2015CS10260
2015CS10667

LAB MINI PROJECT REPORT

Currency Exchange ATM

Specifications:

->Set of currencies: {Rupees,Euro, Dollar, Yen, Pound}
->Currency exchange rates are predefined constants and stored in the dispenser(as fractions).

->If input currency is one of {Euro, Dollar, Yen, Pound} then output currency will be assumed to be Rupees.
->If input currency is Rupees, then output currency can be one of {Euro, Dollar, Yen, Pound} (there will be a option).

->The number of denominations of each currency stored in the dispenser initially is decided by the programmer.

->LCD IO: We used the PmodCLS given in the kit and since the Minimum number of output characters we will require to be displayed are 48, we will use the “wait for” statement to display output for output currency in 2 screens(available estate on LCD : 32 Chars),each screen in output will stay on display for some time. Each signal for the alphanumeric characters had to be defined explicitly and this is something which couldn’t be helped. This definition had to be done in hexal numbers since the LCD was built this way

->Denominations which are stored in the dispenser:

Rupee	Yen	Dollar	Pound	Euro
1	1	1	1	1
10	10	5	5	5
50	50	10	10	10
100	100	20	20	20

500	500	50	50	50
1000	1000	100	100	100

Overall approach:

1) The set of currencies {Rupees,Euro, Dollar, Yen, Pound} is mapped to the set {1,2,3,4,5,6}.
2) The set of modes {Auto mode, Higher denomination mode} is mapped to the set {1,2}.
3) The set of denominations of each currency is mapped to the set {1,2,3,4,5,6}(higher to lower), (there are 6 denominations stored of each currency).
4) There are 5 states {S1, S2, S3, S4, S5}. Each corresponds to an output state of LCD, output of the LCD will be according to one of these states.

State S1
User will enter mode {Auto mode, Higher denomination mode}.
State S2
User will enter input currency {Euro, Dollar, Yen, Pound, Rupees}.
State S3
User will enter amount digit by digit. Amount can have any number of digits from 1 to 4.Each digit can have value 0 to 9.
State S4
User will enter output currency {Euro, Dollar, Yen, Pound, Rupees}.
Now all the input has been taken and computation can be performed.

5) In transition from state S4 to S5 computational block is invoked

State S5
The output amount is displayed along with the number of each denomination of output currency.
Error message is displayed if the given amount can not be formed by the available denominations in the currency dispenser.
6) For each state LCD computational block will decide what to display depending on current state and other parameters.

And it will send the output string to LCD output block and LCD output block will display it on the LCD.

For example if state is S2 and user has to enter inputcurrency, the output string will be
“Enter Inputcurrency: currentinputcurrency”

User can select any currency by switching between them using push buttons.

7) LCD computational block will control the LCD output block and gives it the exact string to display.
8) We store the number of denominations remaining in a 2D array for all currency.

Logics followed in

Higher Denomination:

The Machine will take the input in One currency. and one integer input amount will be obtained from the array which stores the digits of input amount. This input amount will then be multiplied with the exchange rate(stored in dispenser) and amount in output currency will be obtained.

Now an iterator will move from the highest denomination of output currency and check for maximum number of times that denomination can be taken(or the number of notes which can be fitted in the remaining ‘input amount’) , this will then be subtracted from the input amount and also from the money remaining in the dispenser, and sent again in the same loop to get the next currency denomination notes(*see example in the image*).

Auto-Mode:

First we adopted a technique to get truly random answers(using random number generator) without the use of a Higher Denominations factor built in but it had a major shortcomings as we will demonstrate: the algorithm worked as such,

Pseudo-code:

1) Generate a random number in the range of indexes supplied for the currency i.e (1 to 6 for denominations of 1,5..100/1000)
2) Check if the remaining amount of exchange*input amount is still greater than the denomination which appeared for e.g if random number was 4 and remaining amount to be converted to denominations was 200 then we add another “tally mark” or “++” to the answer array of the index 4 which is 100 rupees here.
3) If the remaining amount is less than the denomination, then change the range of the indexes allowed for e.g. in the last case if the rem. Amount was 77 and the random number was 4 (i.e 100 rupees) then we reduce the indexes allowed to 1 to 3.
4) Subtract the denomination from the amount and repeat in a loop until, the amount becomes zero or the index range becomes invalid that is when you exit the loop.

Example: input = 12600 rupees = 210 dollars. And remaining denominations are 1-0, 5-0, 10-2, 20-0, 50-3,100-2. Then random numbers generated are 6,6,3 and we are done because that would refer to 210 in the manner, 100,100,10.

FLAW: here if the random numbers were, 5,5,5 in the beginning, we would be left with 60rupees and no way to break it into the given denominations anymore, hence our machine would return an error and we won’t get any answer of the many *non-unique* answers possible..

SECOND APPROACH, a better approach:

We generate two numbers, one randomly and one that is the difference in the input exchange amount and the random numbers *(here random number can be 110 hence the other number is 100)*, now we apply the Highest denomination block for the random number and if there exist a remainder after the HD computation i.e we could not break the number in the lowest possible denomination due to unavailability of the denomination then , we move the remainder over to the difference side this number moved may from a complete 10 or 50 on the other side meaning that it may complete the prerequisite to qualify for a higher denomination and if that happens then we get a valid output, i.e. *(e.g. 220 was to be entered ,and the random number generated was 111 then other number is 109, hence we get 1 remainder after running HD computation on 111, this 1 is carried over to 109 to make it 110, so here if the 1’s note are not available we get a complete ten’s note in both*

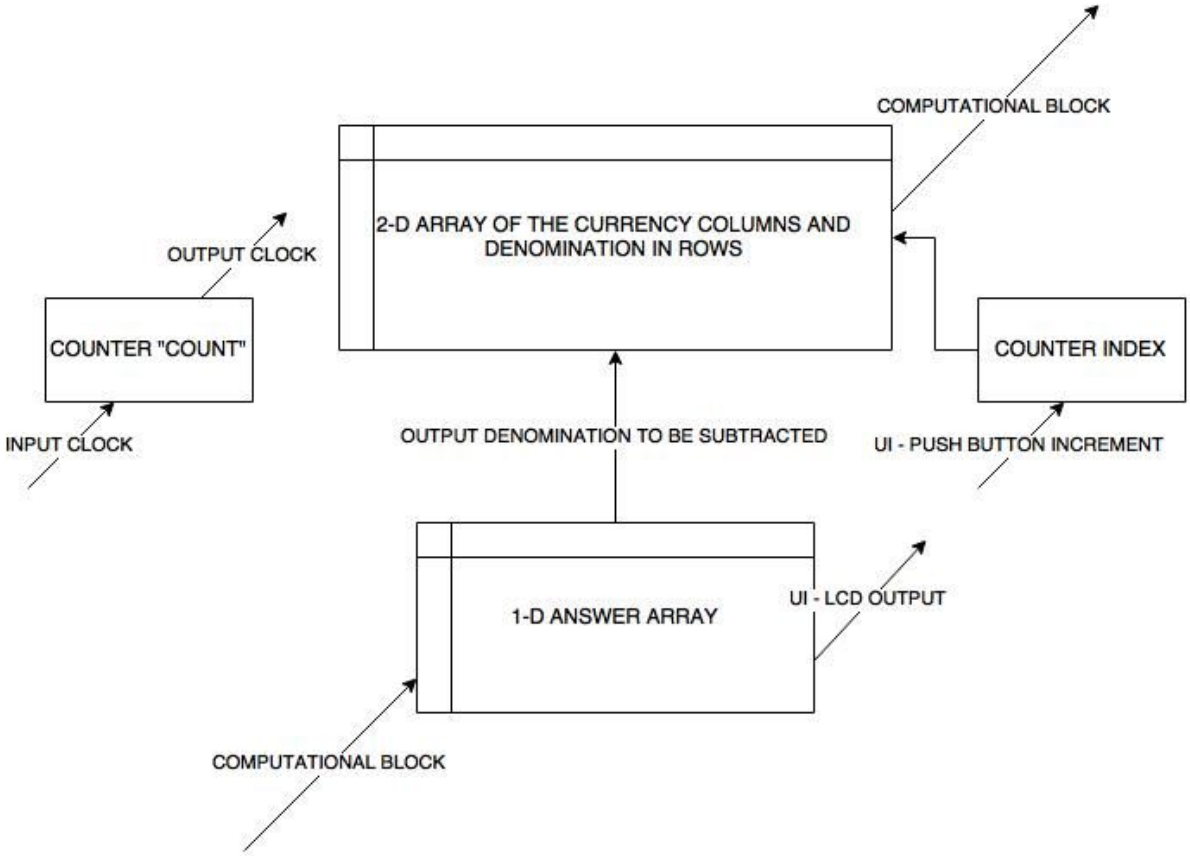
the numbers but notice that if the 10's note was not available then we won't be able to get an answer which will be the case for the number 220 directly considering no 20's note).

Block Diagrams:

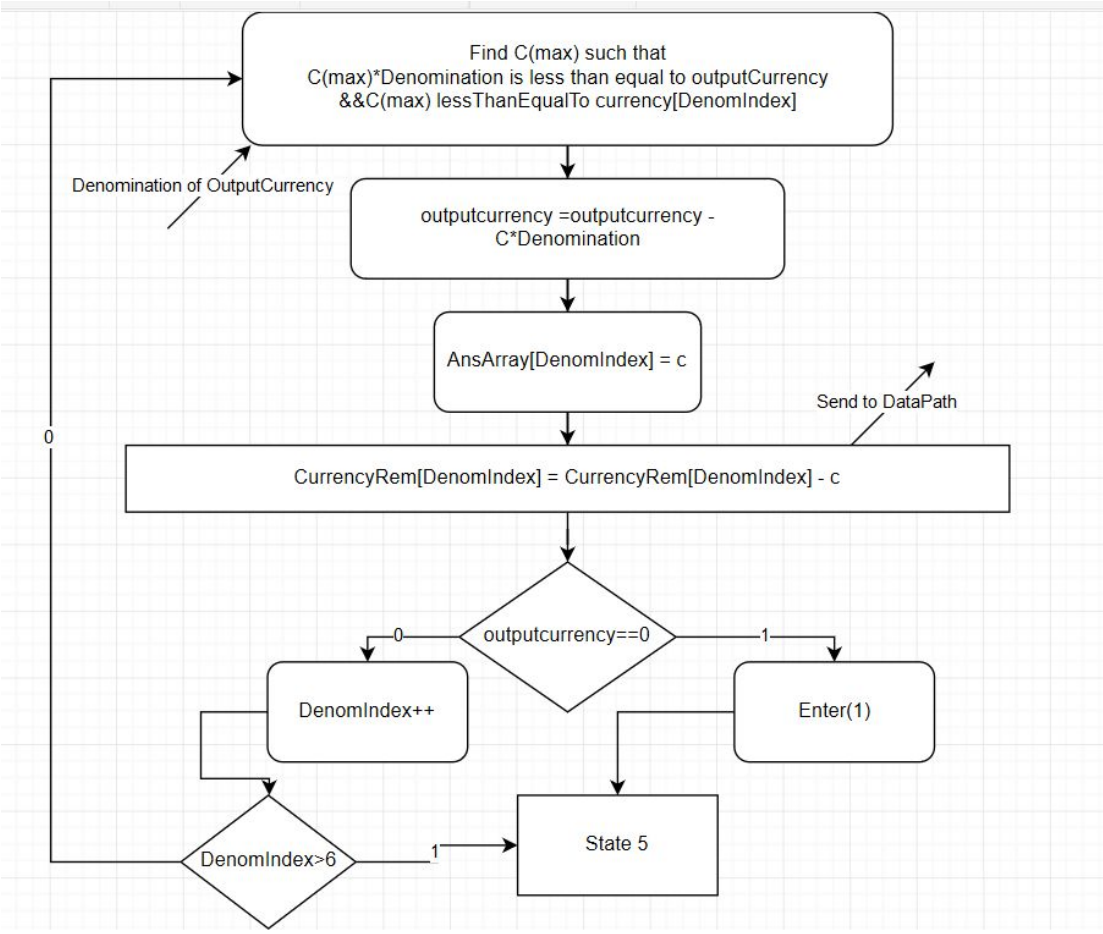
DATAPATH

The usage of arrays makes our life easier by getting easily converted to std_logic_vector with the help of bitwise changes. The computational blocks can access and alter this without worrying about next clock cycle and the need to pipeline the data.

A change in the answer array made by the controller is to be registered in the denomination arrays. And so is mentioned below



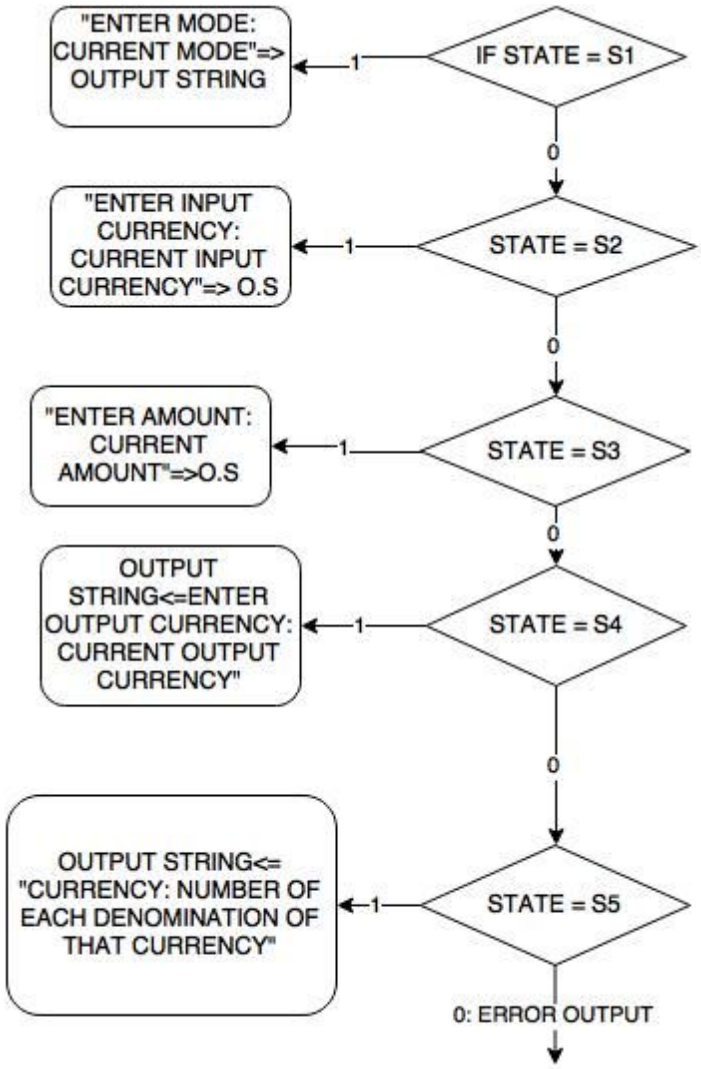
REDUCED LOGIC FOR HIGHER DENOMINATION CALCULATION



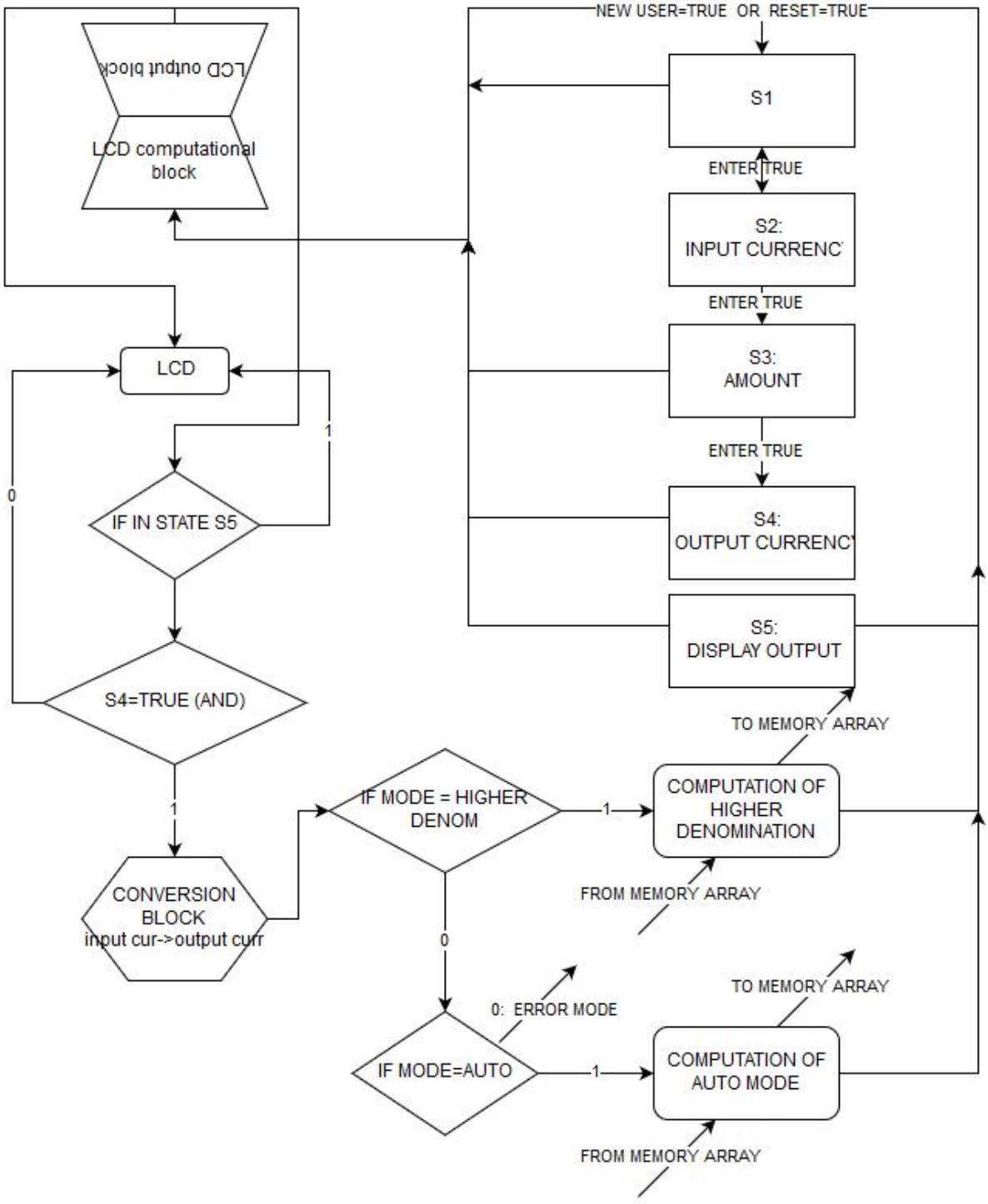
Already mentioned in the higher denomination overall approach stated above.

LCD COMPUTATIONAL BLOCK:

We have implemented a dynamic UI for the LCD in which we input characters one by one at a manual refresh frequency (that is different than the refresh frequency of the LCD screen itself which is very low <600 Hz). output for every mode is hardwired to a certain degree with the exception of counters in increment, if no state is observed display “000000000000” to show error or zero state.

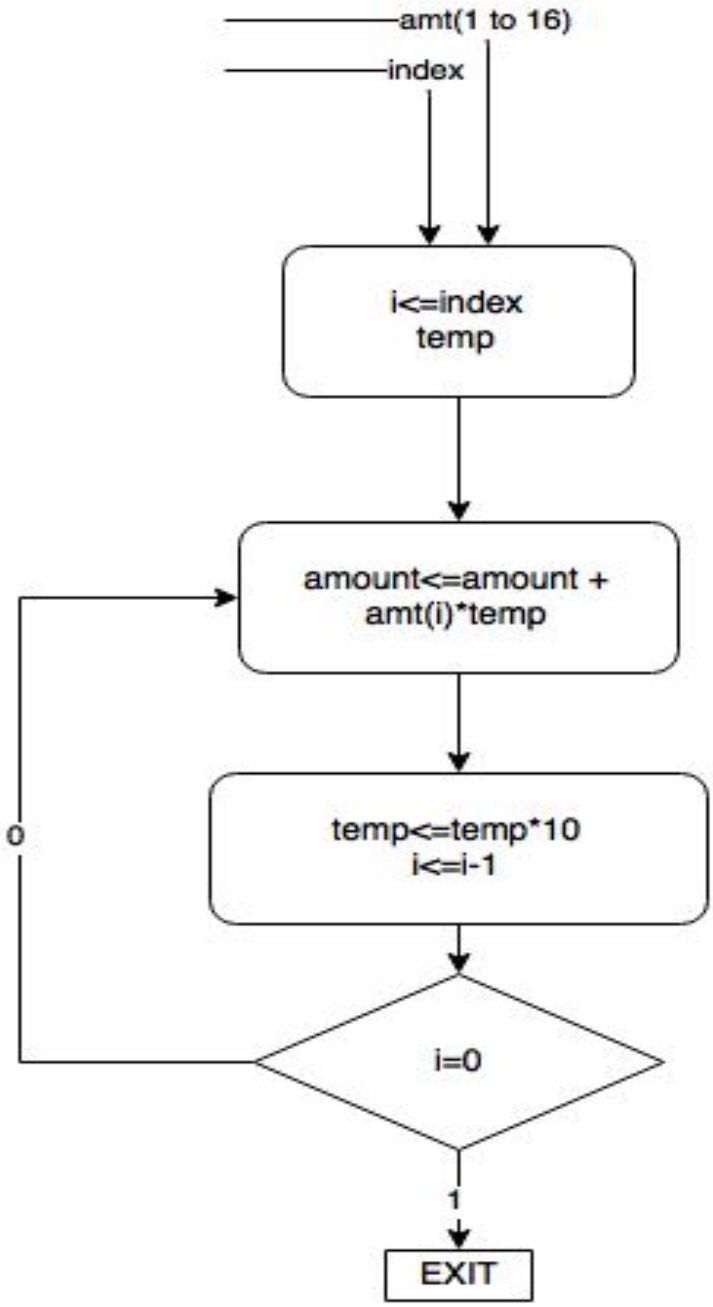


REDUCED ASM CHART WITH COMPUTATIONAL ABSTRACT BLOCKS



We also required the array"amt" to be converted to one single unsigned integer which can be used for calculations and this was done through the underlying process: i.e each bit is added together in powers of 10(i) so as to

get a decimal number. Or in another format, a decimal is basically a*10^0+b*10^1+c*10^2... and so on.



Demonstration:

As we have already abstracted piece by piece computational blocks, we run borderline cases for each block on the machine and verify that the machine functions well under the given circumstances.

The output is on LCD and is easy to understand.
Dispenser will be run on various inputs and output be verified.

Sources used:

https://reference.digilentinc.com/_media/pmod/pmod/cls/pmodcls_demo.zip
https://reference.digilentinc.com/_media/atlys/atlys/atlys_ise_gpio_uart.zip

Made By:
Prakhar Kumar 2015CS10667
Shubham Tanwar 2015CS10260