

The Wire-free Breadboard – A Feasibility Study on Digital Circuit

Ray-Shine Run, Jun-He Chang*, Ming-Chi Yen

Department of Electronics Engineering, National United University, Taiwan

Copyright©2016 by authors, all rights reserved. Authors agree that this article remains permanently open access under the terms of the Creative Commons Attribution License 4.0 international License.

Abstract For over forty years, the breadboard had been a useful tool for basic electronic experiments. However, even for an experienced user, a complicated wiring work could be a nightmare during the process of experiment. Not only it takes time for wiring, but also a non-logical error may arise whenever there is a loose contact between jump wire and the electrical terminals of the breadboard. That is to say, sometimes, you cannot ensure that the confirmed circuit works correctly even after a careful wiring work. In this paper, focused on digital circuits, we have designed and implemented a “wire-free breadboard” prototype. Based on the technology of FPGA, a circuit system, which being located on the *bottom* side of the breadboard, has been built to substitute the jump wires on the *top* side of the traditional Breadboard. To save the time of wiring work and to avoid the non-logical errors of electronic experiments are the main motives of this research.

Keywords Breadboard, Wiring Work, Jump Wire, FPGA

1. Introduction

The breadboard is a popular tool which commonly used in fundamental electronics laboratory. Like a living fossil, more than 40 years, its generic structure is never changed! How could it be? In my humble opinion, “solderless” is the reason for that. To avoid troublesome procedure of soldering in developing circuit, apparently, is the original intention of this great invention, and it does satisfy the desire of user. Not only the original function continues to be applied in general electronics experiments, breadboards are also being integrated into some training equipments. For example, a small piece of breadboard has been embedded into the microcontroller development board, so that the user can attach an IC on this breadboard, just a few jump wires, the functional availability of the microcontroller can be expanded. In short, there is reason to be believed that the

breadboard can still survive for another 10 years! However, is there absolutely nothing to improve on breadboard? The answer is negative. As we know, to implement a complicated circuit on breadboard, a tedious process of the wiring work is unavoidable. For example, it takes 4~6 hours for wiring a complex circuit as shown in Fig.1.

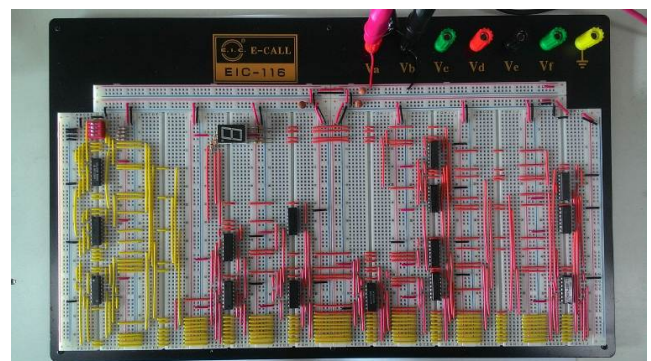


Figure 1. The wiring work on breadboard for a complex circuit

Moreover, a depressing non-logical error might await the tired mind. A popular but unpleasant experience is that you can not guarantee a well verified circuit to work exactly on breadboard, even with a careful wiring work. What I want to emphasize is that it may take another few hours to find out the fatal “loose contact” jump wire which ruined the hard effort. To make an improvement on the above deficiency, we have proposed a concept of “wire-free breadboard” which possesses the advantages below:

- A. *Save the time of the wiring work.*
A digital circuit system, which leveraged the FPGA as core technology, functionally substitutes the jump wires.
- B. *Avoid the non-logical errors.*
Since there is not a single jump wire left on the breadboard, no longer the “loose contact” errors will be happened.
- C. *Easy to Reproduce the previous well-verified circuitry.*

Since the connection relationship of components on the breadboard can be filed by computer, only a few steps (described later) to be followed, it is easy to reproduce the previous well-verified circuitry.

2. Methods

To construct the “wire-free breadboard”, there are three parts of work to be integrated, namely: *mechanical modification on breadboard*, *circuit system*, and *computer human interface*. The system diagram of the “wire-free breadboard” is shown in Fig.2

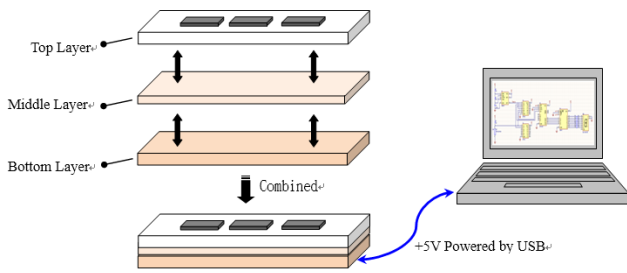


Figure 2. The system diagram of the “wire-free breadboard”

2.1. Mechanical Modification on Breadboard

Since all the jump wires are “missing”, the *electrical connections* between components on the breadboard must be done in another way. We have designed a “sandwich” structure (see Fig.3) to meet the requirements for the modified breadboard. The top layer is essentially made from the traditional breadboard with the *base* removed. Besides, to be linked with the middle layer, two parallelized *pin headers* has been carefully soldered on the *electrical terminals* of the breadboard (see Fig.4). The middle layer is a homemade PCB with female headers on both sides to make a bridge connection between top layer and bottom layer. The bottom layer is the circuit system, the kernel of “wire-free breadboard”, which will be described later.

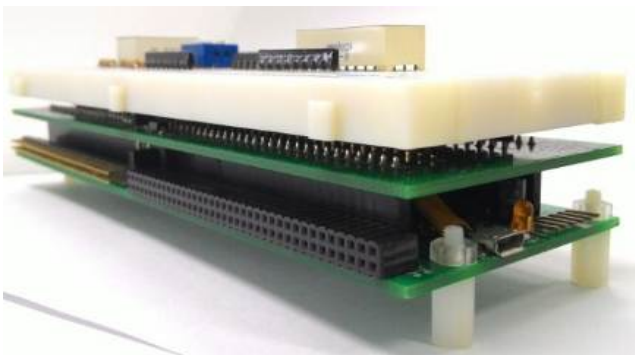


Figure 3. The “sandwich” structure for the modified breadboard



(Before)



(After)

Figure 4. The pin headers soldered on the electrical terminals of the breadboard

2.2. Circuit System

Basically, for a typical operation of digital circuit, the transmitted signal must be kept consistent (without distortion) all through the transmission path. In other words, more precisely, the signal voltage level on both ends of the transmission medium must remain unchanged as far as possible. For a practical application of typical breadboard circuit, the jump wire is an excellent option to meet the above requirement. Obviously, we need something different to be a replacement of jump wire. In general, the Analog Switch (see Fig.5) is also a reliable selection for signal transmission. However, it is not practical to build the “wire-free breadboard” with the Analog Switch. Why? For example, the CD4066B is a typical Analog Switch ic which contains 4 sets of bilateral switch, which means it will take 25 pieces of CD4066B for 100 signal transmissions, not including the related circuits for controlling all the bilateral switches. So, it is difficult to realize the required circuit system with Analog Switch at present.



Figure 5. The Analog Switch for *voltage* signal transmission

However, from another perspective, all the problems may be solved if we just focus on *digital* application. As we know, the *logic* level rather than the *voltage* level is concerned in digital system, so the *logic buffer* (see Fig.6) can be suitable solution for the requirement of signal transmission. In my opinion, if the system clock is lower than 20 MHz the logic buffer is an even better option than the jump wire, since it not only transmits the signal with the correct *logic* level, but also restores the signal to the rated *voltage* level. There are many logic buffers (ICs) can be appropriate substitute for the jump wire, however, the FPGA seems the most suitable one which meets the requirement perfectly!

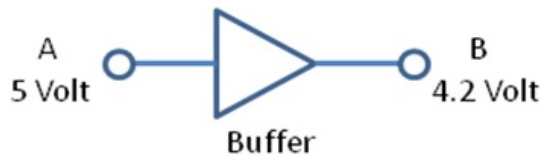


Figure 6. The Buffer for *logic* signal transmission

Looking into the structure of FPGA (see Fig.7), it is like a magic *mini* breadboard on which hundreds of logic components (see Fig.7, CLB) being arranged in a matrix form. More importantly, each CLB is surrounded with four Programmable Switching Matrix (see Fig.8, PSM) which are responsible for signal transmissions between CLBs. The PSM is just like the *tiny* jump wire inside the FPGA, obviously, it is the key to fulfillment of the “wire-free breadboard”!

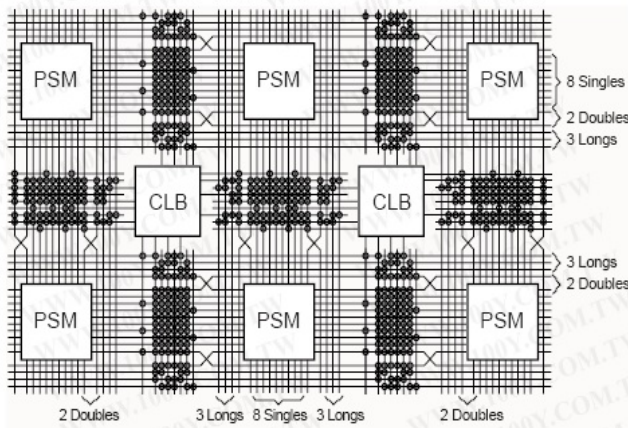


Figure 7. The matrix structure of the FPGA(Xilinx)

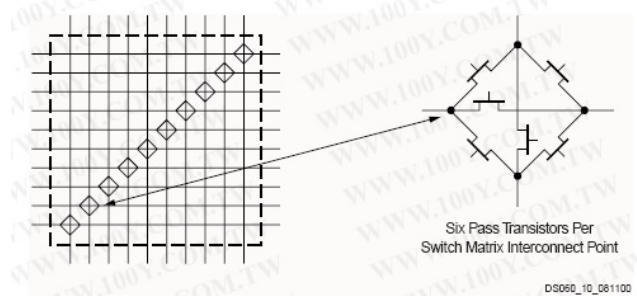


Figure 8. The structure of Programmable Switching Matrix (PSM)

The routing path of signal for the “wire-free breadboard” is shown in Fig.9. Instead of being used as a component for typical logic function, the FPGA simply works as a medium for digital signal transmission. “Receiving the signal from one pin, Sending the *same* signal to another pin” is the only job for FPGA in this research. So, the propagation delay of the signal transmission is quite short. For example, the “PAD to PAD” delay of Spartan-3AN family (XC3S50AN,TQG144,-5) is around 6 ns, which is qualified for the requirement of system clock (20MHz) as mentioned previously.

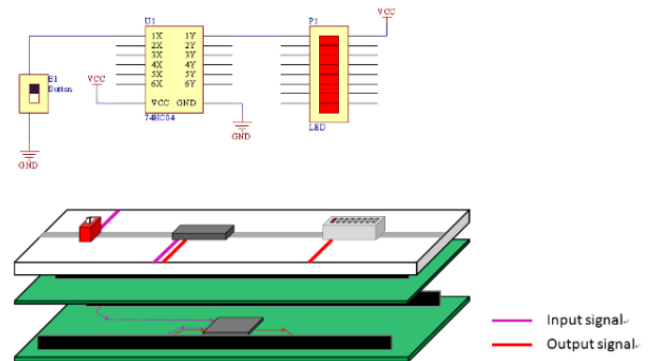


Figure 9. The routing path of signal for the “wire-free breadboard”

2.3. Computer Human Interface

Although the FPGA is the key component to build the “wire-free breadboard”, for users who simply want to make fundamental experiments on breadboard, it is not necessary to understand the FPGA detailly. However, it is definitely a highly professional work to use the FPGA as a logic component. So, a friendly computer human interface is needed before the jump wire is replaced by the FPGA. The flow chart of the human interface is shown Fig.10., and the details described below.

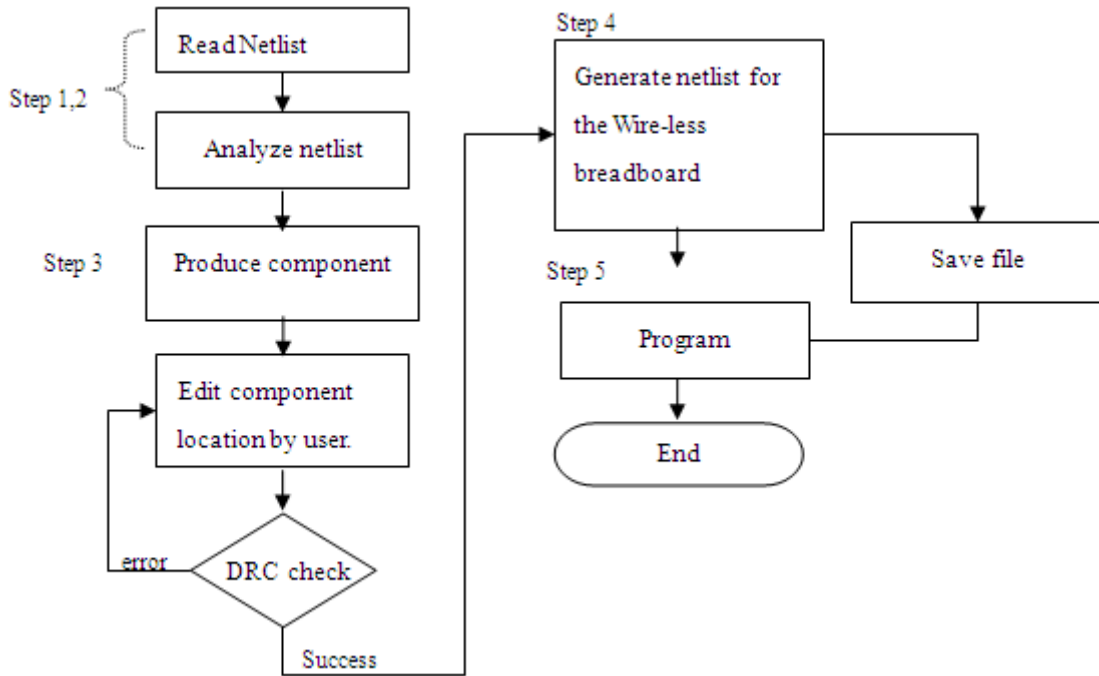


Figure 10. The flow chart of the computer human interface for the target system

2.3.1. Drawing a Schematic (step-1)

Although there are many tools to draw a “schematic-like” circuit diagram, a professional software (such as OrCAD, Protel, etc.) is needed to get a *regular* schematic for the first step. A “3 to 8 address decoder” schematic is shown Fig.11.

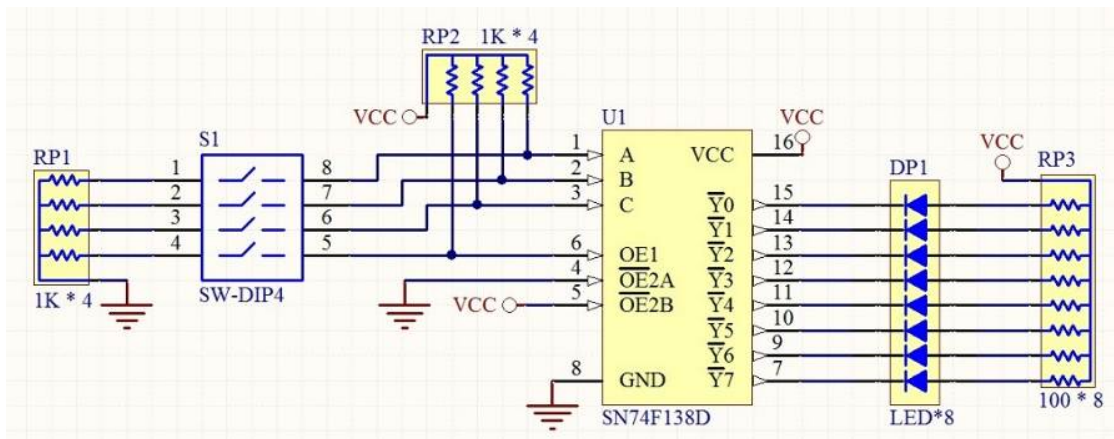


Figure 11. The schematic for a “3 to 8 address decoder”

2.3.2. Generating the Netlist (step-2)

A schematic is an *image* read by *eyes*, however, a netlist is *text* file read by *software*. In short, the netlist describes the connection relationship of components on the schematic. So, the schematic must be able to be transferred to the netlist to meet the follow-up procedures. Fig.12 shows the netlist of the schematic as above.

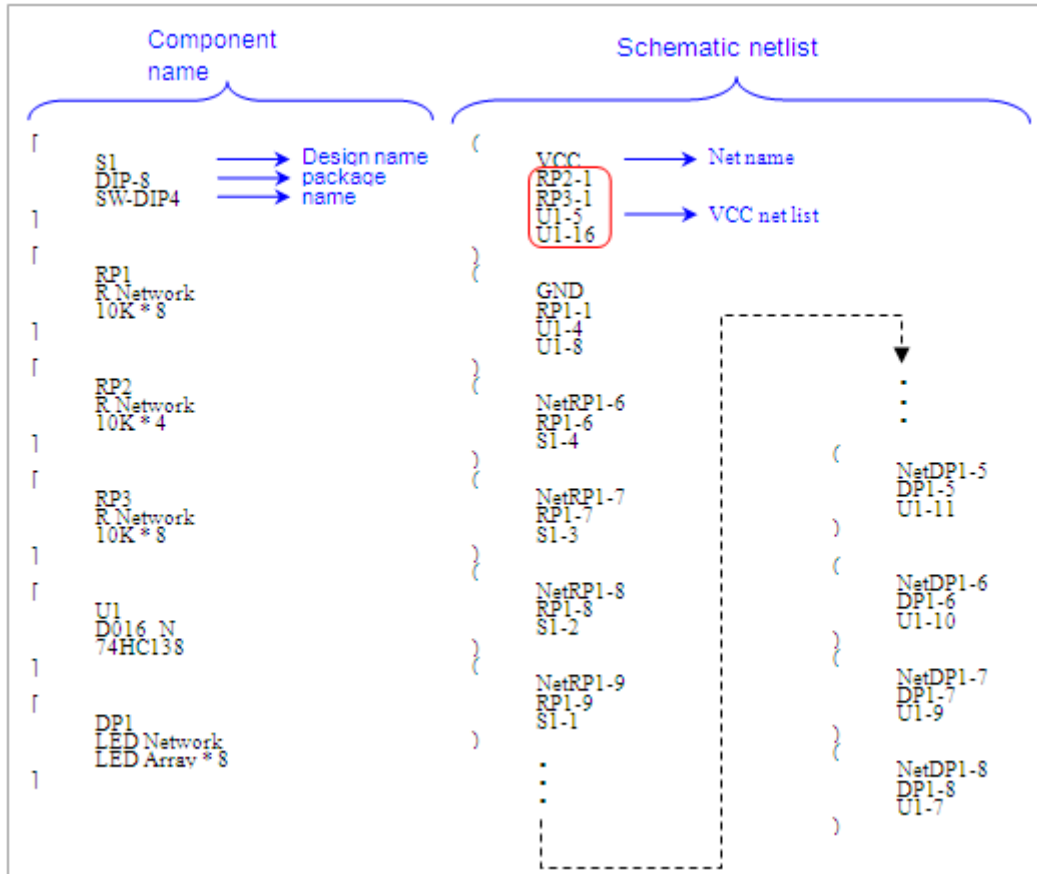


Figure 12. The netlist for the above “3 to 8 address decoder” schematic

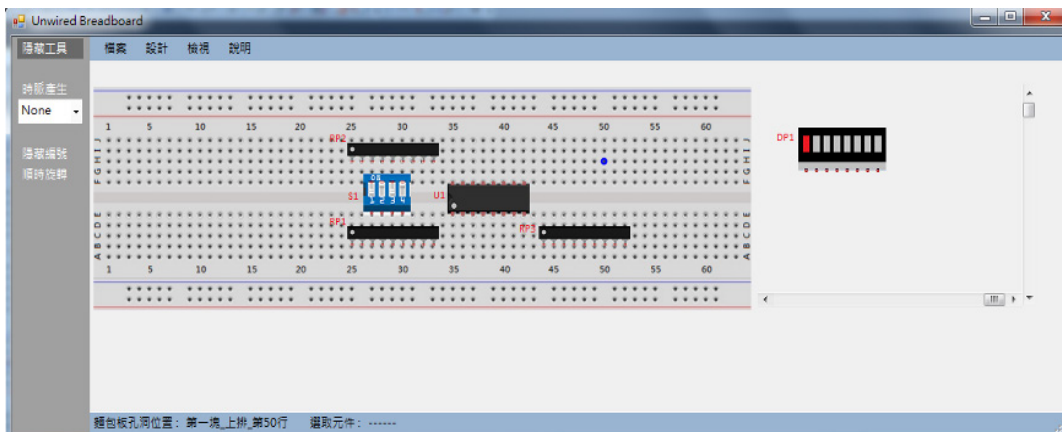


Figure 13. The homemade software for component placement on a pseudo breadboard

2.3.3. Component Placement (step-3)

A homemade window based software (see Fig.13) was built to help the user to execute the component placement on a *pseudo* breadboard which is displayed on the computer screen. Also, the connection relationship of the components on the *pseudo* breadboard can be filed by the software. Basically, there is no learning barrier to operate the homemade software even for beginners. However, the user must place the *real* components on the “wire-free breadboard” exactly according to the map (the connection relationship) of the *pseudo* breadboard.

2.3.4. Design the VHDL for the FPGA (step-4)

The VHDL is a *hardware description language* for digital circuit design, and mostly used for designs with the PLD components (ex. FPGA). At present, other than VHDL, Verilog is also common for engineers. No matter what kind of above, not a short period of time is required for learning. However, for this specific application, the functional requirement of the FPGA is as simple as I/O pin *rearrangement*. In other words, we only need a VHDL program to describes the *signal assignment*, not the *logic functions*! At this stage, the VHDL program is simplified as

a “form filling” work for users. Although it fulfilled the requirement of this project, it is still too difficult for beginners. The VHDL program should be integrated (hidden) to a more friendly human interface in the near

future. The following shows the user constraint file (Fig.14-1) and the VHDL (Fig.14-2) for the “3 to 8 address decoder”.

```

NET Wire_In_A<0>      LOC = P94;      #↑ 27      FPGA Pin-94(input), connect to switch Pin 1
NET Wire_In_A<1>      LOC = P93;      #↑ 28
NET Wire_In_A<2>      LOC = P92;      #↑ 29
NET Wire_Out_A<0>     LOC = P19;      #↓ 35      FPGA Pin-19(output) , connect to 74HC138 Pin-1(A0)
NET Wire_Out_A<1>     LOC = P20;      #↓ 36
NET Wire_Out_A<2>     LOC = P21;      #↓ 37
NET Wire_In_E        LOC = P91;      #↑ 30
NET Wire_Out_E<1>     LOC = P22;      #↓ 38
NET Wire_Out_E<2>     LOC = P23;      #↓ 39
NET Wire_Out_E<3>     LOC = P24;      #↓ 40
NET Wire_In_Y<0>      LOC = P82;      #↑ 36      FPGA Pin-82(input) , connect to 74HC138 Pin-15(Y0)
NET Wire_In_Y<1>      LOC = P81;      #↑ 37
NET Wire_In_Y<2>      LOC = P80;      #↑ 38
NET Wire_In_Y<3>      LOC = P79;      #↑ 39
NET Wire_In_Y<4>      LOC = P78;      #↑ 40
NET Wire_In_Y<5>      LOC = P77;      #↑ 41
NET Wire_In_Y<6>      LOC = P76;      #↑ 42
NET Wire_In_Y<7>      LOC = P25;      #↓ 41
NET Wire_Out_Y<0>     LOC = P71;      #↑ 45      FPGA Pin-71(output) , connect to LED-Bar
NET Wire_Out_Y<1>     LOC = P70;      #↑ 46
NET Wire_Out_Y<2>     LOC = P69;      #↑ 47
NET Wire_Out_Y<3>     LOC = P68;      #↑ 48
NET Wire_Out_Y<4>     LOC = P66;      #↑ 49
NET Wire_Out_Y<5>     LOC = P64;      #↑ 50
NET Wire_Out_Y<6>     LOC = P61;      #↑ 51
NET Wire_Out_Y<7>     LOC = P60;      #↑ 52
NET PWR_138          LOC = P83;      #↑ 35      FPGA Pin-83(output) , connect to 74HC138 VCC Pin
NET PWR_PR           LOC = P28;      #↓ 44
NET GND_138          LOC = P26;      #↓ 42      FPGA Pin-26(output) , connect to 74HC138 GND Pin
NET GND_DIP          LOC = P7;       #↓ 25
NET PWR_DIP          LOC = P96;      #↑ 25

```

Figure 14-1. The format of the user constraint file

```

-- X138.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity X138 is
port
(
  Wire_In_A : in   STD_LOGIC_VECTOR(2 downto 0);    --Set I/O Port
  Wire_Out_A : out  STD_LOGIC_VECTOR(2 downto 0);
  Wire_In_Y : in   STD_LOGIC_VECTOR(7 downto 0);
  Wire_Out_Y : out  STD_LOGIC_VECTOR(7 downto 0);
  Wire_In_E : in   STD_LOGIC;
  Wire_Out_E : out  STD_LOGIC_VECTOR(3 downto 1);
  PWR_138 : out   STD_LOGIC;
  PWR_PR : out   STD_LOGIC;
  GND_138 : out   STD_LOGIC;
  GND_DIP : out   STD_LOGIC;
  PWR_DIP : out   STD_LOGIC;
end X138;

architecture X138_ARCH of X138 is
begin
  Wire_Out_A <= Wire_In_A;    --signal assign
  Wire_Out_Y <= Wire_In_Y;
  Wire_Out_E(1) <= Wire_In_E;
  Wire_Out_E(2) <= '0';    --Output Logic 0.
  Wire_Out_E(3) <= '1';    --Output Logic 1
  PWR_138 <= '1';
  PWR_PR <= '1';
  GND_138 <= '0';
  GND_DIP <= '0';
  PWR_DIP <= '1';
end X138_ARCH;

```

Figure 14-2. The VHDL for the “3 to 8 address decoder”

2.3.5 programming the FPGA (step-5)

It is not difficult to get a free-licensed FPGA development software, for example, the Xilinx ISE Design Suite. However, a debug tool (ex. Xilinx's Platform Cable USB) is still needed for programming the FPGA device. Fig.15 shows the scenario of programming the FPGA which embedded in the "wire-free breadboard".

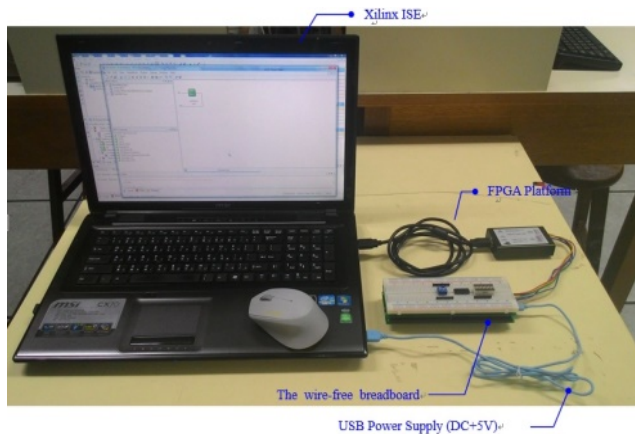


Figure 15. The scenario of operation for the "wire-free breadboard" system

3. Results

3.1. The Prototype

The pictures in Fig.16 show the outputs (led bar) of "3 to 8 address decoder" respond to different inputs (dip-switch). The point is not to verify the correctness of the logic design, certainly it is right, but to prove the feasibility of a *wire-free* breadboard.

There is a dynamic demonstration post on youtube video, which is an example of application of the 8051 mcu.

<https://www.youtube.com/watch?v=Kmilldw-e9U&feature=youtu.be>

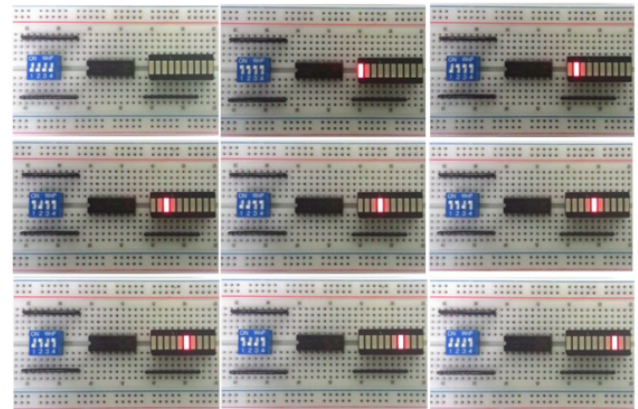


Figure 16. The corresponding outputs of "3 to 8 address decoder" for different inputs

4. Conclusions

It may be a bit of trivial demonstration of digital circuit but, with no doubt, it is a provement of the feasibility of "wire-free breadboard". There is something needs to be done before the "wire-free breadboard" can be used easily. The below is we want to do in the near future:

- A. a more user-friendly human interface for beginners.
- B. a bigger wire-free breadboard for test of complicated circuits.

REFERENCES

- [1] Wikipedia, <http://en.wikipedia.org/wiki/Breadboard>.
- [2] Parallax, <http://www.parallax.com/catalog/boards/basic-stamp>.
- [3] Arduino, <http://arduino.cc/en/Main/ArduinoStarterKit#.UxHrXtum024>.
- [4] Xilinx, <http://www.xilinx.com/products.html>.
- [5] SVF file and XSVF file format for Xilinx Device, http://www.xilinx.com/support/documentation/application_notes/xapp503.pdf