# Objective 13 FPGA programming

Field Programmable Gate Arrays (FPGA) are integrated circuits that can be programmed to execute specific functions after they are manufactured. They allow designers to create specialized integrated circuits quickly and efficiently.

You have probably completed the Frostavator terminal already, but we will discuss it here because it has hints for the FPGA challenge.

|  |  |
| --- | --- |
| A picture containing text  Description automatically generated | Text  Description automatically generated |

## Terminal Frostavator Logic Gates

Grody Goiterson needs help to get the elevator in Frost Tower running, and you need the elevator to reach other floors. Grody’s hint will show you how to interpret the icons in the Frostavator.

|  |  |
| --- | --- |
| A picture containing text  Description automatically generated | Text  Description automatically generated |
| A green screen with white text  Description automatically generated with medium confidence | <https://www.geeksforgeeks.org/introduction-of-logic-gates/> |

### Step 1 Question: Can you fix it?

The inputs to the Frostavator are constant. Inputs 2 and 4 are off, inputs 1 and 3 are on. Your task is to rearrange the logic gates so that Outputs 1, 2, and 3 are on. Then the Frostavator will power on and you can access the higher floors of Frost Tower.

|  |  |
| --- | --- |
| Graphical user interface, text, application  Description automatically generated | A screenshot of a computer  Description automatically generated with low confidence |

### Step 1 Answer

There are many possible combinations that will power the Frostavater. This is one of them. A screenshot of a video game

Description automatically generated

## Hints after solving the Frostavator

Grody has these hints for us after we fix his elevator.

|  |  |
| --- | --- |
| A green screen with white text  Description automatically generated with medium confidence  <https://www.youtube.com/watch?v=GFdG1PJ4QjA>  A green screen with white text  Description automatically generated with medium confidence  <https://www.fpga4fun.com/MusicBox.html> | Text  Description automatically generated |

## Objective 13 FPGA Programming

Program the FPGA to play tones at a randomly specified frequency so that Crunchy can call home with the communicator he has cobbled together.

### Step 1 question Program the Array

This challenge was quite new for me, so I do not have much advice to give. There are pointers, though.

1. Use the same concept as the code in Prof. Petabyte’s talk at 5:00 where rst and clk are in the same always@() statement. Then split them out using an If Then Else statement with rst and clk each in their own Begin-End block so that only one executes at a time. If they can execute simultaneously, you will be off by about 0.002 Hz on the 500 Hz, 1 kHz, and 2 kHz tests.
2. Use the divide by two/countdown technique that is at the end of the Music box 1 example in the “people do this for fun” link.
3. The frequency input is 100 times what it should be. It is an integer with no decimal point, but it has significant digits to the 1/100th place.
4. Do not forget to subtract 1 when you are converting the 125 MHz clock to the counter size. It is another way to cause the 0.002 Hz error.
5. I never got the rounding to work entirely correctly. It was close enough that it passed the test occasionally, which was enough to complete the achievement.
6. If you get a strange error, look for a missing semicolon the line before the error
7. And as Prof. says, “We are NOT writing programs,” where you have loops.

### Step 1 Answer

Here is the code that I used.

Graphical user interface, text

Description automatically generated

Once you have successfully programmed the FPGA, a chip will appear in your badge. If you put it into the device Crunchy has constructed (and all other objectives are solved) amazing things will happen.