

System architecture

Hardware architecture

The system is built around a **Nios II processor** and contains the following main components:

- **Nios II processor:** Main processor that runs the software.
- **On-chip Memory:** Used to store code and data.
- **JTAG UART:** Enables serial communication for debugging.
- **Timer:** Used to manage time management in the clock function.
- **PIO (Parallel I/O):** Handles inputs and outputs.
- **On-chip Flash:** Non-volatile memory for storage.
- **VGA IP:** Handles the display of the clock on the screen.
- **PLL (Phase-Locked Loop):** Handles clock signals.

Software architecture

The software is responsible for initiating the system and running a clock function.

Flow in the program:

1. **Initialization:** The system starts and displays the designer's name on the screen.
2. **Time update:**
 - Keeps track of time in seconds.
 - Calculates hours, minutes, and seconds.
 - Prints the time on the VGA screen.
3. **Delay:** A simple delay() function is used to wait 1 second between updates.

Booting process

1. The system will start when the power is turned on.
2. The FPGA loads the Nios II processor and starts the code from the On-chip Memory.
3. The program runs and starts displaying time.

See the next figure.

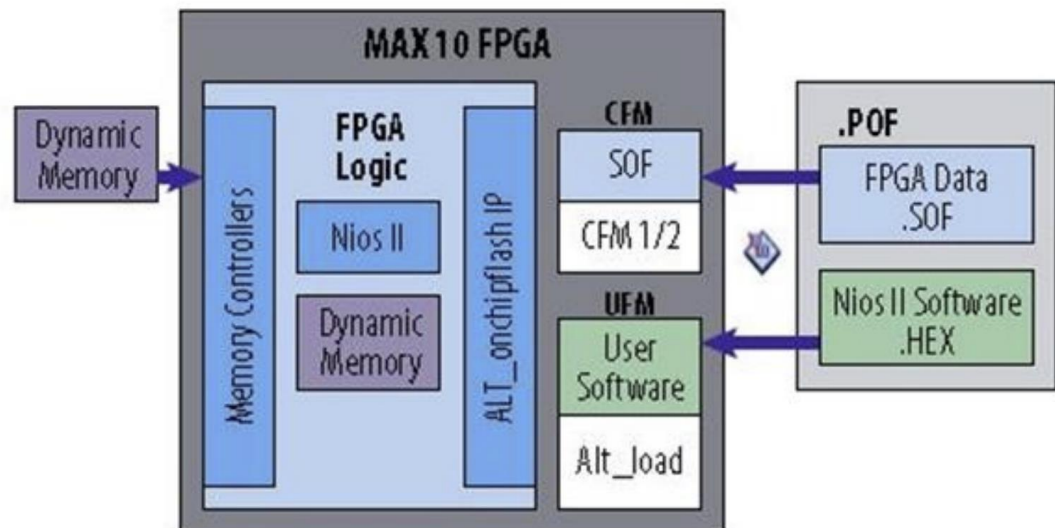


Figure 1- SW and HW architecture

How the boot was created

The booting into your system is the process that causes the system to boot automatically when the power is turned on. This process consists of several steps, from turning on the power until the application starts running. Here's a detailed description of how the boot was implemented:

1. Power-On Reset

When the power is turned on, the **MAX-10 FPGA** will enter a **reset state**. In this state, all hardware components are initialized, and the system prepares to load and run the program stored in the built-in **flash memory**.

- **Flash memory** is configured to contain two main parts:
 1. **Bootloader**: A small piece of program code that is responsible for initializing the system and loading the main application.
 2. **Application Code**: The main program to run after the system has been booted.

2. Bootloader Execution

After the reset state, the **bootloader will start** running. The bootloader is a small program that is stored in flash memory and has the following tasks:

1. **Initialize hardware components**:
 - The bootloader initiates all the necessary hardware components, such as:

- **PLL (Phase-Locked Loop):** Configures the system clock.
- **Timer:** Initiates the timer used for the clock function.
- **VGA IP:** Prepares the VGA monitor to display text.
- **JTAG UART:** Prepares communications for debugging.

2. Load the application code:

- The bootloader retrieves the main application from the flash memory and loads it into **the Nios II processor's memory (RAM)**.
- The application code contains the program to run, including functions to display the designer's name and implement the clock.

3. Transfer control to the application:

- Once the bootloader completes its tasks, it transfers control to the main application by jumping to the application's boot address.

3. Application execution

After the bootloader transfers control, the main application will start running. The application performs the following steps:

1. View welcome message:

- The application uses **VGA IP** to print a welcome message on the screen, including the designer's name (e.g., "Design: Menus Hees").

2. To start the clock function:

- The application uses **the Timer** to implement a clock that counts hours, minutes, and seconds.
- Every second, the time on the screen is updated using VGA IP.

3. Continuous operation:

- The system continues to run the clock function unlimitedly, and the time is updated every second.

4. Summary of the boot process

The boot process of your system can be summarized in the following steps:

1. Power-On Reset: The system starts and initializes the hardware.

2. Bootloader Execution:

- Initialize hardware components (PLL, Timer, VGA IP, etc.).
- Load the main application from the flash drive to RAM.

- Transfer control to the application.

3. **Application execution:**

- Display the welcome message on the screen.
- Start and update the clock function.

5. Technical details

- **Flash Memory Configuration:** Flash memory is partitioned into two parts:
 - **Bootloader section:** Stores the bootloader code.
 - **Application Section:** Stores the main application.
- **Nios II processor:** Responsible for running both the bootloader and the application.
- **PLL:** Generates a stable system clock that is used by all components.
- **VGA IP:** Used to print text on the screen.

6. Conclusion

The booting into your system is an important part that ensures that the system boots correctly and automatically when the power is turned on. By using a bootloader to initialize the hardware and load the application, you have created a robust and reliable system. The boot process is efficient and ensures that the application can run immediately after booting.

The size of HW and SW

The next figure shows the size of the SW/HW total in the POF file

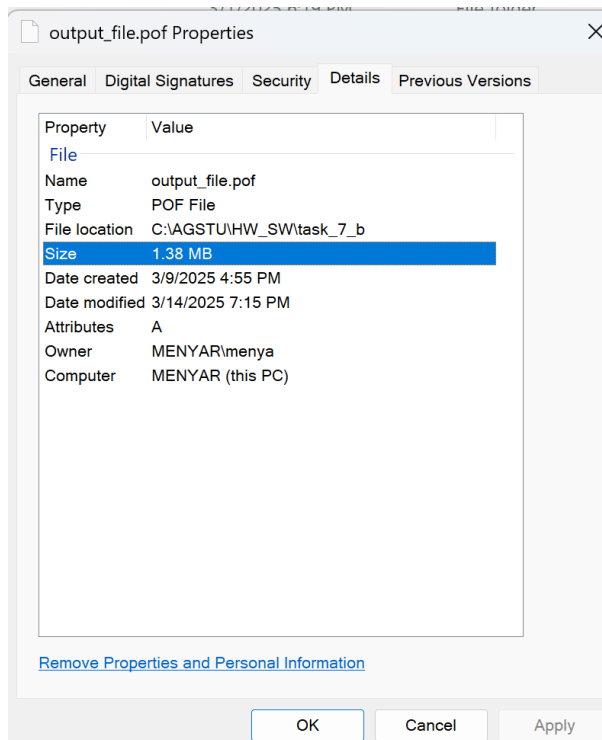


Figure 2 SW/HW Size

Flow Summary from **Quartus Prime** (version 22.1), which summarizes the results of a successful compilation of an FPGA project. See the next figure.

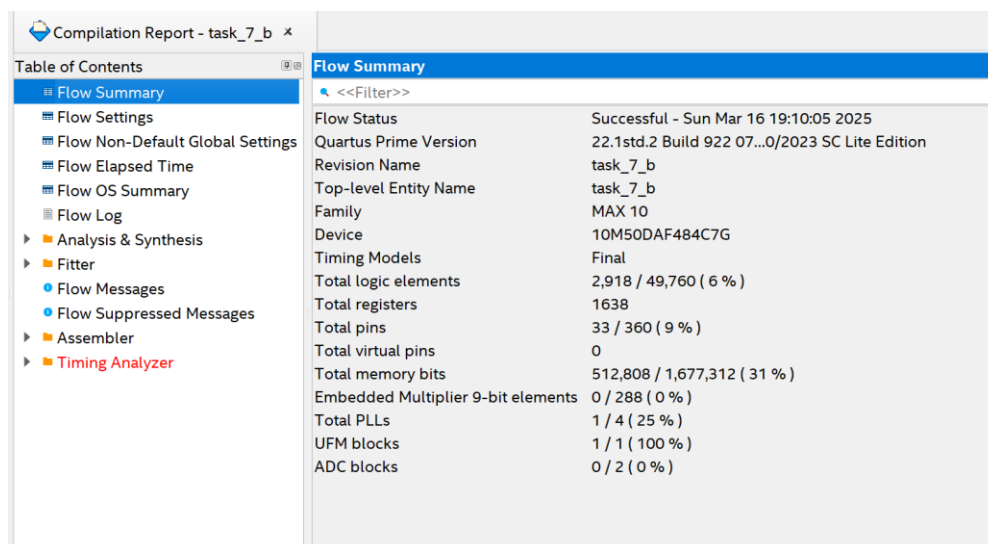


Figure 3 Flow Summary

Demonstration firemen

<https://www.youtube.com/watch?v=VM3odpAnjuE>