

Optimized Superconducting Circuit (2020 summer 580 project)

● Introduction

The goal of this project is aim to optimize a SFQ circuit by implementing security design.

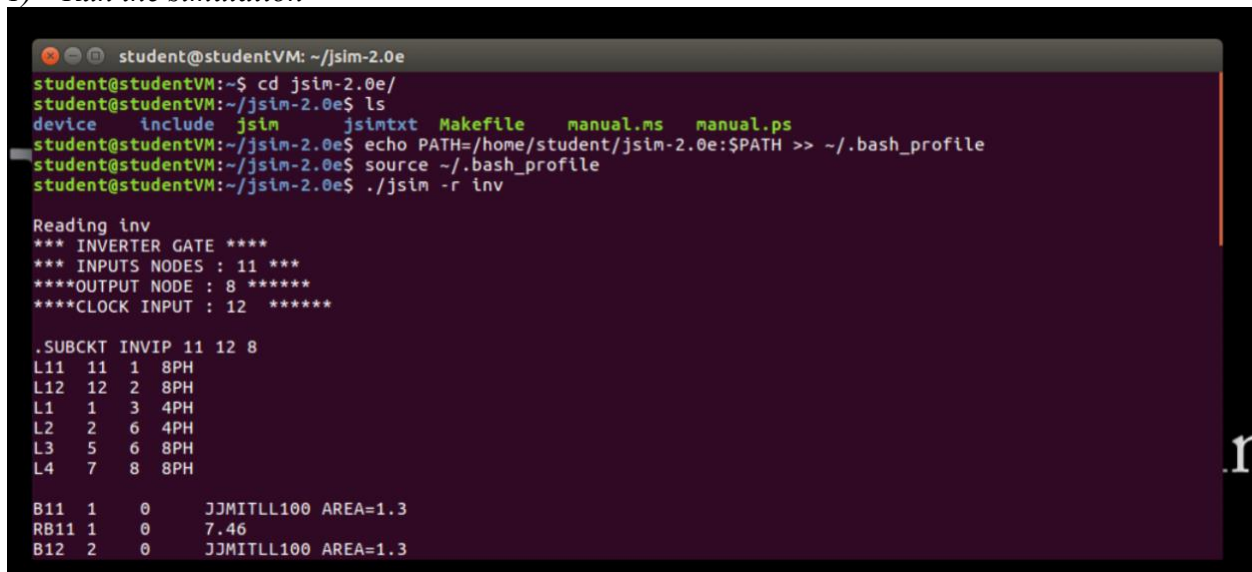
● Simulator

There are three sub-folders: *circuit_example*, *jsim-2.0e* and *matlab_script*.

- 1) *circuit_example* includes AND, OR, INV, DFF, JTL, XOR and FA circuit example (SFQ logic)
- 2) *jsim-2.0e* is the simulator for this project. (Linux version)
- 3) *Matlab_script* would be used for waveform plot after the simulation in MATLAB

● User manual

- 1) *Run the simulation*



```
student@studentVM: ~/jsim-2.0e
student@studentVM:~$ cd jsim-2.0e/
student@studentVM:~/jsim-2.0e$ ls
device  include  jsim      jsimtxt  Makefile  manual.ms  manual.ps
student@studentVM:~/jsim-2.0e$ echo PATH=/home/student/jsim-2.0e:$PATH >> ~/.bash_profile
student@studentVM:~/jsim-2.0e$ source ~/.bash_profile
student@studentVM:~/jsim-2.0e$ ./jsim -r inv

Reading inv
*** INVERTER GATE ****
*** INPUTS NODES : 11 ***
****OUTPUT NODE : 8 *****
****CLOCK INPUT : 12 *****

.SUBCKT INVIP 11 12 8
L11 11 1 8PH
L12 12 2 8PH
L1 1 3 4PH
L2 2 6 4PH
L3 5 6 8PH
L4 7 8 8PH

B11 1 0 JJMITLL100 AREA=1.3
RB11 1 0 7.46
B12 2 0 JJMITLL100 AREA=1.3
```

- i. Get into the directory of *jsim-2.0e*, which contain the *Makefile* file.
 - ii. Enter the command `echo PATH=<jsim path>:$PATH >> ~/.bash_profile`
 - iii. Source the file with command: `source ~/.bash_profile`
 - iv. Run the circuit file: `./jsim -r <circuit file>`
 - v. Get a *jsim.raw* file in the *jsim-2.0e* directory.
- 2) *Plot the waveform*
 - i. Move the *jsim.raw* file to the same directory with MATLAB scripts.
 - ii. Run the *PlotVoltage.m* file to get the waveform.

There are some syntax error in the *PlotCurrent.m*, you could correct it if needed.

● Example

Songyu, another student who built a Full adder circuit in SFQ logic. You could also check it in the circuit folder

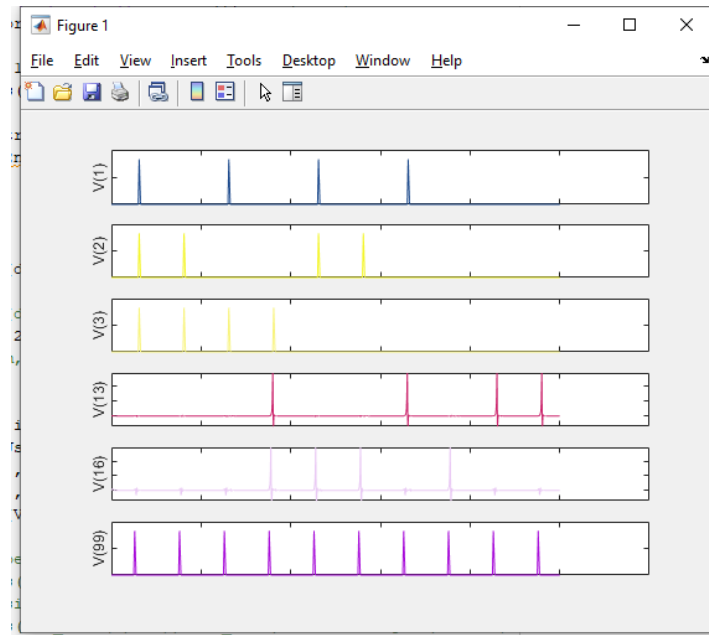


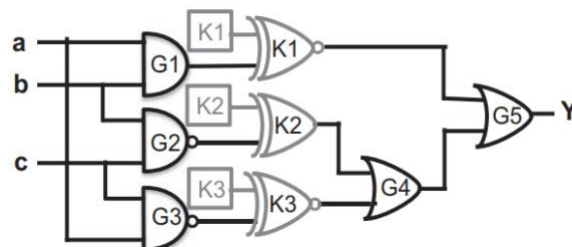
Figure 1, Waveform for FA

Simulation results for his full adder:

- i. V1, V2, V3 is the input
- ii. V13, V16 is the output
- iii. V99 is the clock.

He also provided a example of RSFQ circuit with security design. The security part is referred from paper:

SAT Attack Resistant Logic Locking, state Obfuscation-Based, Toward Increasing the Difficulty of Reverse



We could only get the correct function of circuit by providing correct key input (110).

SAT attack resistant circuit was added to mislead reverse engineer. As shown below, when input has the same value with K, output will be masked. The reverse engineer cannot detect circuit by using his algorithm.

locking techniques are to the SAT attack.

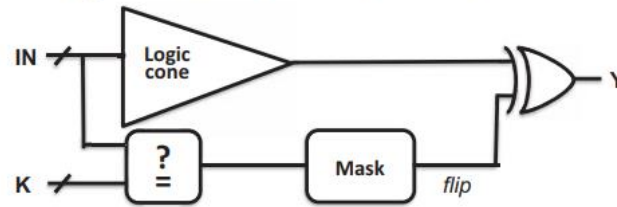


Fig. 5: SAT attack resistant circuit. The *flip* signal is asserted upon a match between an input value and a key value.

Our circuit with security design example is FA_as

- i. Input: V1, V2, V3.
- ii. Original Output: V13, V16
- iii. Output after the security circuit: V61, V62
- iv. Input K for SAT attack resistant circuit: V71, V72
- v. Input K for logic locking circuit: V81, V82, V83 (key is 110)
- vi. Clock: V99

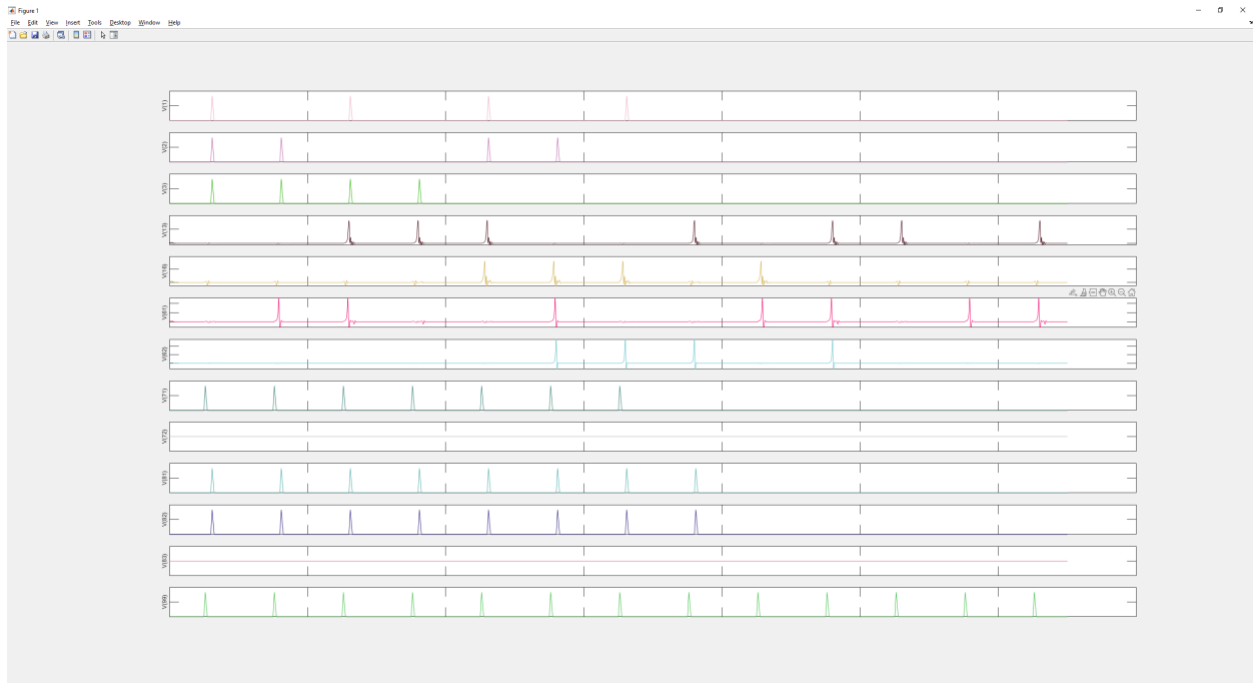


Figure 2, Waveform for FA_as