EI331 Course Project

By: Wang Haoxuan

Instructor: Wu ChenTao

November 28, 2018

I. PROJECT 6: BANKER'S ALGORITHM

This project aims to solve the banker's algorithm. The algorithm would determine whether the system is in a safe state, and if it is, the algorithm would provide an order for the processes to run in a safe manner. Four kinds of data is needed for the implementation of the algorithm: *available*, *maximum*, *allocation* and *need*. The available data id provided by the user through the terminal, and the initial value of the allocation is 0. Also, we have the knowledge that $need = maximum - allocation$, and the system is safe for a certain process if and only if every needed resource of the process is enough in the available queue. By these conditions, we can form the basic relationships between these variables, and the algorithm's structure can be formed easily.

The testing of the algorithm is shown in the figure below. There are four processes and four kinds of resources. We first run the code and input four numbers as the value for the available queue. *RQ* command is interpreted as a process requiring resources from the availabe and *RL* command is realeasing part or all of the allocated resources of a process. The *maximum* value of the processes are stored in a file named *maximum.txt*. We first let process 0 request resources of 3,1,2,1. Which is safe, and the request would be successful. Then we make process 0 release the resources of amount 3,1,2,1, which is safe too based on the previous procedure. The third command requires process 1 to ask for resource 10,10,10,10, but the requiring amount is too large and is over the maximum value needed, thus it is unsafe and the operation is ignored. The fourth command wants to release resource amount of 10,0,0,0 from process 2, but since process 2 is assigned no allocation, this operation is also regarded as unsafe. We also tried to request resource of 5,6,7,5 from process 4. This is unsafe because though it satisfies the maximum bound, it required more than the available resources. At last, we can type in $*$ to quit the program.

```
jerrywang@ubuntu:~/Documents/EI338/ch8$ ./main 10 5 7 8
RQ 0 3 1 2 1
Request successful!
RL 0 3 1 2 1
Customer 0 has released resource amount 3, 1, 2, 1 seperately
RQ 1 10 10 10 10
Unsafe! Need < Request
Request unsuccessful! This operation is ignored.
RL 2 10 0 0 0
Cannot release resources of this amount!
Customer 2 only has resource amount 0, 0, 0, 0
RQ 4 5 6 7 5
Unsafe! Available < Request
Request unsuccessful! This operation is ignored.
*
Process terminated
```

FIG. 1: A screenshot of the RTL Schematics produced from the Verilog code.

Code is shown in the *code* file.