HEPCAT FPGA LAB

Riley Gleason

UC Irvine

7/23/2024

FPGA LAB

Intro:

- Field programmable gate arrays (FPGAs) offer high logic density and Reconfigurability making them essential in modern data acquisition and control systems.

- The two leading FPGA manufacturers are Altera and Xilinx
  -Xilinx often preferred for data acquisition designs.

- This lab utilizes a Xilinx FPGA from the Artix-7 family on a Digilent Basys3 development board which includes digital inputs and outputs, a built-in ADC, switches, LEDs, and a 4-digit 7-segment display.

FPGA LAB

Explanation:

Why do we use FPGAs(Field-Programmable Gate Array)?

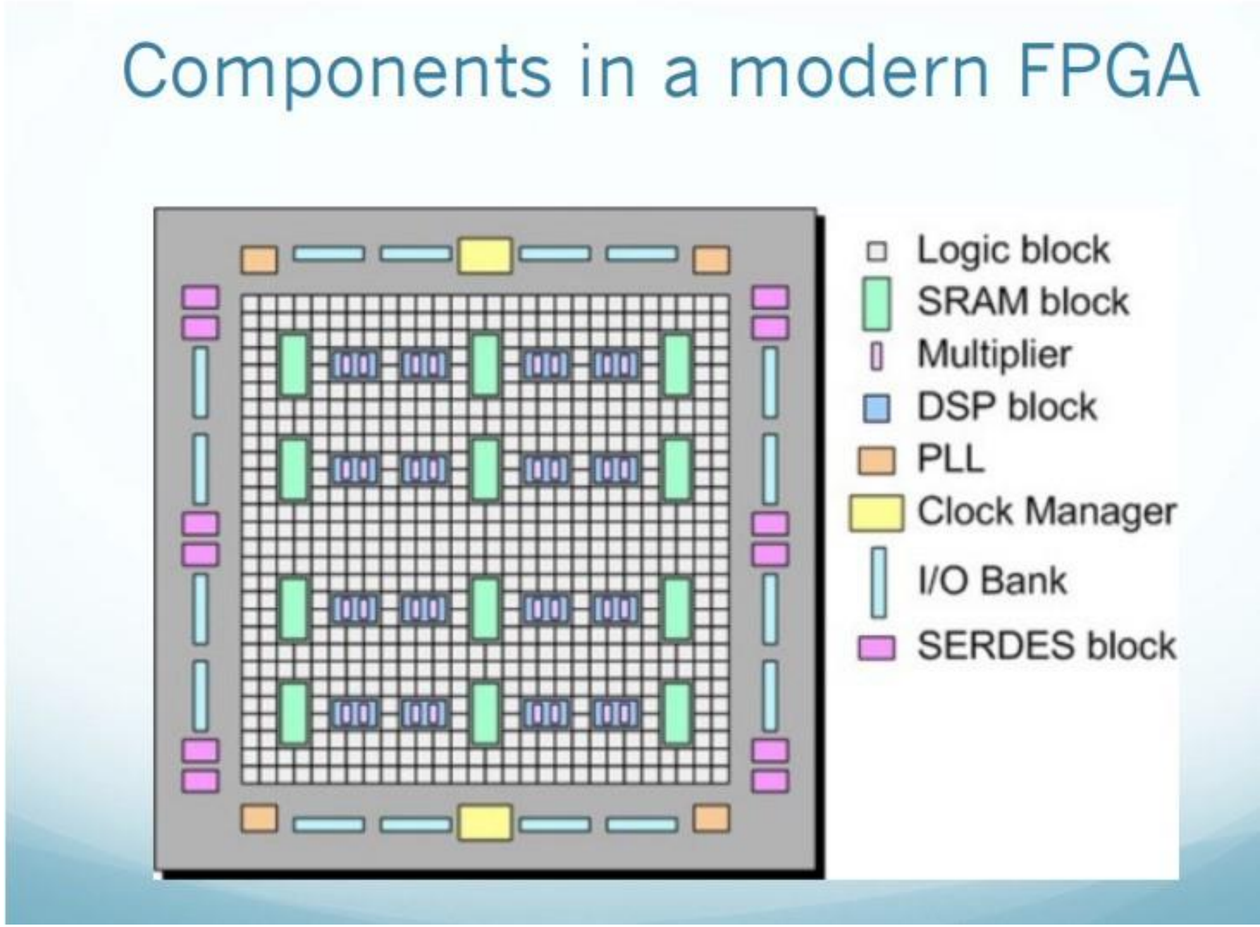
- FPGAs allow researchers to design and implement custom hardware accelerators tailored to the specific algorithms and computations needed for data processing, enabling efficient and high-speed execution.

- Facilitates rapid design, testing, and modification of hardware configurations and algorithms
- Empowers effective exploration of various design alternatives
- Excel at parallel processing
  - FPGAs can be programmed to handle multiple parallel tasks simultaneously

FPGA LAB

Explanation:

- Integrated circuits allow us to program customized digital logic in the field
  - They contain configurable logic blocks (CLBs)
  - interconnected by programmable routing resources
- Each CLB typically contains a combination of look-up tables (LUTs), flip-flops, multiplexers, and other components.



FPGA LAB

Goal:

Make LEDs blink on the Basys3 board

Steps:

1. Writing Verilog code to specify the desired logical behavior.
2. Constraining the mapping between the internal logic signals and the pins that connect to the board
3. Compiling your code and generating the configuration (.bit) file.
4. Downloading the configuration to Xilinx chip on the Basys3 board.
5. FPGA chip’s resources

FPGA LAB

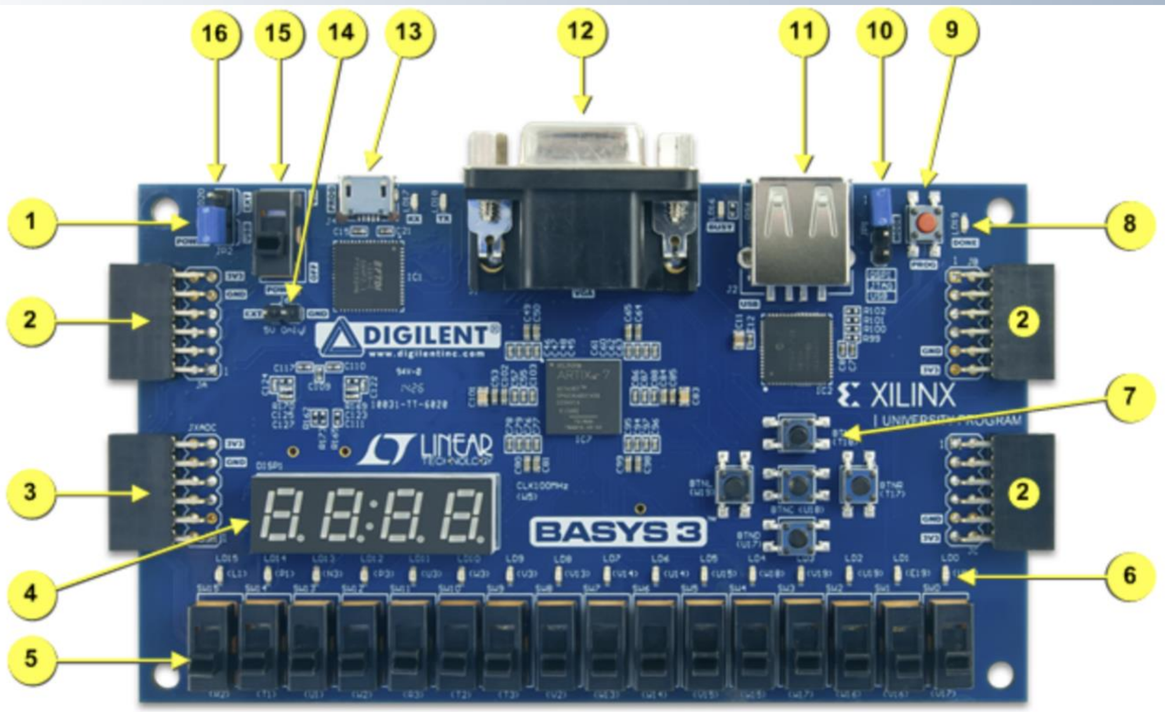
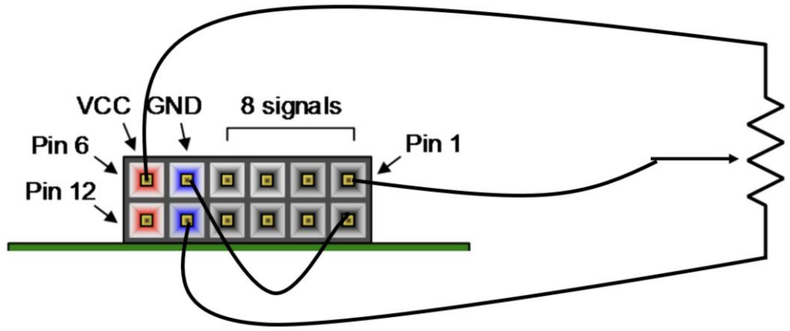


Figure 1. Basys3 board features

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod connector(s)	10	Programming mode jumper
3	Analog signal Pmod connector (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper



ADC Input

Basys3 board

Step 1

Explanation:

- Verilog is a hardware description language (HDL)
- Allows designers to describe the behavior and structure of electronic circuits.
  -It can represent both combinational and sequential logic.
- Abstraction Levels:
  -Behavioral Level: Describes the functionality of a circuit without detailing its implementation.
  -RTL (Register Transfer Level): Describes data flow and timing of data through registers and combinational logic.
  -Gate Level: Describes the circuit in terms of logical gates and interconnections.
- Supports parallel nature of hardware.
- Simulation and Synthesis: Verilog can be simulated to verify the logic and functionality of a design. It can also be synthesized to generate a netlist for physical implementation in hardware.

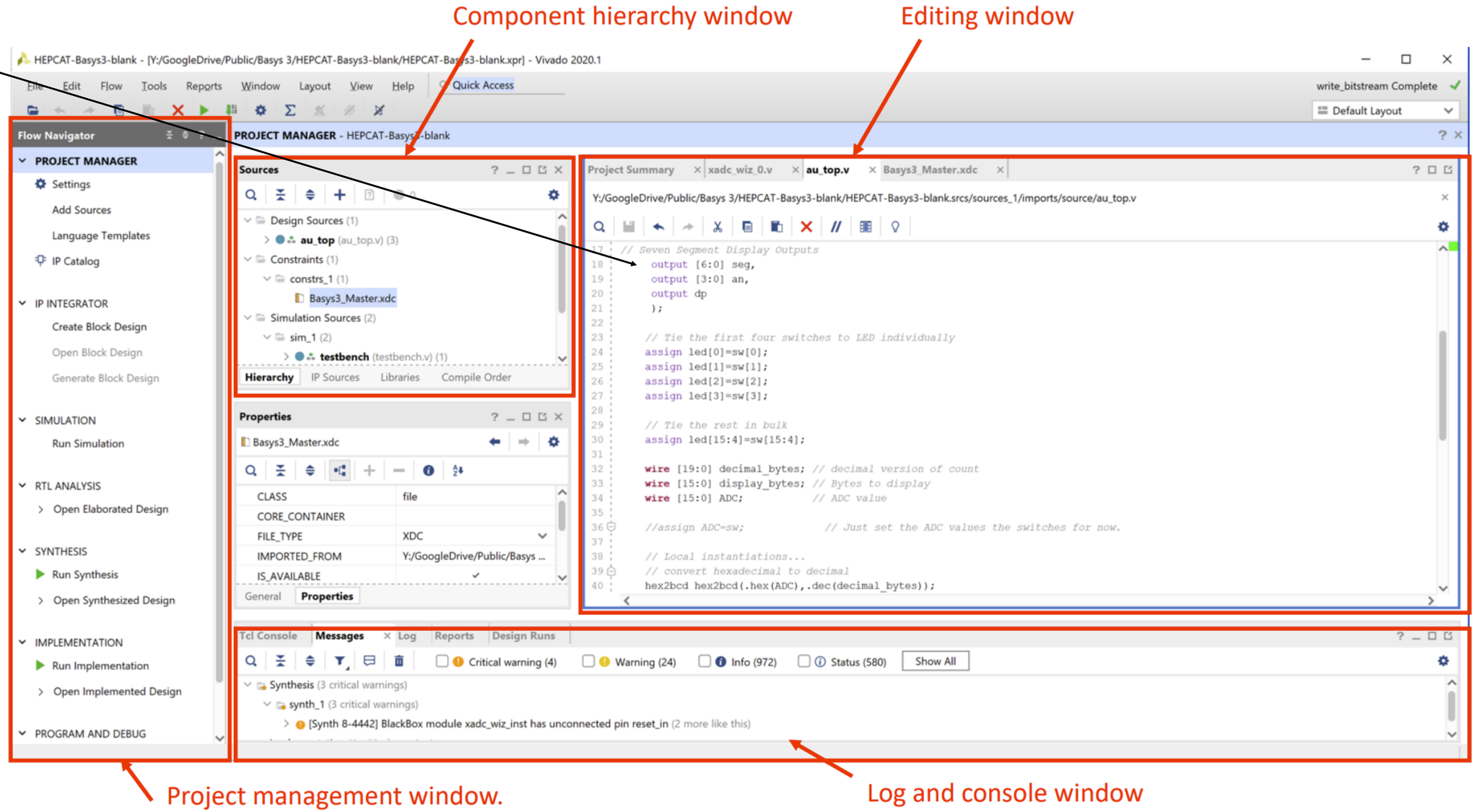
FPGA LAB

Step 1

Explanation:

- HDL (Hardware Description Language) is a specialized computer language used to describe the structure, design, and operation of electronic circuits, particularly digital logic circuits.
- HDLs enable designers to model complex systems and simulate their behavior before physically implementing them in hardware.

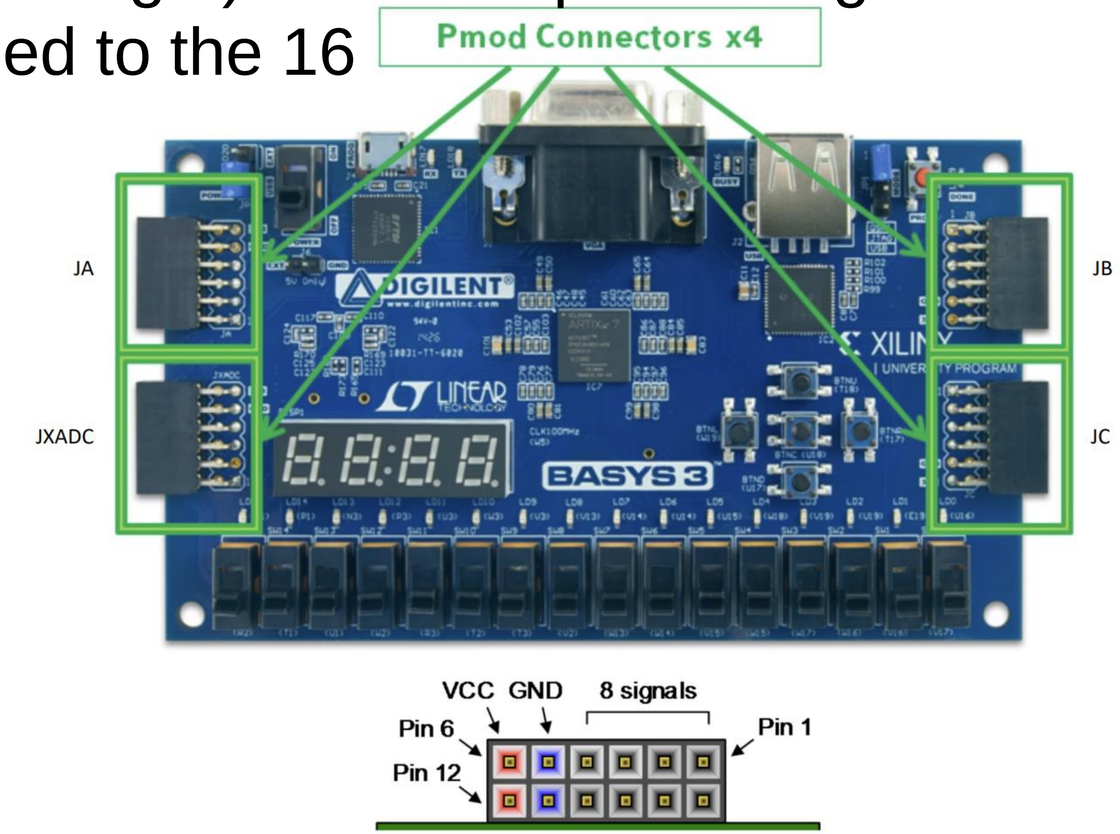
Step 1 Explanation:



Step 2

Explanation:

- By using the mapping(on the right) and a simple Verilog code
- You will connect 16 switched to the 16 LEDs



Digital Mapping (Jx=JA,JB,JC)

Jx Pin	Input/ Output
1	Jx[0]
2	Jx[1]
3	Jx[2]
4	Jx[3]
7	Jx[4]
8	Jx[5]
9	Jx[6]
10	Jx[7]

Analog Mapping

JXADC Pin	Input	Analog Input
1	JXADC[0]	vauxp6
2	JXADC[1]	vauxp14
3	JXADC[2]	vauxp7
4	JXADC[3]	vauxp15
7	JXADC[4]	vauxn6
8	JXADC[5]	vauxn14
9	JXADC[6]	vauxn7
10	JXADC[7]	vauxn15

Figure 4: The mapping of the PMOD connectors to the inputs as defined in the constraint file.

Step 2

Digital Display:

- Additionally, there will be a digital display (4 digits) which will be another item to code producing hexadecimal and converting that to decimal using Binary Coded Decimal (BCD)
  -four bits are used for each decimal digit 0-9

Step 3

Explanation:

- Synthesis transforms the Verilog code into a detailed description of how the actual hardware should be constructed using the basic components available in the target.
- Implementation: The design is configured for the specific chip, including connections to the I/O pins.

Step 3

Explanation:

What Happens During Synthesis

- Debug: The synthesis tool reads and interprets the Verilog code.
  It checks for syntax and semantic correctness and elaborates the design, which involves resolving all instances and hierarchy.
- Optimization: The tool optimizes the design by minimizing the number of logic gates and other components required. This step includes various optimizations such as logic minimization, constant propagation, and resource sharing.

Step 3

Explanation:

What Happens During Synthesis

- Mapping to Gate-Level Primitives: The high-level constructs in the Verilog code (like always blocks, if-else statements, and arithmetic operations) are mapped to a netlist of gate-level primitives.
- Technology Mapping: The gate-level netlist is further mapped to the specific library of components provided by the target technology (e.g., FPGA, ASIC). Each component in the netlist is replaced by a corresponding element from the technology library, which includes specific details about the electrical characteristics and physical layout.

Step 3

Explanation:

Definitions:

- ASIC (Application-Specific Integrated Circuit) is a type of integrated circuit (IC) designed for a particular use or application, as opposed to general-purpose ICs. ASICs are customized for specific tasks, making them highly efficient for their intended purposes but less flexible than programmable chips like FPGAs.

- Gate-level primitives are the basic building blocks used in digital circuit design. These primitives represent simple logic gates, flip-flops, and other fundamental components that form the foundation of more complex digital systems.

Step 3

Explanation:

Definitions:

- Types of Gate-Level Primitives
  Logic Gates: These are the fundamental elements that perform basic logic functions.
  -The primary types of logic gates are: AND, OR, and NOT Gates(etc.)
- Flip-Flops and Latches: These are used for storing state information (memory elements).
- Multiplexers (MUX): Selects one of many inputs to pass through to the output based on a select signal.

Step 3

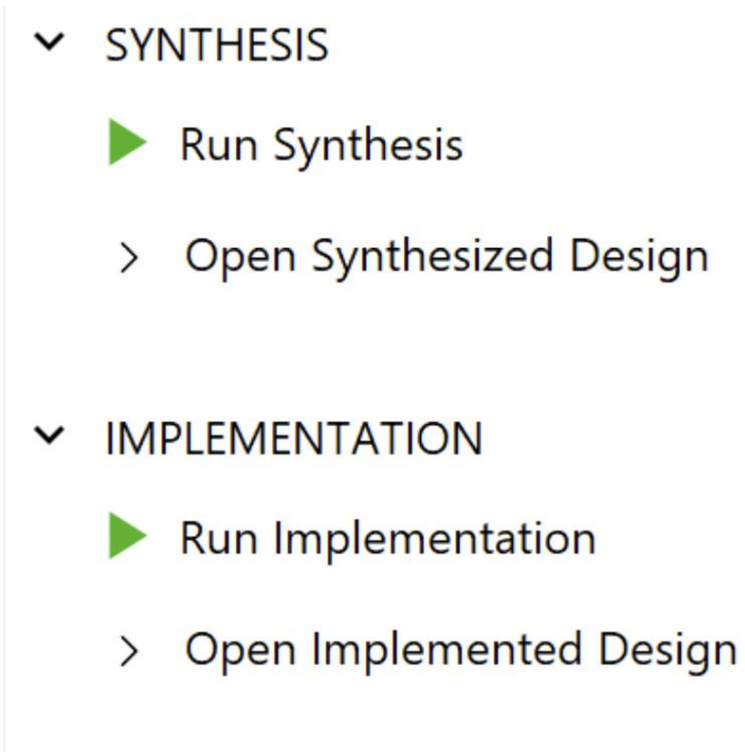
Explanation:

What Happens During Implementation

- Note: In the lab, this is not something we will be diving deep into
- Implementation is the process of transforming the synthesized gate-level netlist into a physical layout ready for fabrication. This involves floorplanning, placement, clock tree synthesis, routing, power planning, timing analysis, design rule checking, layout versus schematic checks, and chip integration.
- Each step is crucial for ensuring that the final chip meets performance, power, and area constraints and operates correctly in its intended application.
  (see back up slides for details on every step listed above)

Step 3
Explanation:

- In the lab, you will be clicking Run Synthesis -debugging if your code is wrong
- Then clicking Run Implementation



Step 4

Explanation:

- After the implementation phase, where the design is laid out and physically optimized for the target chip, the next step is to generate a bit file.
- This process is specific to programmable logic devices like FPGAs rather than ASICs, which have a fixed design once fabricated.
- The bit file is a configuration file that programs the FPGA to behave according to the implemented design.

Step 4

Explanation:

- The bitstream contains configuration bits that define the state of each programmable element within the FPGA. This includes setting the logic functions of LUTs, configuring flip-flops, and establishing interconnections.
- Bit File Structure
- Contains several key elements:
  -Header Information: Metadata about the bit file, such as the target FPGA model, bitstream version, and creation date.
  -Configuration Data: The binary data that configures the programmable logic blocks, routing resources, and I/O settings.
  -Checksum: Data integrity check to ensure the bit file is correctly transferred and programmed.

Step 4

Programming the FPGA:

- Once the bit file is generated, it can be downloaded onto the FPGA. This is done using a programming tool or hardware.
- The FPGA configuration process involves the following steps:
  -Loading the Bit File: The bit file is transferred to the FPGA programming interface.
  -Configuration: The FPGA reads the bit file and configures its internal resources accordingly.
  This process typically takes a few milliseconds to seconds, depending on the size and complexity of the design.
- Initialization: After configuration, the FPGA initializes and begins operating according to the programmed design.
  -Which would be your LEDs flashing in the way you coded them to

Steps 1-4
With different coding:

- Once you have completed the previous four steps, you will be redoing them by making changes to the code

First, change the first four LEDs so they are defined as follows:

- Make LED0 the logical AND of the first four switches.
- Make LED1 the logical OR of the first four switches.
- Make LED2 come on if ADC is below half the range (`assign led[2]=(ADC<12'h800);`
- Make LED3 come on if ADC is greater than half the range (inverse of LED2).
- Tie the first output of the JB connector (JB[0]) to the state of LED3; ie, assert it if the signal is more than half the range.

- And redoing the steps again but making use of a function generator which will give us variable waveforms and frequencies
- Then doing the steps with a 16 bit counter

FPGA LAB

Step 5
FPGA:

- The lab will be concluded by looking at how much of the chip's resources this module took

Backup

Step 3

Explanation:

Definitions

- Floorplanning: This step involves creating a rough plan of where each major functional block will be placed on the chip. The goal is to minimize the distance between interconnected blocks to reduce signal delay and power consumption.
- Placement: tool places each logic gate and flip-flop onto the chip. The placement tool aims to position components in a way that optimizes performance and meets timing constraints. Proper placement is crucial for minimizing signal delays and ensuring efficient routing.
- Clock Tree Synthesis (CTS): A well-designed clock distribution network ensures that the clock signal arrives at all parts of the chip simultaneously. This minimizes clock skew, which can cause timing errors and reduce performance CTS involves creating a balanced tree structure of clock buffers and interconnects to distribute the clock signal evenly.

Step 3

Explanation:

Definitions

- Routing: creating the physical connections between the placed components. The routing tool generates metal wire paths to connect inputs and outputs of logic gates, ensuring that all signals can travel from their source to their destination without conflicts. Routing also needs to consider signal integrity, avoiding issues like crosstalk and ensuring that signals remain clear and undistorted.
- Power Planning: designing a robust power distribution network that delivers stable voltage levels to all parts of the chip. This includes placing power and ground lines, adding decoupling capacitors, and ensuring that the power supply can handle the design's current requirements.

Step 3

Explanation:

Definitions

- Timing Analysis: Timing analysis checks that all signal paths meet the required timing constraints, such as setup and hold times. This ensures that the chip will operate correctly at the intended clock frequency. Tools like static timing analysis (STA) are used to verify that signals propagate within the allowed time frames.
- Design Rule Checking (DRC) and Layout Versus Schematic (LVS) Checks: DRC ensures that the physical layout complies with the manufacturing process rules, such as minimum spacing between wires and layer usage. LVS checks that the physical layout matches the original schematic design, ensuring that the design intent is preserved in the final layout.

Step 3

Explanation:

Definitions

- Chip Integration and I/O Planning: This final step involves integrating the design with other parts of the chip and planning the connections to the I/O pins. This includes ensuring that the design can communicate with external components and interfaces, such as memory, sensors, and communication ports.

The kit for this lab contains material for five complete lab stations, with the idea of setting of four stations and having one spare. The complete contents of the kit.

- 5 – Dell Latitude 5480
  - 8GB DDR4
  - 256GB SSD
  - Win 10 Pro
  - Minimum installation of Vivado IDE 2022.1
  - Two example Basys 3 projects
- 5 – Digilent Basys 3 development boards
- 5 – USB-A to micro-USB cables to connect development boards to computers.
- 5 – FNIRSI-1014D 2 Channel 100 MHz digital oscilloscope w/function generator. Each includes
  - 2 probes
  - 1 BNC to alligator clip adapter
  - 1 USB power adapter
  - 1 USB-A cable to connect to computer
- 6 – mini breadboards to wire circuits
- 1 – box of jumper cables for circuit wiring
- 12 – 10k trim pots, to use to set the DC level into the analog input.
- 1 – Hard disk with Vivado installation software, Basys 3 example projects, and README.txt instruction file.
  - To be used if the software needs to be re-installed on these or other computers.