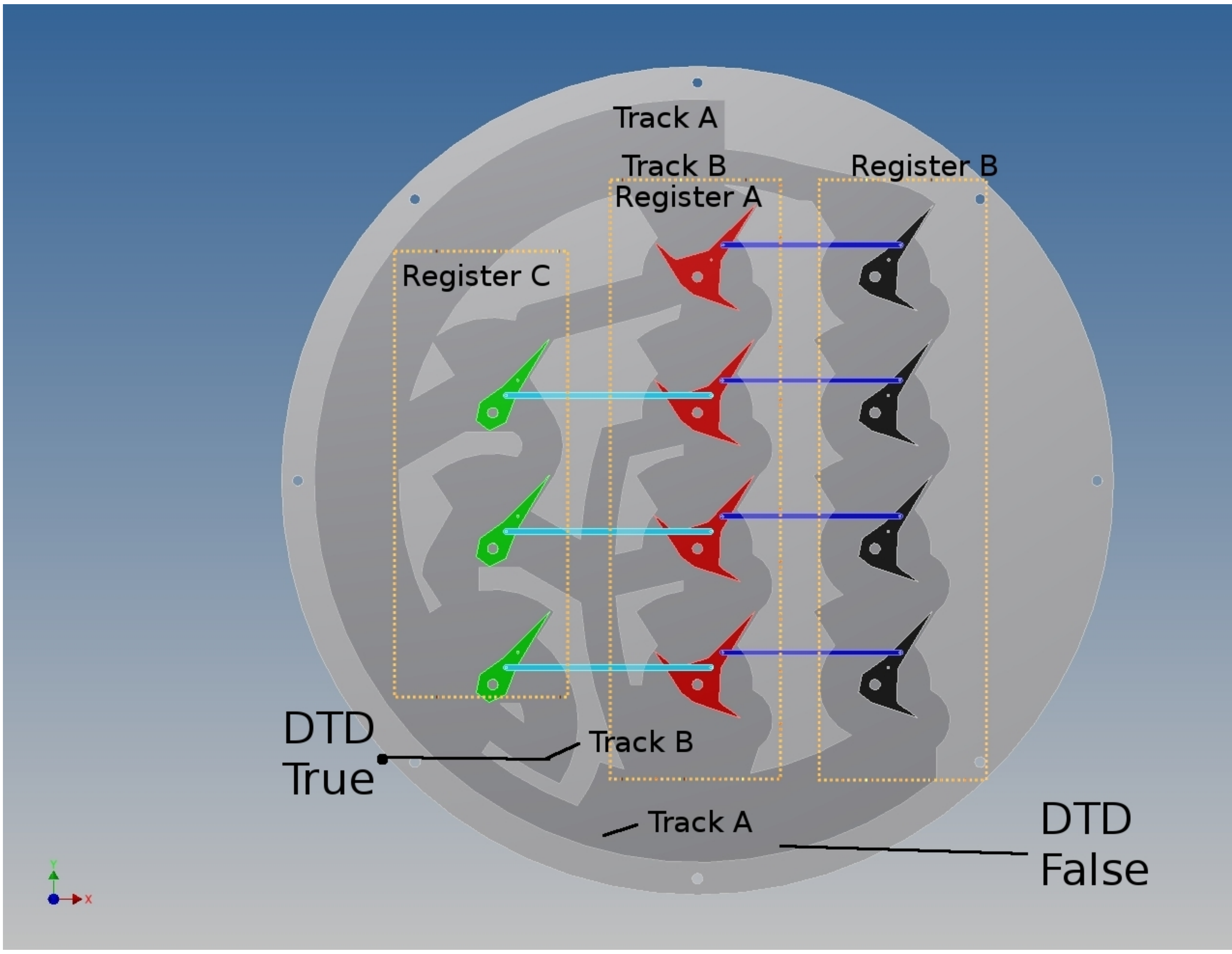
Mechanical CPU Clock



{youtube}0H4LTOYpAM4{/youtube}

Introduction

The Mechanical CPU Clock shows the basic building blocks of a CPU (ALU.Buses,RAM,registers, and a Control Unit). It executes a set of instructions which will emulate a simple wall clock.

The inspiration for this project came from trying to teach my son and daughter about how a computer works (in reality, I was always fascinated with mechanical computers and clocks, but I had to give a better excuse to my wife for buying a laser cutter specifically for this project). After looking around the web at various mechanical computers, I could not find something that represented all the components of CPU. However, I draw my inspiration from the following projects (and their derivatives):
Marble adding machine: www.youtube.com/watch?v=GcDshWmhF4A
DigicompII: http://digi-compii.com/
Ball Logic: http://brain.wireos.com/?p=2207

In the following sections I will attempt to explain how everything works as it is being built. I am not going to go deep into the subject of computer science and will purposely try to avoid some terms, so to not confuse the novice reader. If I do use any terms, I will try to explain them simply. However, there might be some needed background information that I will miss, so please do not hesitate to contact me about trying to explain the concepts in more details (I don't promise anything, but I will do my best). Even if you are not going to build the clock, going through the sections will help in the understanding of how the clock/CPU works (the build sections will go over the concepts of ALU,RAM,register,control unit and buses). Again, one of the motivations for this project was to get people to understand how a CPU (the heart of a computer) works.
More details about the clock can be found here: http://www.liorelazary.com/index.php?option=com_content&amp;view=article&amp;id=46:mechanical-cpu-clock&amp;catid=10:clocks&amp;Itemid=15

CPU instructions

For a CPU to do any meaningful work, it needs to be told what to do. This comes in the form of instructions. For this wall clock I will only concentrate on the hours to make things simple. That is, we will increment a variable named Hours and check to see if its equal to 11 (we will use 0 base indexing for the hours, so 12 is represented as a 0). If the check is true, then we will reset the variable back to 0. We do not want to code the above statement directly into the CPU, since we want it to be general purpose CPU (otherwise we are just making a clock). Therefore, we will implement a basic instruction set and write the clock code around these primitive instructions (the set of instructions to execute are known as an assembly language). Lastly, we will keep the hour variable in a Register, which is a term given for a special memory component used to hold data and operate on it. We will also use a simple 1 unit (1 bit) memory address to act as a flag for the control unit, and name it DTD for personal reasons (stands for dedicated to Dani. A long personal story). Note that this CPU will be programed with the clock instructions, but it could be programed with any other code to do a number of other things.

Here are the basic instructions (assembly language) that we will use:
INCRMENT: Increment register A by adding 1 to it.
EQUAL: If register A is equal to a specific number, then skip the next instruction.

CLEAR: set register A to 0, set DTD to false
JUMP: jump to a specific instruction at a given line number
SET_DTD: set DTD to true
CHECK_DTD: if DTD is true, then skip the next instruction.

Here is the assembly language code for the clock (register A will hold the hours).

- 1: CHECK_DTD
- 2: JUMP 5
- 3: CLEAR
- 4: JUMP 1
- 5: INCREMENT
- 6: EQUAL 11
- 7: JUMP 1
- 8: SET_DTD
- 9: JUMP 1

Line 1 checks to see if DTD is set to true (this is used to indicate if we need to reset the hour). If its true the we jump to line 3 and set the hour to 0 and the DTD to false. If DTD is false then we continue with the next instruction and jump to line 5. Line 5 increments the hour (register A), while line 6 check to see if it equal to 11. If register A is equal to 11, then we jump to line 8 and set the DTD to true and jump back to the beginning (line 1). Otherwise, we go to the beginning. If we run though this code once an hour, then by reading register A, we can tell what is the current hour.

Reading the time:

{youtube}iFKargQxN8k{/youtube}

The time (hour) is read in binary (http://en.wikipedia.org/wiki/Binary_numeral_system) from the middle register (register A, look for the highlight in the video where register A is at). Note the position of the 4 flip-flops (the 4 upside-down red T levers). If a lever is pointing to the right, then the digit is 0, if its on the left then its a 1. Write down the digits from each of the 4 flip-flops, from top to bottom. Then rewrite the number from left to right by rotating 90 degrees to the right, so that the left most digit is from the bottom and the right digit is from the top. For example. If the clock has the flips-flops at this position:

flip-flop 1 pointing left == 1

flip-flop 2 pointing left == 1

flip-flop 3 pointing left == 1

flip-flop 4 pointing right == 0

which is rewritten to: 0111 which is a binary representation of 7 decimal. So the time is 7 o'clock.

Another way is to:

- add 1 if the top dial is pointing to the left or nothing if its pointing to the right
- add 2 if the second dial is pointing to the left or nothing if its pointing to the right
- add 4 if the third dial is pointing to the left or nothing if its pointing to the right
- add 8 if the forth dial is pointing to the left or nothing if its pointing to the right

So the above example translates to: 1 + 2 + 4 + nothing = 7 decimal

Here is a simple conversion from binary to decimal (the number system that humans are used to). So if register A has

- 0000 is 0 but its 12 in the clock case
- 0001 is 1
- 0010 is 2
- 0011 is 3
- 0100 is 4
- 0101 is 5
- 0110 is 6
- 0111 is 7
- 1000 is 8
- 1001 is 9
- 1010 is 10
- 1011 is 11

In the video the clock goes through a full cycle, 0,1,2...11 and back to 0, so see if you can follow it.

The minutes can also be read from the position of the ball and the lever (a bit hard to see in the video). The numbers on the outside represent the minutes in HEX format. To convert the number to decimal (the number system we are used to) you take the first digit to the left multiply that by 16 and add the right digit.

Example: 32 in HEX = 3*16+2 = 50 in decimal

00 is 12, 05 is 5, 0A is 10, 0F is 15, 14 is 20, etc.

For more information see: Mechanical CPU Clock

Project Files