ATM

DSD First year project

05.06.2024.

Denis-Laurentiu Pescar and Dan-Adrian Muresan

PROJECT SUPERVISOR: Ing. Diana Pop

Table of Contents

1. Specifications……3
Constraints.....3
2. Design……4
2.1 Black Box……4
2.2 Control Unit and Execution Unit……..5
2.2.1 Mapping the inputs and outputs of the black box on the two components……6
2.2.2 Resources (breakdown of the Execution Unit).....6
2.2.3 State diagram of the Control Unit…… ……20
3. User manual……21
4. Technical justification for the design…….……23
5. Future developments…….……23
6.References……24

ATM

1 Specifications

The device simulates an automated teller machine on the FPGA board which has 4 functionalities:

i. Initially the device is in the idle state, after the press of the “start” switch, the device prints “CARd” on the SSD and waits for the “NEXT_OK” button.
ii. After the “NEXT_OK” button is pressed on the PMOD keypad it goes into an intermediate state, which prints “Pin” for 3 seconds and waits for the 4 digit PIN to be introduced from the PMOD.
iii. After the PIN is introduced and is validated it goes into an accepted state and then directly into the choose operation,otherwise, it goes into an error state, which prints “Err” and then goes back into the introducing pin state.
iv. When the device is in “choose operation” state the following operations can be chosen:
    a) SOLD state which prints the sum at that specific address of the PIN
    b) DEPOSIT state which waits from the user to introduce an amount to deposit, from the buttons of the FPGA, if the sum can be deposited, then it goes into an accepted state otherwise into an error state
    c) WITHDRAWAL state which waits from the user to introduce an amount to withdraw from the buttons of the PMOD keypad, if the sum can be withdrawn then it goes into an accepted state, otherwise into an error state.
    d) TRANSFER state which waits for the user to introduce another pin to transfer to. After introducing a valid pin, the user will introduce the amount to transfer. If the amount is valid, the transfer will be executed.
    e) AOP state which waits for the user to press the OK button and it will check if the AOP_ok switch is on. If yes, the user will perform another operation, if not, the card will be ejected.

Constraints

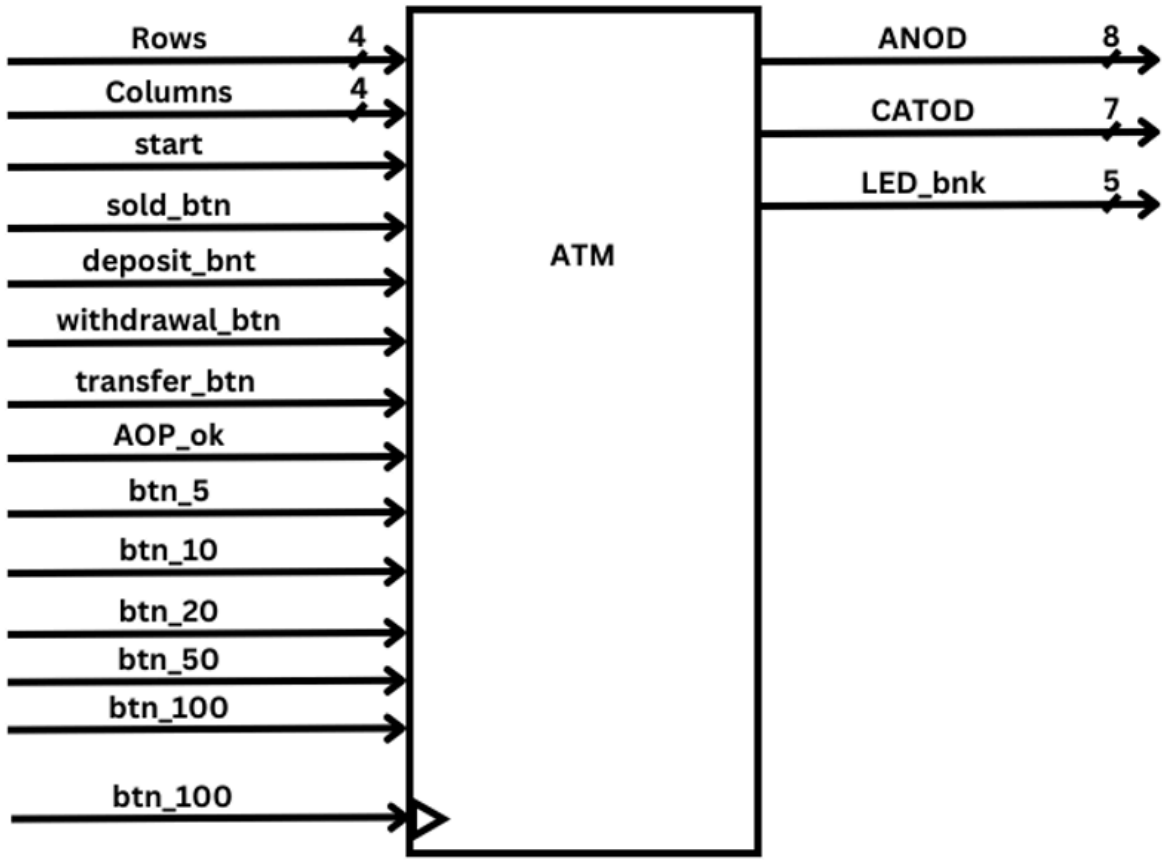
When introducing the pin, the deposit amount, or the sum to withdraw, the user must be careful of the limits of the FPGA and the PMOD keypad.

- PIN must be a 4 digit number from the PMOD keypad, the user cannot press non-digit keys(A-F) ,because nothing will happen and the device will wait for 4 digits to be finished.

- The “F” button on the PMOD keypad works as a “NEXT_OK” button which helps traversing through the states.
- The maximum deposit amount is 5000, when reaching this amount, the device will wait for the user to deposit.
- The maximum withdrawal amount is 1000. If the user tries to withdraw more than this, an error message will be displayed.

2 Design

2.1 Black Box



Inputs:

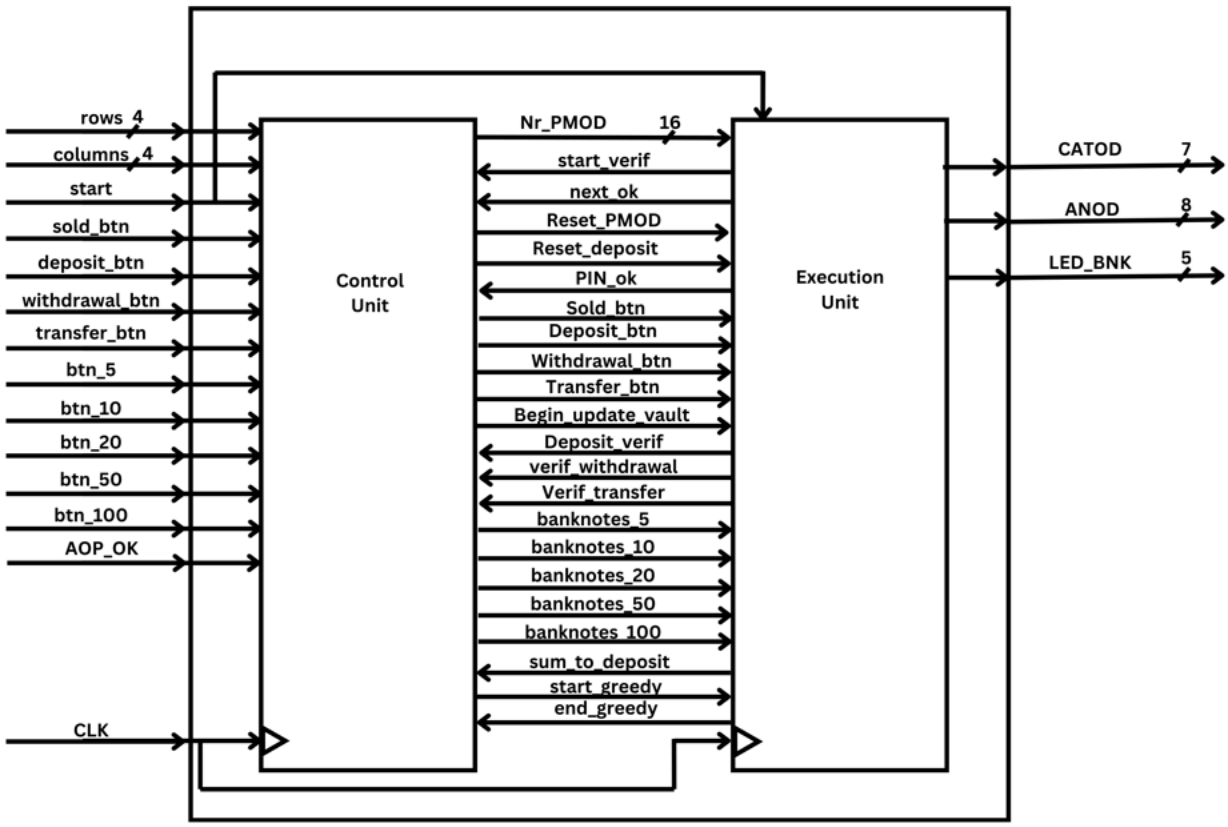
- rows and columns are 2 inputs on 4 bits, to be able to control the PMOD keypad

- start is an input which signal the start of the program
- sold_btn,deposit_btn,,withdrawal_btn,trasnfer_btn,AOP_ok are 5 inputs which signal the choose what operation to execute
- btn_5,btn_10,btn_20,btn_50,btn_100 are 5 inputs which controls the sum to be deposited

Outputs:

- CATOD is an output on 7 bits which controls what to display on each ANODE
- ANOD is an output on 8 bits which controls what anode to light
- LED_BNK is an output on 5 bits which lights depending on the extracted banknote

2.2 Control Unit and Execution Unit

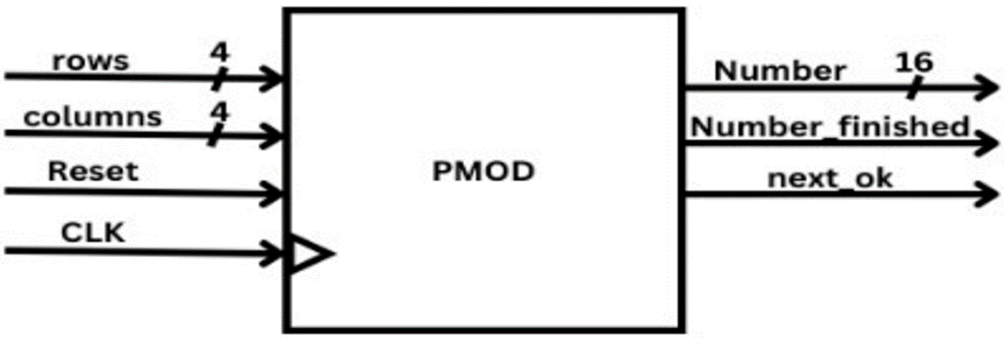


2.2.1 Mapping the inputs and outputs of the black box on the two components

Data inputs	Nr_PMOD, banknotes_5, banknotes_10, banknotes_20, banknotes_50, banknotes_100, sum_to_deposit,
Control inputs	start, sold_btn, deposit_btn, withdrawal_btn,,transfer_btn,AOP_ok, start_verif, next_ok, Reset_PMOD, Reset_deposit, PIN_ok,Deposit_verif, verif_withdrawal, verif_transfer, begin_update_vault, rows, columns, btn_5, btn_10, btn_20, btn_50, btn_100,start_greedy,end_greedy
Data outputs	CATOD, ANOD,LED_BNK

2.2.2Resources (breakdown of the Execution Unit)

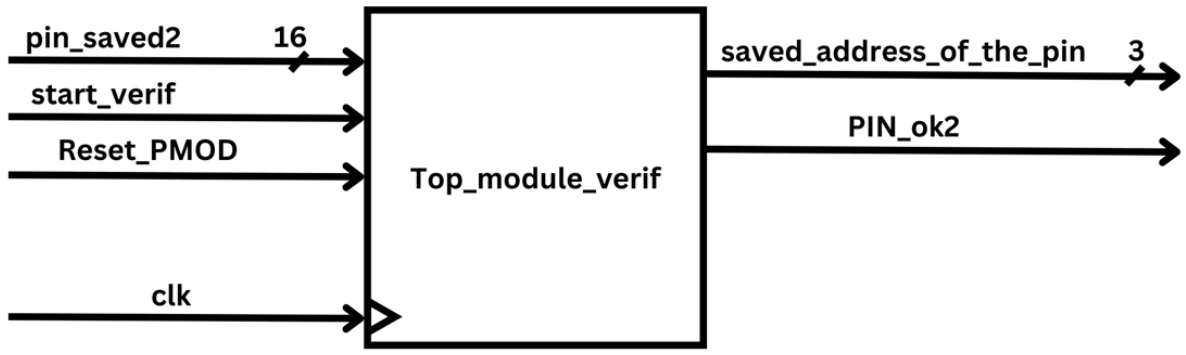
1. PMOD component



This component takes the inputs of the PMOD keypad controller and outputs the Number of 4 digits in real time. Also there are 2 more outputs, ”next_ok”, which helps to transition between states and ”number_finished” to be able to work with the number, only after 4 digits were introduced. Inside this black box, there are several other components:

- Decoder: traverses the matrix of the PMOD keypad and returns in a vector which key was pressed,also it verifies if 2 buttons were pressed in the same time
- Button debouncer: it debounces the buttons on the keypad by generating a mono impulse signal each time a button is pressed. It contains 2 D Flip Flops and a Clock Divider (100MHz -> 200 Hz).
- The same Clock Divider is also used in the main component as the clock for the Counter and the 4 Registers.
- Counter : a 2 bit counter with Carry signal active high and an Enable signal. The enable signal is active if a button corresponding to a digit was pressed on the keypad.
- 2 Demultiplexers: One has a 4 bit datapath and is used for loading the digit read into the corresponding register. The other has a 1 bit datapath and is used for enabling the load function for the corresponding register. The selections of both demultiplexers are the states of the counter. This way, each time a correct button press is detected, the counter is incremented and the digit is loaded appropriately in the register.
- 4 Storing registers on 4 bit each. These have the LD signal which enables the loading of the 4 bit number on the DIN input into the register.
- All synchronous components are working on the same clock (50 Hz) and are provided with a Reset signal.
- The values stored on the registers are then concatenated into a 16 bit number that represents the output of the main component. When the carry of the counter becomes 1 (all registers have been loaded) the signal Number_finished becomes 1.
- A special function is assigned to the „F" key of the Keypad. When an „F" is detected, the signal next_ok becomes 1 for 1 clock cycle. This signal has the role of an OK button to signal the machine to move on to processing the given input and move to the next state.

2. Top_Module_Verif component

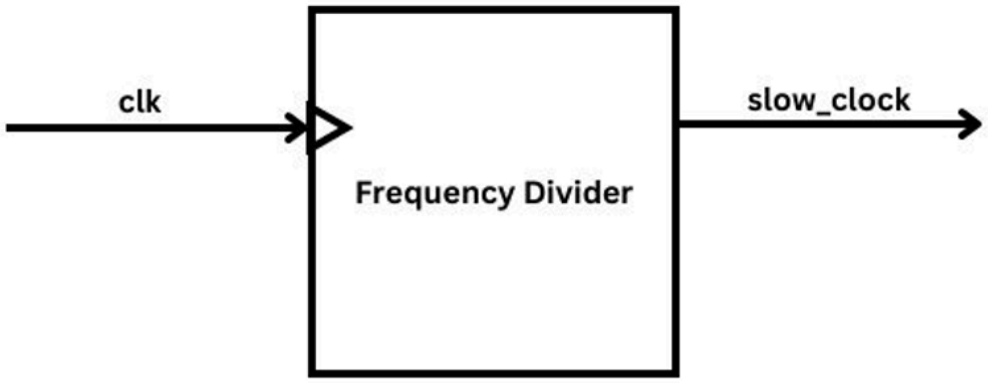


This component takes as input the saved pin introduced from the PMOD, the "Number_finished", a clock, and a Reset, the same one as in the PMOD component. The outputs of this resource are the address of the pin, and a signal that we found the introduced pin at that specific address.

We are using it two times, once for the first pin, when we start the project, and the other time for the transfer operation, when we need to introduce a new pin. Inside the black box,there are several other components:

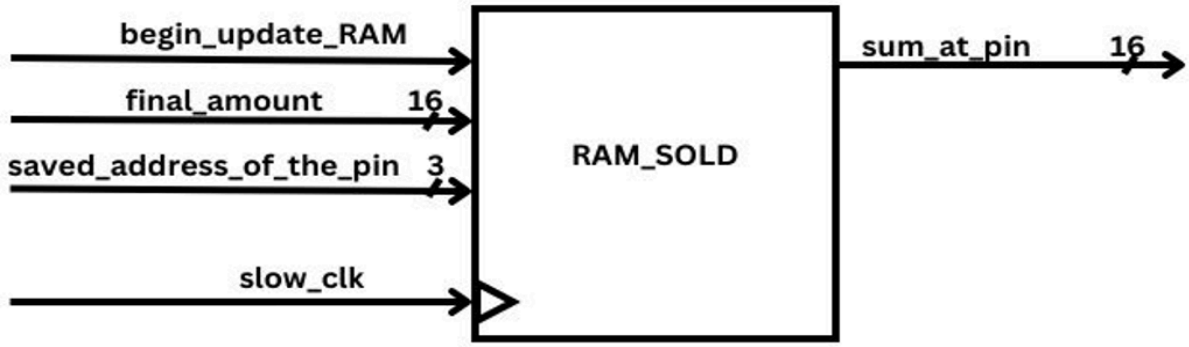
- A ROM memory 2^16 x8 , which holds the 6 pin values (the others are constant 0) on 16 bits.
- A counter, on 3 bits, that counts from 0 to 5, as there are 6 addresses in the memory, and returns the address of the current pin. This counter also has an enable, which is active high only when the 4 digits of the pin were introduced and we haven't found a valid pin yet, and the carry of the counter is 0.
- Then, we are using a comparator to return the "PIN_ok" signal, which compares the pin introduced from the PMOD with the pins in the ROM.If a valid pin is found, the PIN_ok signal becomes 1.

3.Frequency_divider



In order to synchronize all the transitions and the many operations done in the project, we are going to need a frequency divider. This divider takes the clock of the FPGA, which is 100MHZ and divides it to 200 HZ.

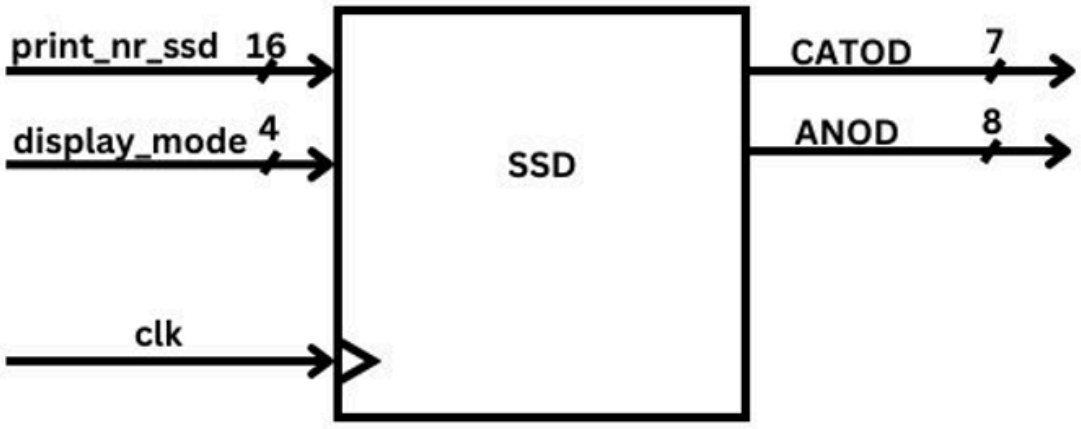
4.RAM_SOLD



This ram memory 2^16 x8, holds the values of the amount of money for each pin, saved in the rom memory. The address corresponds with each other (in the ram and rom memories) to be able to traverse through them easier.

It takes as inputs, a write_enable(begin_update_RAM), which will change during transitions, provided what we want to write/read, the divided clock, the address of the pin introduced from PMOD, and also an input to write when begin_update_vault is 1.

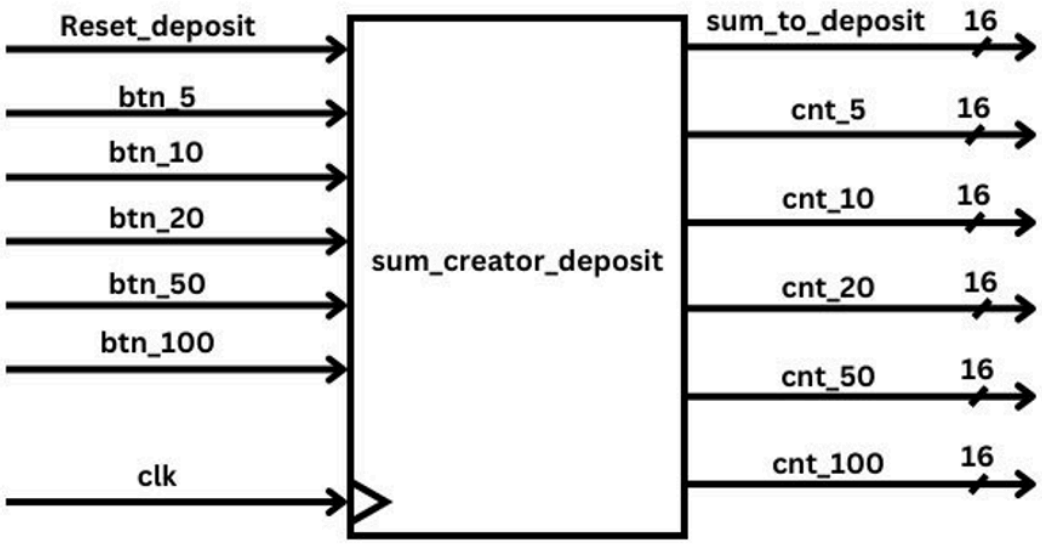
4.SSD



The SSD component prints all the messages throughout our states, and also the numbers that we are introducing. It has several components:

- A frequency divider, which divides the initial clock with 2^17, in order to give the impression that all the anodes are active, and to be able to see what it outputs on the cathodes. We are never going to use the last 4 anodes, as we do not need them, only the first 4.
- There are 2 1:8 multiplexers. One is used to output which anode we are using, and the other to model the seven segment display on each cathode.
- The component used for the cathodes also takes a display mode on 4 bits, so that we are able to control what digit/letter should be printed on the cathodes. Also there is a BCD number on 16 bits, which displays different numbers based on what state we are in.

5.Sum_creator_deposit

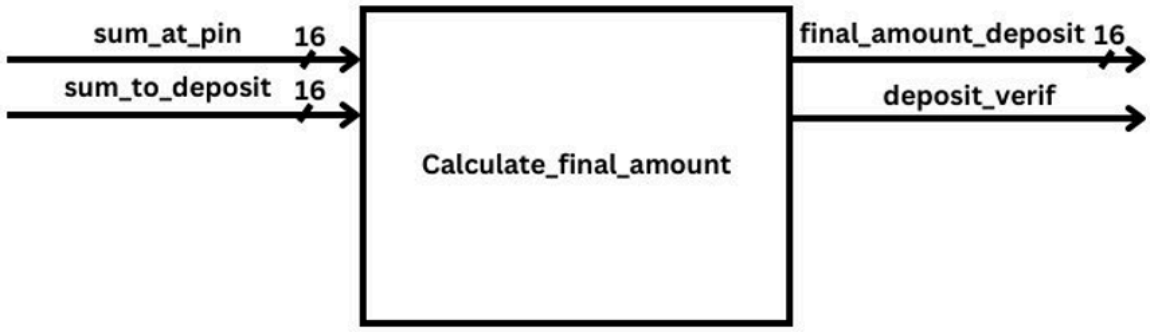


The sum_creator resource is used in the depositing state, by introducing the sum from the buttons. Each button represents a different banknote (5,10,20,50,100). It outputs the sum to be deposited on the SSD and the number of banknotes of each type. It's composed of several others elements:

- It has a different frequency divider from the other components, because we want to be able to compute the sum in real time and also see each banknote when we press the button. It divides the initial FPGA clock with 2^25.

- Each button has a debouncer, which takes another frequency and 3 D flip flops. Each clock divider for the debouncers, divides the clock with 2^23.The outputs of the 3 D flip flops are going into and and gate, so it counts until we release our finger from the button.
- Inside the main component the sum and the count of the banknotes is built by accepting a maximum amount of 5000 €.
- As the sum is constructed in decimal, we are also going to need a decimal to BCD converter, which is used in many other components. It takes each decimal digit and converts it into a 4 bit BCD digit, then divides the decimal number to 10, then takes the other digit, until the number reaches 0. After the algorithm is done, the number is on 16 bits BCD.

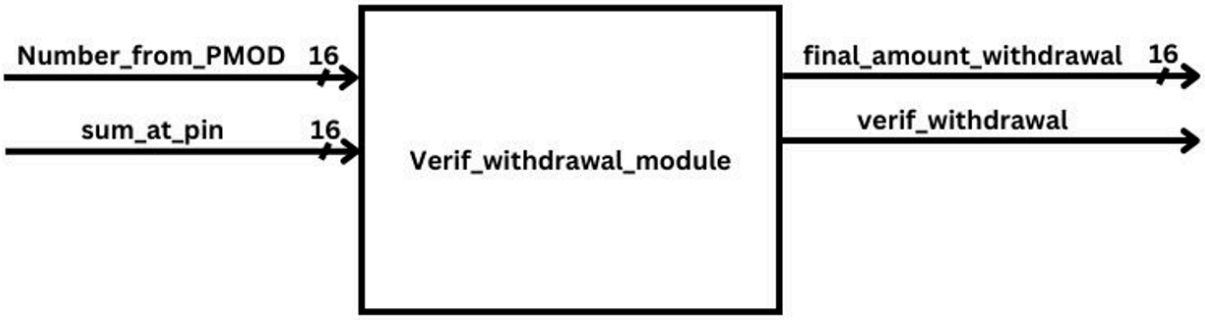
6.Calculate_final_amount



This component is an adder, which adds the sum at that specific pin in the ram memory, and the sum to be deposited. It outputs the final amount to be deposited and a deposit_verif signal which checks if the final amount is greater than 9999. As we are receiving the inputs in BCD, we are going to need some conversion components:

- A BCD to decimal converter, which takes each 4 digits of the BCD number and multiples the integer representation of them based on the decimal position(thousands,hundreds etc.)
- The same decimal to BCD converter from the sum_deposit is used here, to output the final sum in BCD.

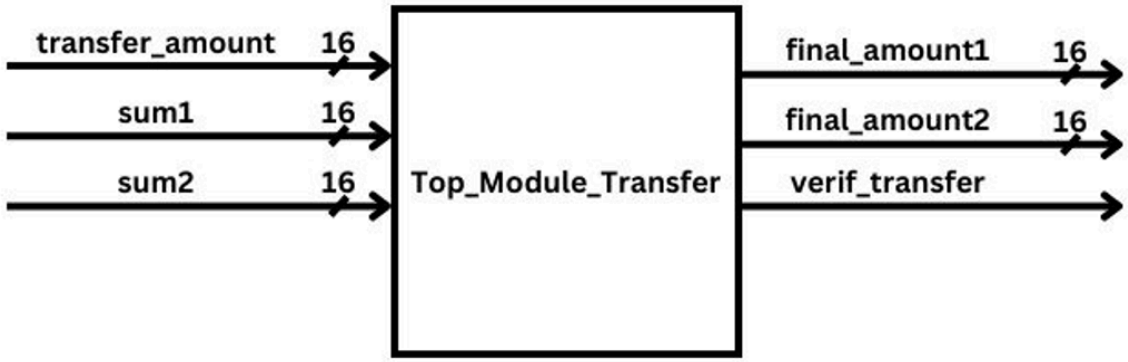
7.Verif_withdrawal_module



The verif_withdrawal module takes the sum at the current pin and the sum introduced from PMOD that we want to extract, introduced at the starting of the program, performing a subtraction between those two. It is composed of several elements:

- 2 BCD to dec converters, the same used in all the others components,converting the BCD numbers into decimal in order to perform the subtraction. It also checks that we can only withdraw a maximum amount of 1000 €, if the sum is less then the current sum of the pin and if it is divisible by 5.
- It uses a decimal to BCD converter to output the final amount to withdraw, and a signal to check if the withdrawal could be performed.

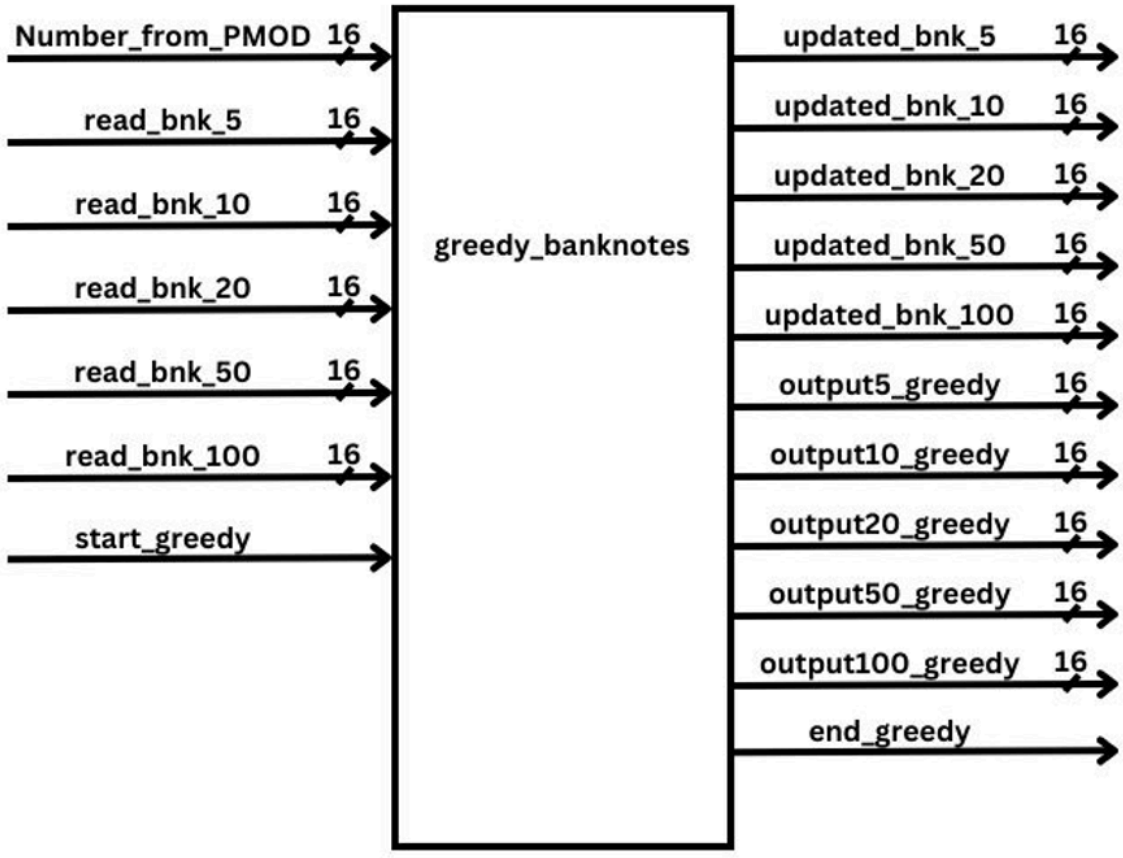
8.Top_module_Transfer



The transfer amount component is again used for performing arithmetic operations, for the transferred amount between sums at each pin. After the second pin is introduced from PMOD, the user needs to introduce the sum that wants to be transferred from PMOD.

- It has 3 BCD to dec converters, in order to make all the inputs into decimal, and then to perform the subtraction from the sum1 which is the sum of the first pin and addition to the sum2 which is the sum of the second pin.
- The updated sums will then be converted from decimal to BCD, using 2 converters. Also, the verif_transfer signal will become 1 if the transferred sum is smaller than the sum at the first pin and if the sum is less than 9999.

9.greedy_banknotes



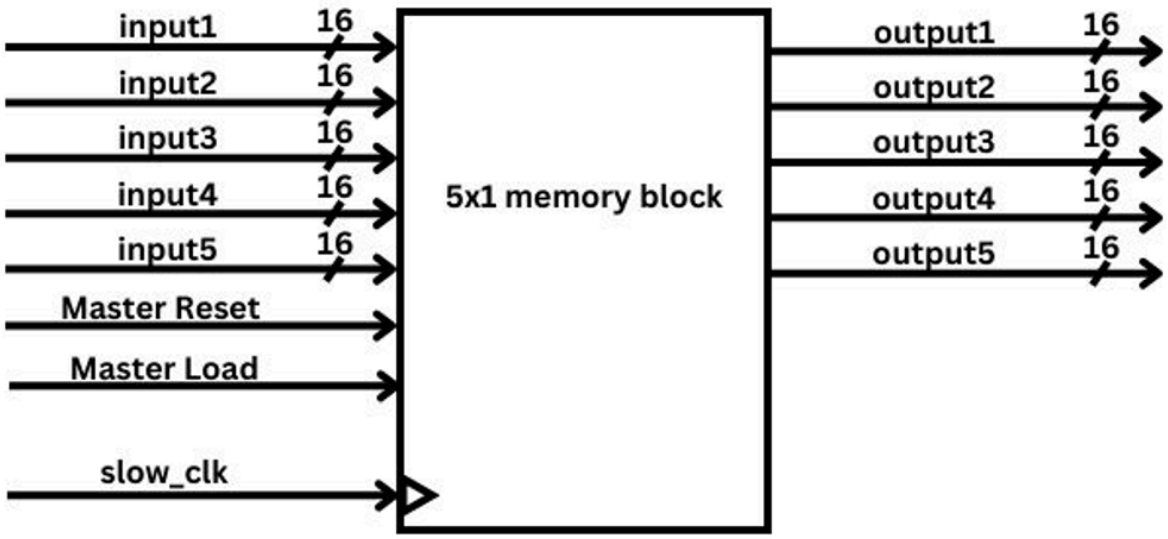
The greedy banknotes component is the actual greedy algorithm for withdrawal. When we want to withdraw, the input “start_greedy” will become active in the withdrawal state, and the number of banknotes given by this algorithm will be printed one by one on the SSD, simultaneously lighting up a led corresponding to each banknote. It takes the sum introduced for withdrawal from the PMOD and also the number of banknotes of each type stored in the RAM memory for banknotes. It outputs the number of banknotes of each type after the greedy algorithm and also the updated number of banknotes that have to be written in the memory.

- First, it uses a converter from BCD to decimal for the sum introduced from the PMOD. Then, if start_greedy is 1 it converts the number of banknotes from the RAM into integers, in order to perform arithmetic operations on them, as their value is stored in binary.
- The algorithm goes through each banknote in descending order of their value with consecutive if statements. At each one it checks for two conditions:
    - if the sum is greater than the value of the banknote
    - if there are any banknotes left of this value

If these 2 conditions are met, it divides the sum to the value of the banknote. The quotient of the division represents the theoretical number of banknotes that are used. In practice, there might not be enough left in the vault. If there are less banknotes than the theoretical value, we use all of them, perform the subtraction and update the sum. If there are at least as many banknotes in the vault as the theoretical value, the remainder of the division becomes the updated sum that the algorithm will continue with and we subtract the quotient from the current number of banknotes of that type.

- If the algorithm finished checking the sum with all of the types of banknotes, it will draw one of two conclusions:
    - if the sum is equal to 0, then the output end_greedy becomes 1, and the sum was successfully decomposed into banknotes.
    - if the sum does not equal 0, that means that the sum could not be broken down into banknotes, thus being an invalid amount.

10. 5x1 Memory block (with storing registers)



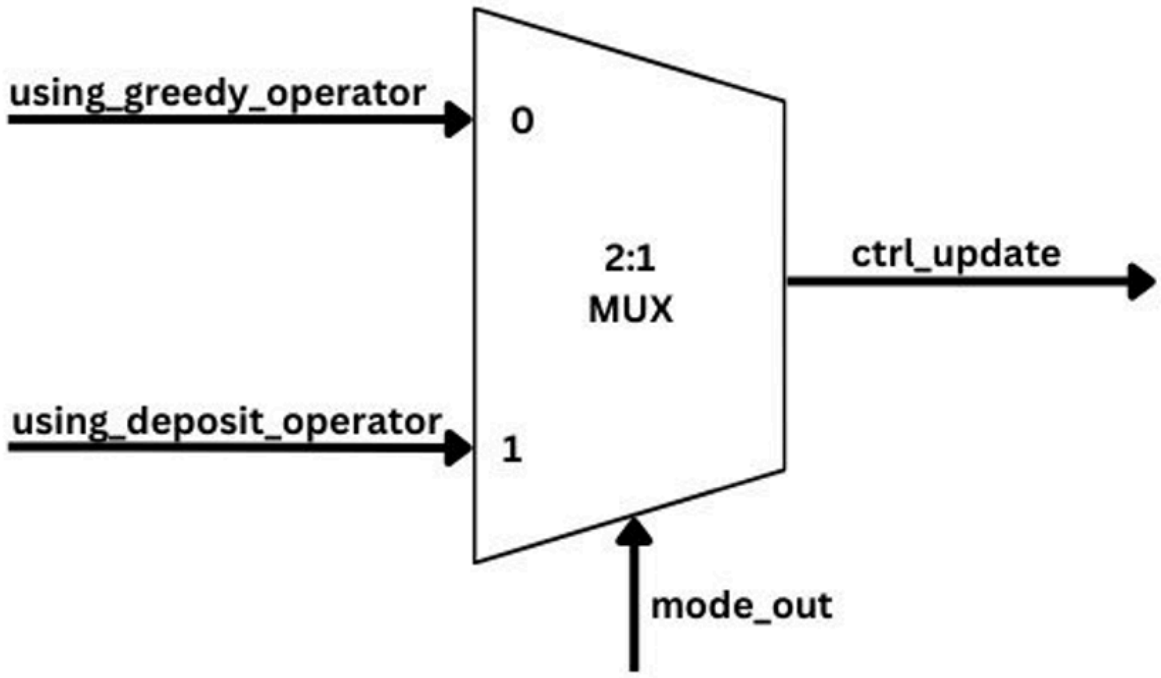
This memory block is composed of 5 storing registers. We are using them to load either the number of banknotes or the updated number banknotes after the greedy algorithm.

- The clock in this component is the divided one, described in the frequency divider resource. The loading of the registers will only happen when end_greedy is 1 and master reset is 0. The master reset input activates when the user has reached the AOP state.

11. 1 bit storing register

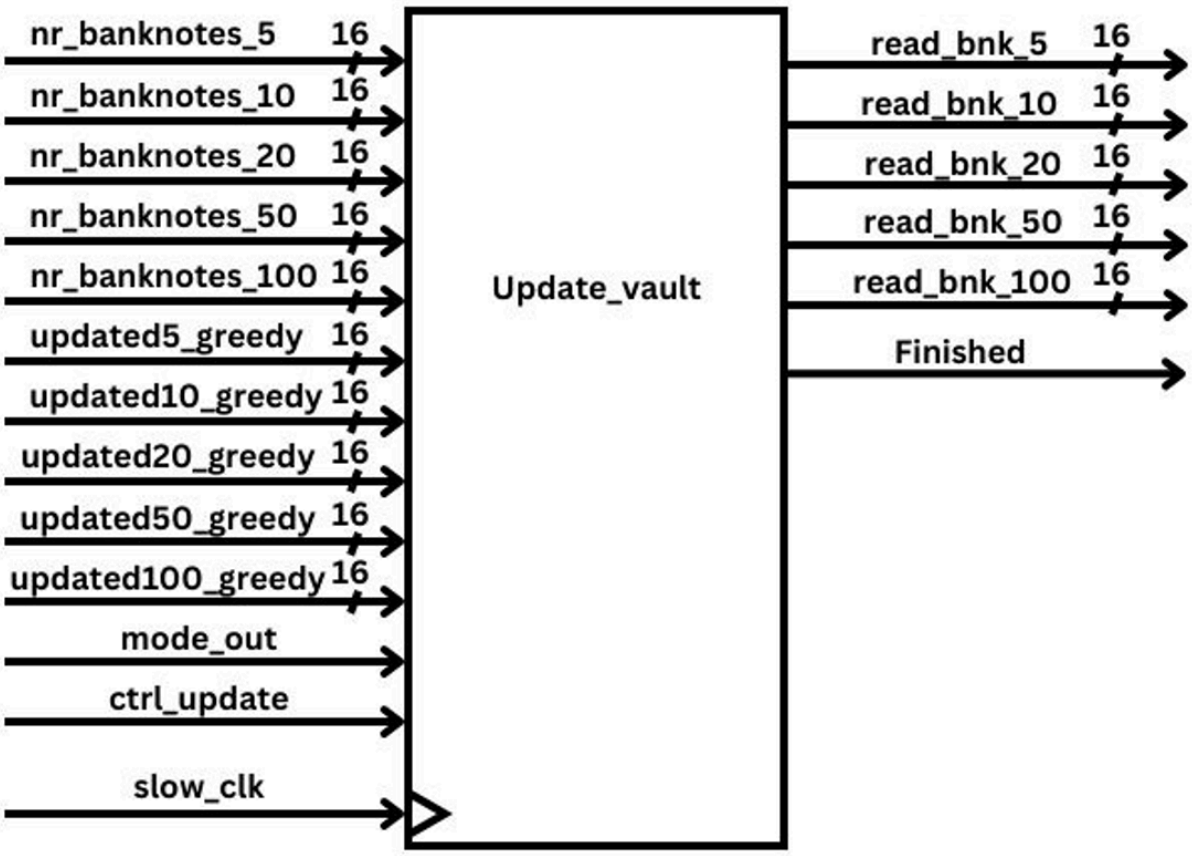
The 1 bit register components are used for storing the value of end_greedy,begin_update_vault and mode.All of them have the divided clock as input, and the same "Finished" reset, which will activate when we are done writing in the RAM vault memory block.The outputs are used to hold the value, until we are done updating the vault.

12. 2 : 1 multiplexer



This multiplexer takes the outputs of end_greedy 1 bit register and begin_update_vault 1 bit register as inputs and the output of the mode 1 bit register as the selection.The outputs of the first 2 registers are used in writing in the RAM vault memory, one is for withdrawing and the other for depositing. The output of the multiplexer is used for writing or reading in the update vault component.

13. Update_vault



This component is responsible for keeping the memory updated with the current number of banknotes after each operation that deals with them is performed (deposit or withdrawal).

Inside there are multiple circuits that work together in harmony to make sure that the memory is updated correctly. It contains the following subcomponents:

- A RAM memory with 5 addresses, just like the one we use for balance. At each address, we store in binary the number of the banknotes corresponding to that address (0 -> 5, 1 -> 10, 2-> 20, 3 -> 50, 4 -> 100). The write enable of this memory is the ctrl_update input.
- Five 16 bit registers on which we load the nr of each banknote.
- Two 1 to 8 demultiplexers, one with the datapath on 16 bits and the other with the datapath of 1 bit.

- Two 2 to 1 multiplexers, one with a 3 bit datapath, the other with a 16 bit datapath.
- Two 8 to 1 multiplexers, one with a 16 bit datapath and the other with a 1 bit datapath.
- Two 3 bit counters, both with an Enable signal and a Reset signal.

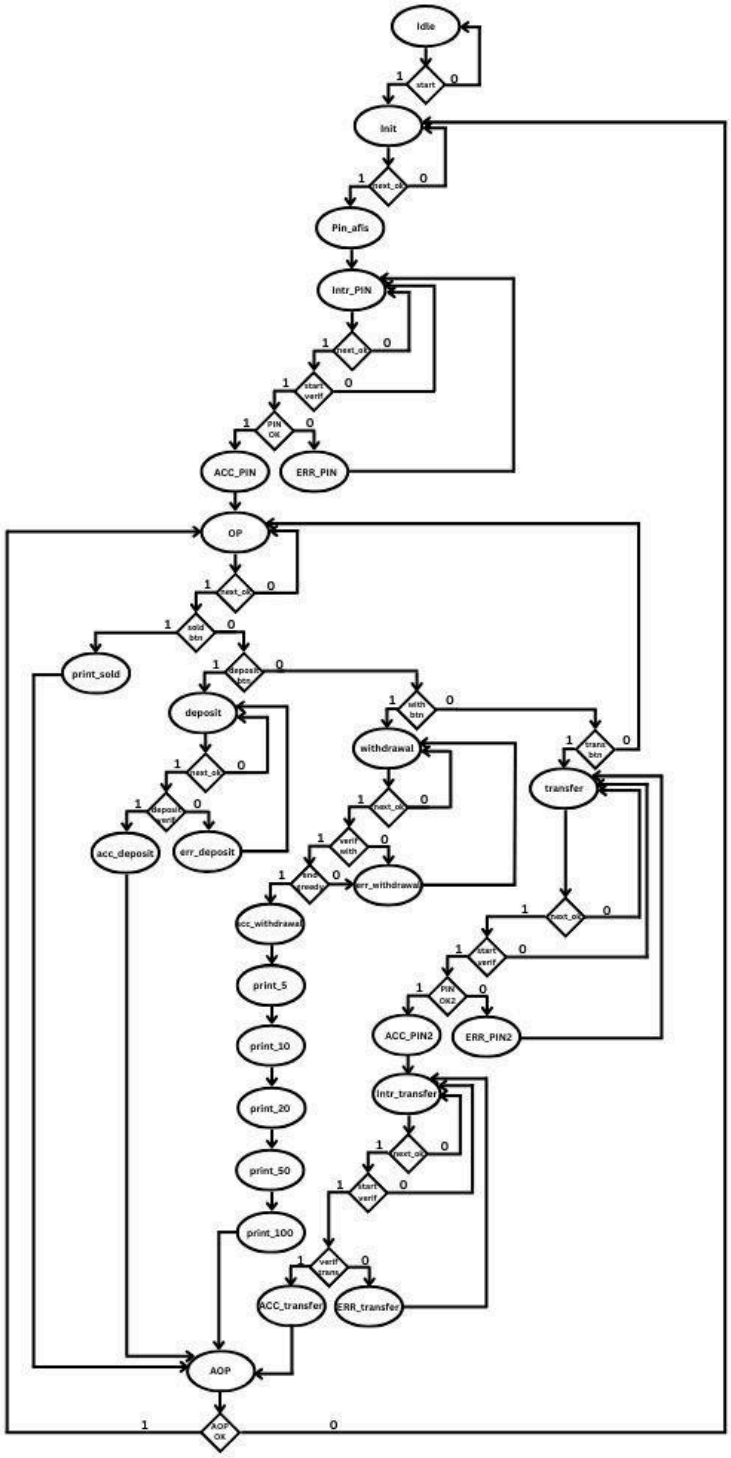
These components work in the following way:

- The component has 2 control signals, begin_update and read_from_vault. Whenever begin_update becomes 1 the other becomes 0. The signal begin_update acts as an enable for the write counter and as a reset for the read counter. Read_from_vault is the enable for the read counter and the reset for the write counter. This system ensures that we are not reading and writing at the same time.
- A 2 to 1 multiplexer decides, based on the value of the input begin_update if we are using the READ_ADDR (in case begin_update is 0) or WRITE_ADDR (in case begin_update is 1).
- The component also has a functioning mode input. This input is taken into consideration if begin_update is 1 (we need to write values in the RAM) and indicates what type of operation the system is doing with the banknotes.
    - If the mode is 1, then we load into the memory, one by one, the updated number of banknotes (the output of the greedy algorithm described previously).
    - If the mode is 0, we take the current values in the memory, store them, and increment them each time a banknote is deposited (with the use of sum_creator_deposit component)
- The output of the first counter is used as a “Read address”. We use this counter to iterate through each address of the RAM memory. We output these values (they will be the inputs to the greedy component that constantly needs the exact nr of banknotes in order to be ready to start) and store them in order to be incremented in the case of a deposit. For this we use the 5 registers mentioned before. These registers have a load enable signal and a data input. The process of storing works like this:
    - When the counter changes the output, a new value is outputted by the RAM. This value goes in the 16 bit wide 8 to 1 DMUX and the constant value ‘1’ goes simultaneously in the 1 bit 8 to 1 DMUX. The first 5 outputs (from 0 to 4) of the first DMUX go into the data input of the corresponding register and the first outputs (from 0 to 4) of the second DMUX go into the load enable of the same registers. What happens is that at each address the counter generates, we load the value into the register and then move to the next. When the counter reaches the end, a carry signal is emitted (Finished_read) to tell the machine that the operation ended.

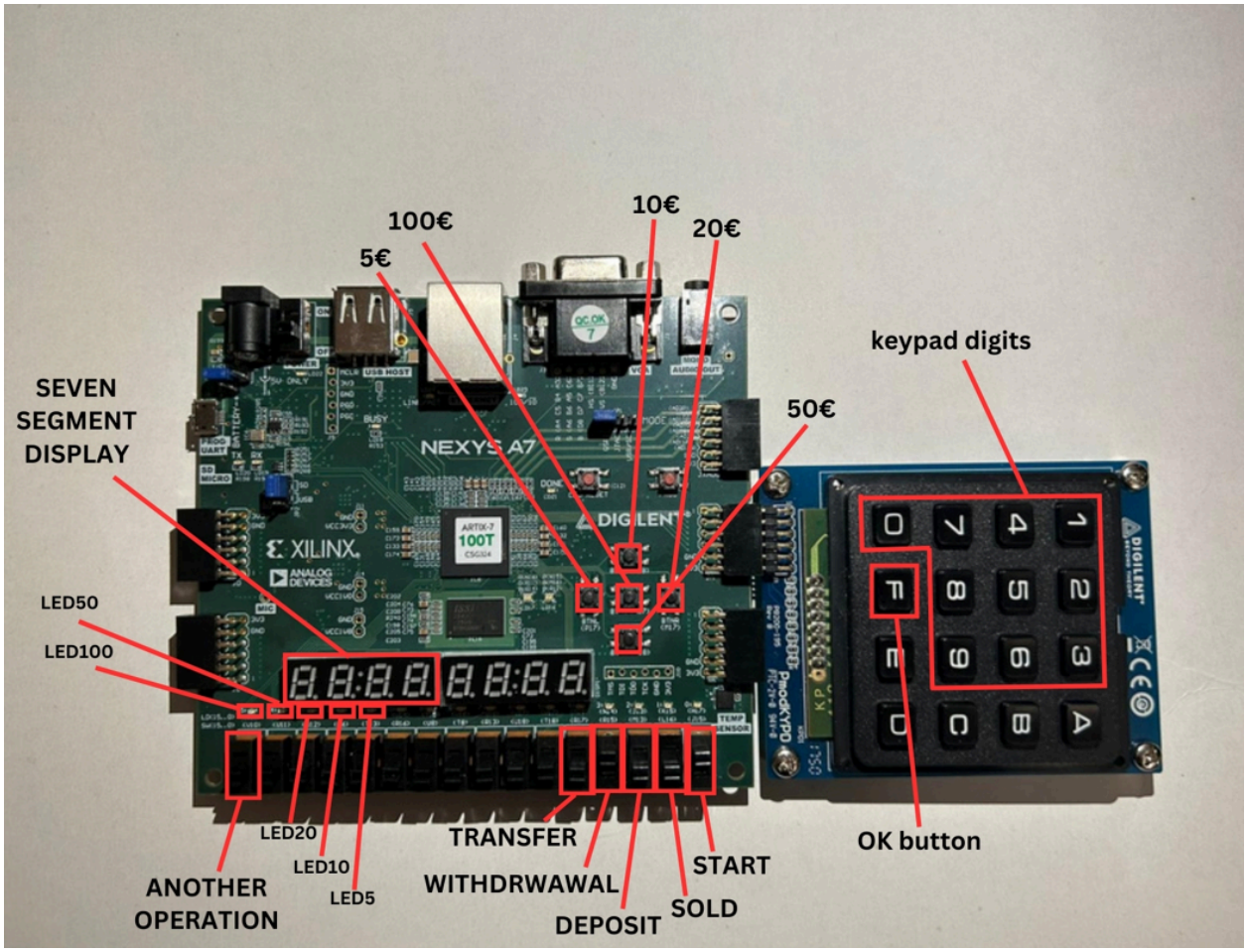
- The process of writing values back into the memory starts when begin_update becomes 1. The counter for writing starts to count, and the outputs are the selection to the 16 bit wide 8 to 1 multiplexer. The output of this multiplexer is the data to be written in the memory.

- The second 8 to 1 MUX takes as inputs the 6-10 inputs of the component(updated nr of banknotes from the greedy algorithm) and the same selections as the previous
- Now we need to decide what to load, the incremented values, or the updated nr of banknotes outputted by the greedy algorithm. This decision is made by the second 2 to 1 multiplexer that takes as selection the functioning mode and as inputs, the outputs of the 16 bit 8 to 1 multiplexers. The output of this multiplexer is the DIN input of the RAM memory.

2.2.3 State diagram of the Control Unit



3 User manual



Initially when the program starts, it outputs on the seven segment display "- - - -".Then the user, must turn on the "START" switch , and the program will output the word "CARd" and waits for the user to press the "OK" button on the PMOD keypad.Then,the user must introduce his 4 digit pin, from the keypad.The pin will be checked and either an approval message (pin was found) or an error message will be printed(pin was not found).

Then, if the pin is correct the program will print the "OP" message and will wait for the user to turn on the operation switches and press "OK". If multiple switches are on, the program will choose the one that is furthest to the right. There are 4 possibilities for the user to choose from:

- When the "SOLD" switch is turned on, on the seven segment display the balance of the user will be shown until he presses "OK".

- When the “DEPOSIT” switch is turned on, the user must enter the amount that he wants to deposit from the buttons. The amount that he wants to enter will be printed in real time on the seven segment display. The maximum amount for depositing is 5000 €. The user will hold down a button as long as he wants to add that value to the sum. When the user is satisfied with the amount to be deposited he will press the “OK” button.
- When the “WITHDRAWAL” switch is turned on, the user must enter the amount that he wants to withdraw from the keypad, the maximum amount being 1000 €.Afterwards,he will press ”OK”. If the amount can be withdrawn an approval message will be displayed, otherwise an error message will be shown. Then, the amount of banknotes of each type, given by the ATM will be displayed on the seven segment display, together with a led corresponding to each banknote.
- When the “TRANSFER” switch is turned on, the user has to enter the pin of the account where he wants the money to be transferred. After the pin is introduced from the keypad, it will be checked in the same manner as when he entered his own pin with the exception that he cannot transfer money to his own account. After the pin is checked, the user must introduce the amount that he wants to transfer. If the sum can be transferred from its own account to the other one, an approval message will be displayed, otherwise an error message will be shown.
- After each operation an “AOP” message will be printed on the seven segment display. If the “ANOTHER OPERATION” switch is turned on, then the message shown will be “OP” and the user can continue to use the ATM for further operations.Otherwise, the “CARd” message will be displayed and waits for another user(card is ejected).
- If the user wants its card to be ejected, he can turn off the start switch.

4 Technical justifications for the design

First and foremost, we chose to use the keypad for easier introduction of numerical data(Pin and various numbers in base 10).

We chose to use the same divided clock frequency for most components(excepting the ones that have their own clock divider for various reasons).

The SSD has its own clock divider, for giving the visual impression that all anodes are active at the same time.

The buttons have a different frequency for debouncing purposes.

Several registers are used along the project, as we need to keep track of the data used at different times.

There are 2 types of memories, used:

- Two RAM memories. One is used for storing and updating the number of banknotes that the ATM has in its vault and the other used for updating the balance of the users after each operation that requires it.
- A ROM memory that stores the pins of the 6 users.

5 Future developments

A few improvements can be made to the project, alongside some quality of life changes such as:

- The implementation of a “Cancel” button that allows the user to cancel an operation prior to its completion and return to the “OP” (choose an operation) state of the machine
- The display of the value of each banknote on the remaining seven segment displays instead of using LEDs.
- Optimizations for the design can be made in order to reduce redundant component usage.
- The structural implementation of certain components would improve the amount of logic that the program has to use.
- A feature to block a certain user after 3 wrong PINs were introduced (for that we would need a card number to identify each account, instead of a PIN).
- A bigger memory to hold more users along with the ability of each user to change their current pin.
- The option to receive the receipt after each use of the ATM.

6 References

https://forum.digikey.com/t/keypad-pmod-controller-vhdl/13134