts directly on the board and then programming them through the TAP pins. In the design phase, ISP devices let designers implement redesigns within a few seconds by making changes directly to devices on the board. This speeds up the design process and reduces time to market. ISP devices also offer benefits for manufacturing. Lower inventory cost is achieved because blank devices can be used for manufacturing and then programmed at test time. This eliminates the need to maintain a separate inventory part number for each programmed part and improves the manufacturing process by facilitating board connectivity testing. Once the design is finalized and the board assembled, manufacturing engineers can use testers for both board connectivity testing and programming. As a result, 1149.1-ISP eliminates the cost of separate programming stations, unnecessary manufacturing steps and excessive handling. This shortens production time, reduces scrap cost and increases reliability.
Altera - BScan testing
Schemat blokowy TAP wg standardu IEEE 1491.1

In its simplest form, the 1149.1 standard is implemented using a four-pin, dedicated test access port, a 16-state, synchronous state machine and a group of data registers. The data registers include the bypass register and a boundary scan register that is used to control the inputs and outputs of the device being tested. It also needs an instruction register and instruction register decoder used to control the data registers. Figure 1 shows a top-level diagram of a basic implementation of the 1149.1 standard.

**Figure 1. IEEE 1149.1 Block Diagram**

There are four pins that make up the Test Access Port (TAP): TDI (Test Data Input), TMS (Test Mode Select), TCK (Test Clock), and TDO (Test Data Output). An additional pin defined by the standard, TRST (Test ReSeT), can be used to asynchronously reset both the TAP controller and the instruction register. All registers, along with the TAP controller, are clocked using the TCK pin.
### Table 2. IEEE Std. 1149.1 Pin Descriptions

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDI</td>
<td>Test data input</td>
<td>Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK.</td>
</tr>
<tr>
<td>TDO</td>
<td>Test data output</td>
<td>Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device.</td>
</tr>
<tr>
<td>TMS</td>
<td>Test mode select</td>
<td>Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur at the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK.</td>
</tr>
<tr>
<td>TCK</td>
<td>Test clock input</td>
<td>The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge.</td>
</tr>
<tr>
<td>TRST</td>
<td>Test reset input (optional)</td>
<td>Active-low input to asynchronously reset the boundary-scan circuit. (TRST is optional according to IEEE Std. 1149.1). This pin should be driven low when not in boundary scan operation and for non-JTAG users the pin should be permanently tied to GND. It is not supported by all families.</td>
</tr>
</tbody>
</table>
BST - repository BoundaryScanRegisters, Bypass Register
BST - konfiguracje rejestrów
Schemat blokowy układu testowania krawędziowego

Figure 2. IEEE Std. 1149.1 Circuitry
Schemat blokowy układu testowania krawędziowego

Figure 3. Boundary-Scan Register

Each peripheral element is either an I/O pin, dedicated input pin, or dedicated configuration pin.
Altera - układy sprzętowe I/O realizujące BST

Figure 6. An APEX 20K, ACEX 1K, FLEX 10K, FLEX 6000 & FLEX 8000 User I/O BSC with IEEE Std. 1149.1 BST Circuitry
### BST - instrukcje

#### Table 11. Boundary Scan Instructions and Descriptions

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLE/PRELOAD</td>
<td>Allows a snapshot of the signals at the device pins to be captured and examined during normal device operation, and permits an initial data pattern to be output at the device pins.</td>
</tr>
<tr>
<td>EXTEST</td>
<td>Allows the external circuitry and board-level interconnections to be tested by forcing a test pattern at the output pins and capturing test results at the input pins.</td>
</tr>
<tr>
<td>BYPASS</td>
<td>Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through the selected device to adjacent devices during normal device operation.</td>
</tr>
<tr>
<td>IDCODE</td>
<td>Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO.</td>
</tr>
<tr>
<td>USERCODE</td>
<td>Selects the USERCODE register and places it between TDI and TDO, allowing the USERCODE to be serially shifted out of TDO.</td>
</tr>
<tr>
<td>CLAMP (&lt;sup&gt;1&lt;/sup&gt;)</td>
<td>Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation, while holding I/O pins to a state defined by the data in the boundary scan register.</td>
</tr>
<tr>
<td>HIGHZ (&lt;sup&gt;1&lt;/sup&gt;)</td>
<td>Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation, while tri-stating all of the I/O pins.</td>
</tr>
</tbody>
</table>
Automat sterownika TAP realizujący BST
Automat sterownika TAP realizujący BST