REPORT

|  |
| --- |
| DCS HW3 |
| Student: 111511076 陳彥宇 |

# State Graph

We first read the input in the IDLE state then determine whether we can sell the products or if restock is needed. The state graph then diverges into 2 sections: PROCESS and REFILL. PROCESS lasts for 1 clk cycle and outputs the amount of money the customer should pay. REFILL does what its name suggests.

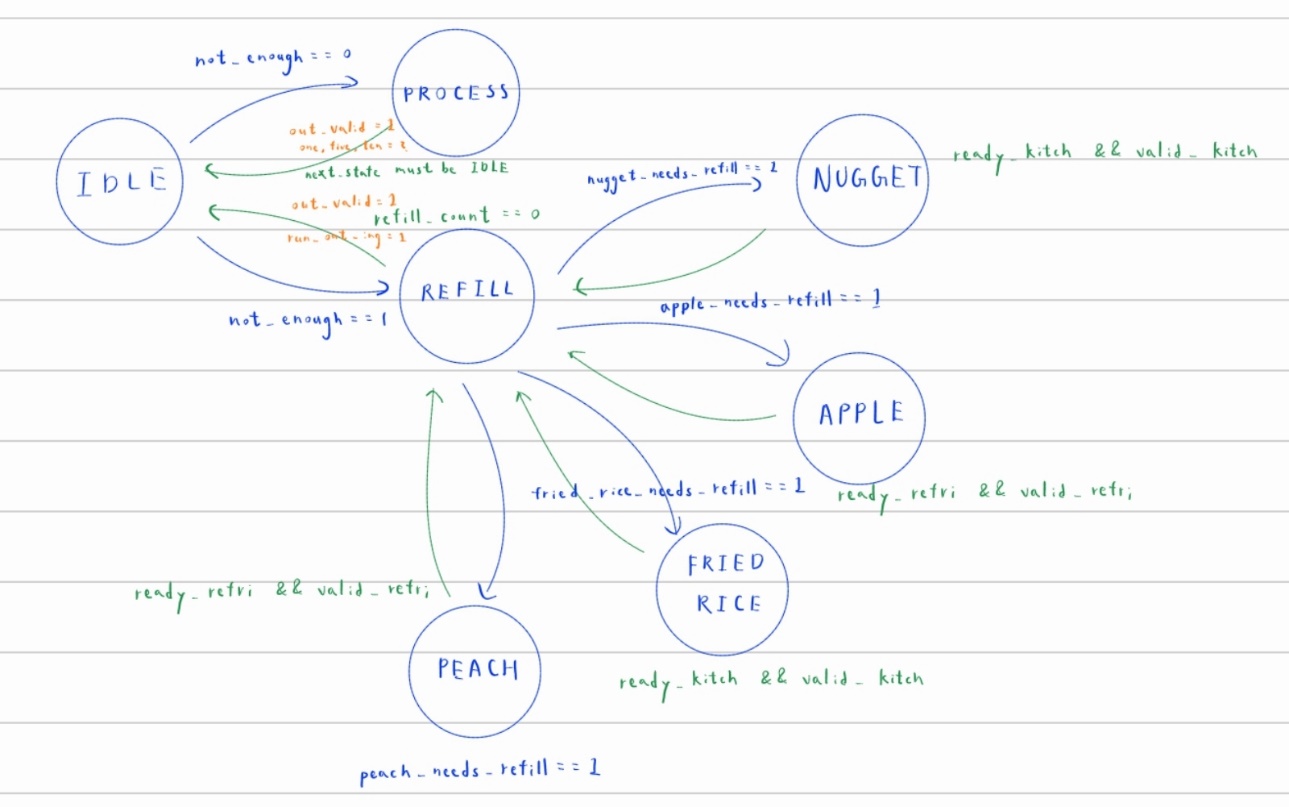
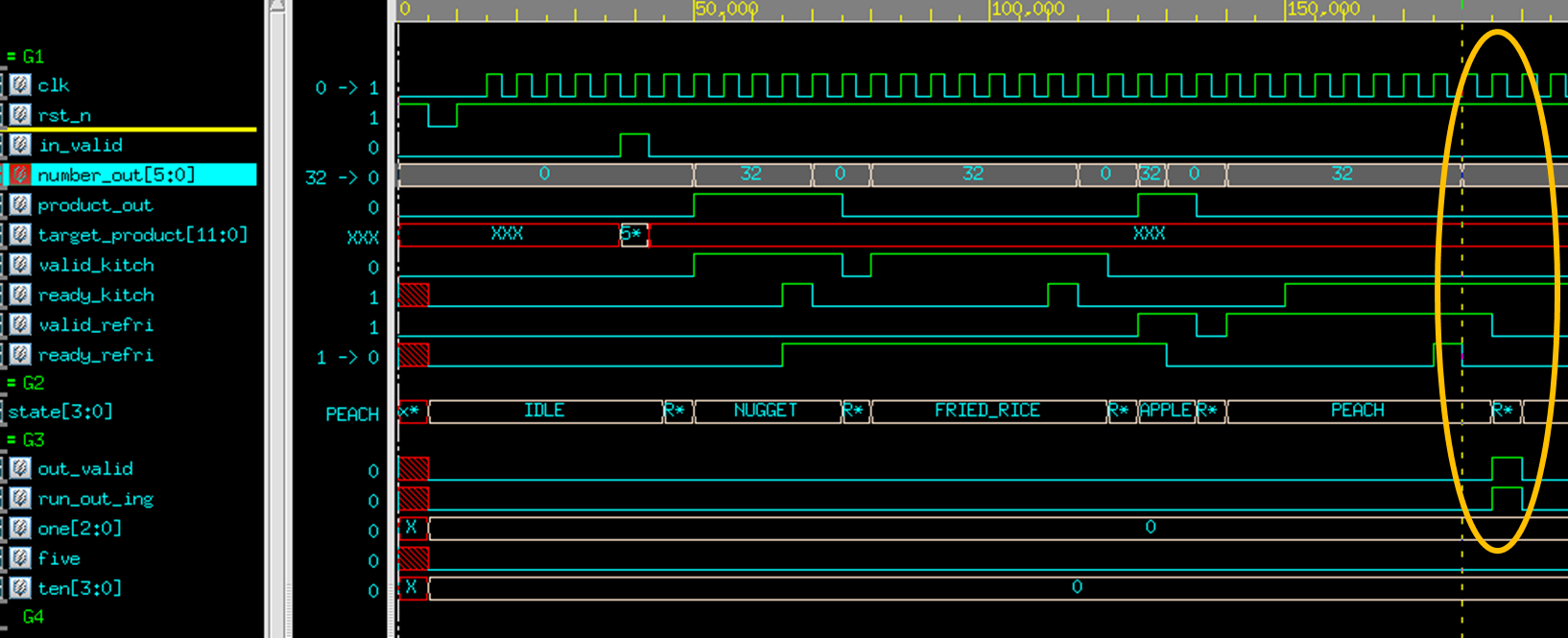


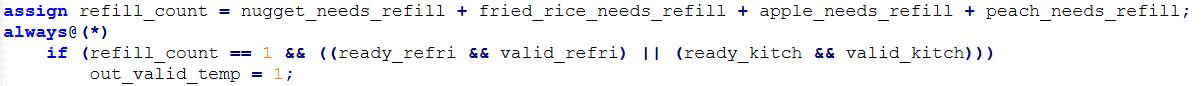
Fig. 1 Design State Graph

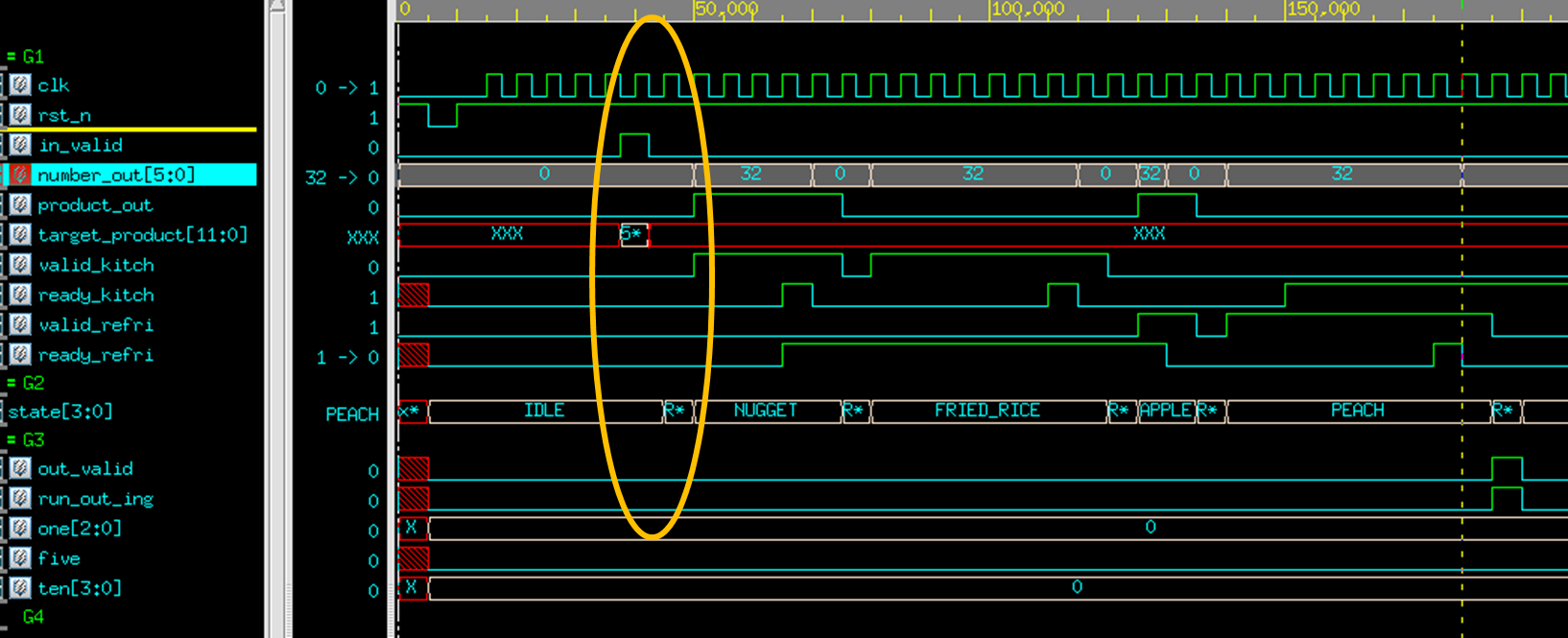
# Area & Timing Optimization:

I believe everyone’s area is around 10000 this time, so there isn’t really a point to delve too much into minimizing the area. However, there’s much to be done regarding latency. I found 3 ways to shorten it.

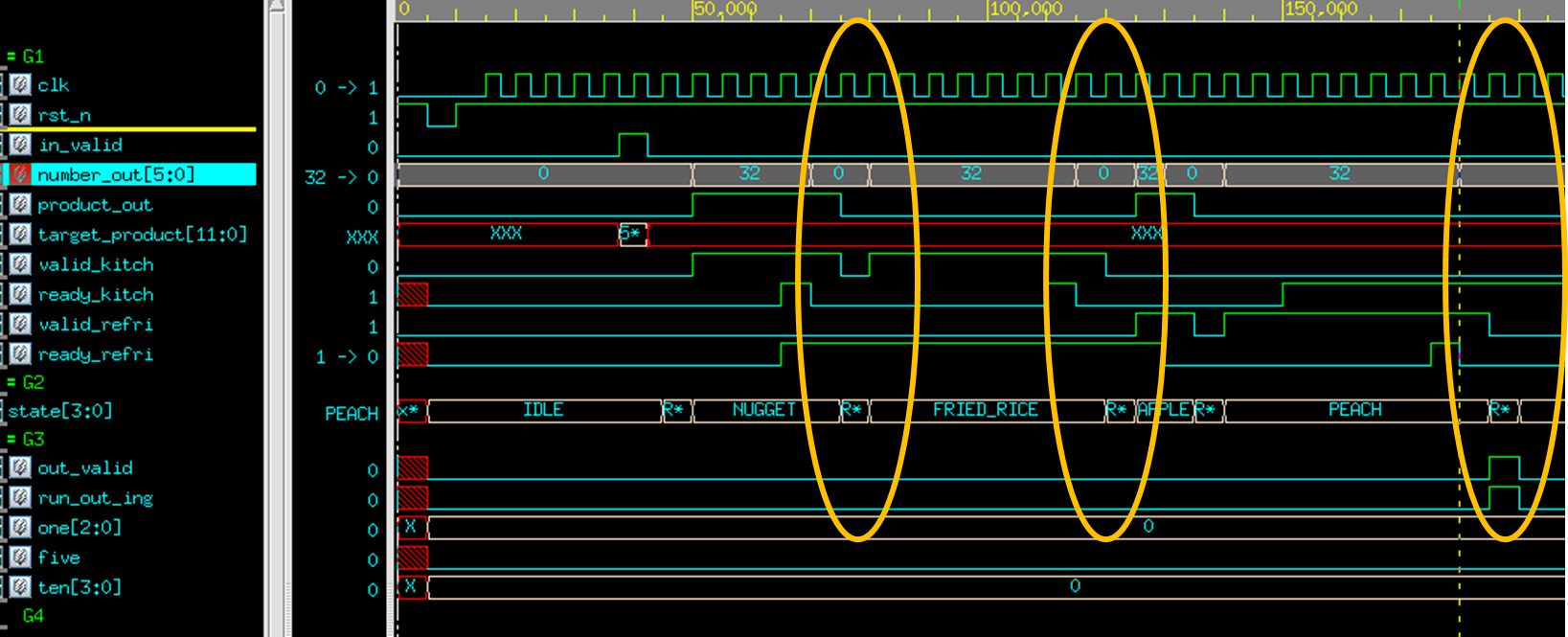


1. The “out\_valid” after we finish refilling can be brought forward by 1 clock cycle. This can be achieved by implementing a reg that determines whether this is the item currently being refilled is indeed the last item.





1. The state switch at from IDLE to REFILL can be brought forward by one clock cycle because “target\_product” is given half a cycle before the next posedge.



1. Originally, what food to refill next was decided by my REFILL state, so after a type of food is restocked, it takes one additional cycle to determine what to restock next. However, it’s more efficient to remove this state altogether and simply decide the next state after the current food finishes refilling, like so:

