A Low-Power Low-Cost Microcontroller for Security Systems

Nai Ka Chung, Tufan C. Karalar, Pak Hei M. Leung, Onur Mutlu, Cheongyuen Tsang

University of Michigan
1301 Beal Avenue
Ann Arbor, MI 48109
(734) 994-8958

427group@umich.edu

ABSTRACT

The aim of this paper is to describe the implementation of a low-power, low-cost 16-bit RISC microcontroller that will act as the core of a stand-alone security system. The controller is designed to provide moderate-high security at very low cost and to be able to operate independent of a database. It is designed using 1.5-µm SCMOS process from MOSIS and operates at a frequency of 12.5 MHz.

Keywords

Card readers, ASICs, security systems, low-power

1. INTRODUCTION

Current access control systems that authorize access to a building or a room are either relatively insecure or costly to implement and set up. Many of the security systems used to provide secure access to a site require database access to check the authentication status of the user, which requires the system to have network compatibility. This increases not only the setup cost of the security system but also the implementation cost of the microcontroller used in the security system. Besides, such systems are vulnerable to network failures and congestion, which may be quite disconcerting to the users if not critical to the security and successful operation of the system. On the other hand, many of the existing stand-alone systems fail to provide a high level of security. These systems usually only consist of a single card reader. If the card user is authorized to access the site, access will be granted without requiring any further identification. There are systems that require some kind of identification, such as fingerprint matching, but these systems are too costly to implement.

The aim of this paper is to present a microcontroller that will provide high security without the need for any database access, hence eliminating the high cost disadvantage of a network-based system but providing a moderate to high level of security depending on the data encryption algorithm used. The microcontroller is designed for consumers that need moderate to high degree of security without the need for a networked access control system. The microprocessor is also designed as a low power system using the low power static CMOS design with several chip-specific low-power components and features.

2. OPERATION

The microcontroller-based system described in this paper is to be set up at the site which needs authorization before entry. The user needs to swipe his/her card through the card reader and then enter a 4-digit PIN on the keypad in order to enter the site. The users of the site are to be given a card, which is preloaded with a 32-bit. When the card is swiped, the data on the card is read serially via an RS-232 interface into two special registers inside the microcontroller. After reading in the data, the controller waits for the user to enter the 4-digit PIN. If the PIN is not entered in a specific amount of time determined by the program running on the microcontroller, the card reader ignores the data in the card registers and waits for another swipe. If the PIN is entered in time, the microcontroller reads in the PIN via the serial RS-232 interface, converts each digit to a 4-bit number in two's complement and stores the whole PIN in a single special register. Then the controller operates on the input data from the card as well as the PIN and carry out a decryption process. If the result of the decryption matches the processed PIN entered by the user, access to the site will be granted by sending an "open" signal to the door which lasts for 10 seconds.

2.1 Data Encryption/Decryption Algorithms

It is obvious that the security of such a system highly depends on the encryption algorithm used in the process of assigning PIN's to the site users. The requirement of a PIN (password) after the user swipes the card is an important high-security feature of the system. In case the card is stolen or lost, any unauthorized user will not be able to enter the system without the acknowledgment of the password. This feature is similar to the authentication mechanisms used in the bank ATM's to provide more security than a bare card reader, but different from the ATM's in that it does not require a network for access authorization.

2.1.1 A Sample Algorithm

In our system, we employ a data encryption scheme based on the DES (Data Encryption Standard) algorithm [1]. The original DES algorithm is targeted to encode and decode a 64-bit number, encoding and decoding in 16 steps each. However, in our system we operate on 32-bit numbers since the code coming from the card is 32-bit. We limit the number of steps to 4 since this level of encryption provides us with sufficient level of security in our system (For more security, more sophisticated techniques can be used).

For our application, the 32-bit number, acquired when the user swipes his/her card, is split into 16 left-hand bits (L_1) and 16 right-hand bits (R_1). At each step, the 16 right-hand bits are assigned to be the 16 left-hand bits of the next step. The 16 right-hand bits of the next step come from the current 16 left-hand bits added to the current 16 right-hand bits processed with a key (K_n). This K_n is a function of the current step. The equations we use are as follows:

for n=2, 3,4 5.

$$L_n=R_{n-1} \qquad \text{(Eq. 1)}$$

$$R_n=L_{n-1} + (R_{n-1} \oplus K_n) \qquad \text{(Eq. 2)}$$

where $K_{2,\ 3,\ 4,\ 5}$= {0x000a, 0x00a0, 0x0a00, 0xa000}, respectively.

After completion of these four steps, we obtain a 32-bit encrypted version of the original reading. We can then obtain a 16-bit number by taking the alternating bits of this encrypted version. Logically shifting this number three bits to the right will ensure that four decimal digits are sufficient to represent the result, which is password of the card user. Finally, in order to compare the result of encryption with the keypad entry we need to convert the BCD number storing in the key-input register into a binary number.

Using this scheme, we can determine 32-bit codes and map them into 4-digit passwords. Although there is considerable redundancy in this mapping, our simulations proved that we could obtain at least 350 of these unique code-password pairs (Appendix A).

3. ARCHITECTURE

3.1 Overview

The chip proposed is built as a 16-bit load-store architecture which uses two-operand instructions. The ISA for the processor is shown in Table 1. Memory operations use register indirect addressing. PC-relative addressing is used by conditional and unconditional branches. Several instructions are to be noted in the ISA. ADDC (add with carry) and SUBC instructions are implemented in order to be able to manipulate 32-bit quantities on our 16-bit architecture. This allows the processor to decrypt the 32-bit data more easily. Arithmetic shift instructions (ASHU, ASHUI) are included in the instruction set to allow the use of more complex decryption schemes. There is also a multiply instruction which will speed up the complex decryption process.

Table 1. Instruction Set Architecture

Instruction	Type (Uses)	Comments
ADD, ADDI	Arithmetic (alu)	
ADDU,ADDUI	Arithmetic (alu)	Does not affect psr
ADDC,ADDCI	Arithmetic (alu)	Add with carry
MUL, POSMUL	Multiplier	Only register ops.
SUB, SUBI	Arithmetic (alu)	
SUBC, SUBCI	Arithmetic (alu)	Subtract w/ carry
CMP, CMPI	Arithmetic (alu)	Compare
AND, ANDI	Logic (ALU)	
OR, ORI	Logic (ALU)	
XOR, XORI	Logic (ALU)	
MOV, MOVI	Data Manip.	
LSH, LSHI	Shifter	Logical left/right
ASHU, ASHUI	Shifter	Arithmetic left/right
LUI	Data manip.	Load upper imm.
LOAD	Data manip.	Load from mem.
STOR	Data manip.	Store to mem.
Bcond	Reg+displacem.	16 conditions
Jcond	Reg. Indirect	16 conditions
JAL	Reg. Indirect	Jump and link
KRD	Special	Input ready
PEND	Special	Switch to idlemode

The processor has eight general purpose registers and three special purpose registers. Two of the special registers are used to hold the 32-bit input from the card, whereas the third one is used to hold the input from the keypad. These special purpose registers cannot be used as general purpose registers.

Our RISC processor is two-stage pipelined. In the first stage, instruction is fetched from memory and in the second stage it is decoded and executed, and register file or memory is accessed for write or read operation. For the specific modules in each stage please refer to the system-level block diagram of Figure 1.

3.2 Architectural Components

The datapath of the processor was designed using a full-custom design technique in order to achieve low power consumption, higher speed (for fast decryption), and smaller area. Included in the datapath are the program counter, instruction register, register file, ALU, and shifter (See Figure 1 and 2).

The ALU of the processor embodies a carry-lookahead adder to increase the clock frequency at which the processor can run at. Although this makes the area occupied by the ALU larger, it is necessary that we make fast the decryption process, which heavily relies on the ALU.

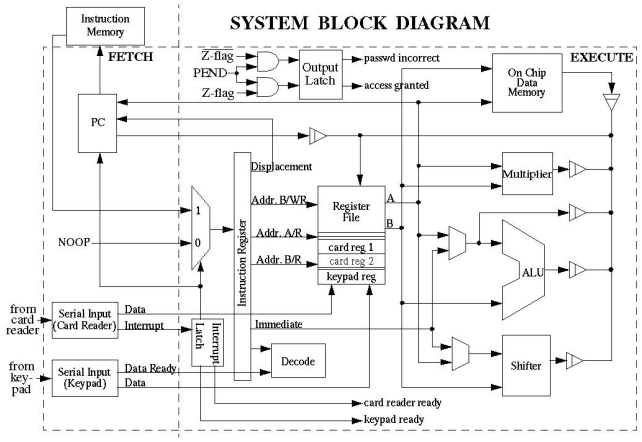


Figure 1. Block diagram of the microcontroller

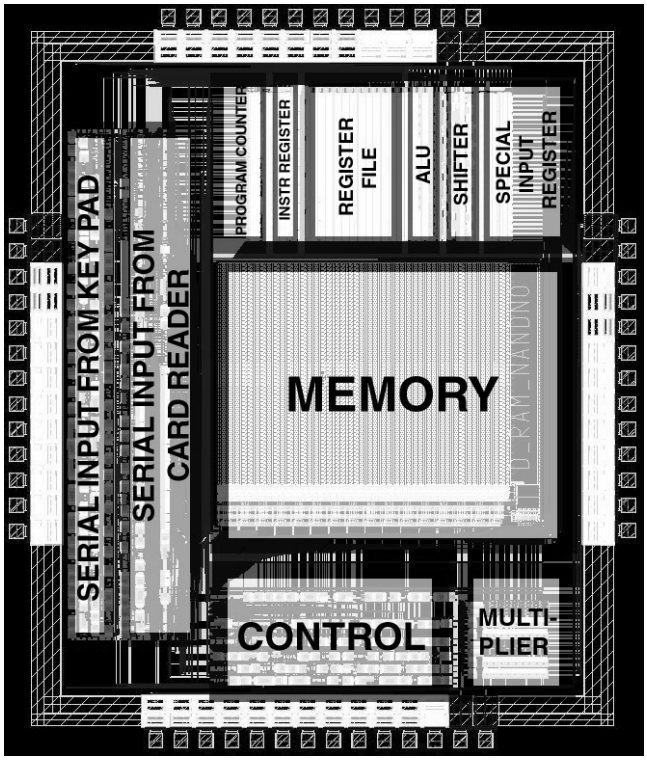


Figure 2. Layout of the card reader microcontroller

3.2.1 Low-Power Features

Aside from the low-power characteristics of the static CMOS family used for most of the parts in the microprocessor, there are several components and features that are specifically customized for low energy consumption.

The shifter used is a power-on/off logarithmic shifter. Since the shifter consumes unnecessary power when it is not being used, it is turned on only when the instruction is a shift. This reduces the overall dynamic power consumption of the chip by zeroing out the unnecessary switching activity in the shifter when no shift is being executed. The same mechanism is used to avoid unnecessary energy consumption by turning off the counters in the serial input interface when the serial input is not performed.

To further reduce the power consumption of the processor, two operation modes are introduced. In the idle mode, the processor executes NOOP's, which reduces the switching activity to a minimum according to the following average power estimation equation for CMOS and hence reduces the power consumption.

$$P_{av} = C_L V_{DD}^2 f$$,where f is operating frequency [2]

The shifter is also turned off in the idle mode. The processor starts in the idle mode and stays in that mode until the user swipes the card and enters the PIN. Once all the required data is placed in the special registers, the processor starts to execute the program. KRD instruction is used to check whether all required data is ready. When the data is ready, an interrupt signal is sent to the PC and the instruction register to start executing the decryption program. The processor returns to idle mode after the execution of the PEND (program end) instruction, which sends another interrupt signal to the instruction register.

Our processor also performs simple static branch prediction in order to reduce power dissipation. Although branch prediction is used to increase performance in deeply-pipelined high-performance general purpose microprocessors, it can also serve to reduce the power consumption in application specific processors. Branches are assumed not taken in our system. When a branch is predicted correctly, the processor will not incur the overhead of inserting a noop after the branch. Hence, the overall power consumption in the flow of the program will be reduced. The effectiveness of this approach depends on the behavior of branches in the program. If most of the branches are taken, then the usefulness of such a prediction scheme is minimal. More complicated branch prediction mechanism is not implemented because that requires considerable amount of logic, which is unfeasible for a two-stage pipelined application-specific processor.

3.2.2 Memory

The microcontroller includes an on-chip DRAM-based memory of 512 words. An on-chip memory is included to facilitate the data manipulations during the decryption process and also perhaps to store the input from the card temporarily. The proposed system does not need a large amount of storage, because it is mainly used for data manipulation. Therefore, having an off-chip data memory

and interfacing it to the chip is not cost efficient. For that reason, a small-sized on-chip memory is preferred. The memory used is a word-addressible DRAM of 1 Kbytes.

On the other hand, the instruction memory of the system is kept off the chip. The suggested instruction memory to be used by the microcontroller is an EEPROM (Electrically Erasable Programmable Read Only Memory) memory so that it can easily be re-programmed in case encryption/decryption algorithm is to be changed. As there is a possibility that these algorithms (hence the programs) need to be changed, it is more feasible to leave the instruction memory off-chip to increase the flexibility of the system. An EEPROM instruction memory meets the flexibility and low-cost goals of our design. The size of the instruction memory depends on the encryption scheme used in the program, but a memory of 8 Kbytes would suffice.

3.2.3 Serial I/O

The microcontroller receives serial input from two different devices. One is the card reader which sends the 32-bit information stored in the card. The other is the keypad, which sends the 4 characters PIN in ASCII format. It is necessary that the card reader and the keypad interfaced to the system using the RS-232 protocol.

The ASCII input sent by the keypad is converted into a binary representation of each number by truncating the higher 4 bit of the ASCII code and then storing the resulting 16 bits in a special register for further manipulation.

3.2.4 Multiplier

A two's complement 8-bit by 8-bit multiplier is included in the microcontroller in order to facilitate the password decryption process. The multiplier uses a modified version of Booth's algorithm to produce a series of partial products, which are then summed by a Wallace-tree adder circuit.

3.3 Timing Information

The proposed system has a critical path delay of 48 ns. However, in order to synchronize the system clock with the frequency of the serial input, we have decided to run the system at a slower rate which is 12.5 MHz. The timing information of the critical path is in show in Table 2 below.

Table 2. Timing information for the critical path

COMPONENT	DELAY (NS)
Instruction register	2
Control Unit	15
Register File	3
ALU	23
Writeback Tristate Buffers	5
Total delay	48

The multiplier, which has a delay of 45 ns, is not included in the critical path, because we allow the multiply instruction to take two clock cycles to complete in exchange of a faster clock speed.

3.4 Chip Specifications

The specifications of the chip are given in Table 3 below.

Table 3. Specifications for the microcontroller

Die Size	5.298 mm x 4.491 mm
Core transistor density	3441 transistors/mm^2
Datapath Size (full-custom)	1.104 mm x 1.908 mm
Datapath transistor density	6842 transistors/ mm^2
Datapath Transistor Count	14,412
Total Transistor Count	81,887
Total Number of Pins	53
Operation Voltage	0-5 V
Operation Frequency	12.5 MHz
Power Consumption	89.5 mW

3.5 Add-On Components

Besides the instruction memory there are several components that need to be interfaced to our system for proper operation. Four LED's that indicate the status of the system are useful in terms of communicating with the user who tries to enter the site. A yellow LED indicates that the system is ready. A blue LED indicates that the user has swiped the card and the system is waiting for password (PIN) input. A red LED indicates that access is denied (password incorrect) and a green one indicates that access is granted. Although these components add to the pin count of our chip, it is an informative and low-cost way to communicate with the user and is therefore necessary.

Also, as mentioned before, a keypad and a card reader which use the RS-232 serial interface need to be connected to the microcontroller.

4. DESIGN FLOW AND METHODOLOGY

Our chip was designed and fully simulated by a five-member group using the 1.5-μm AMI ABN process. The datapath of the processor was designed with a full-custom design methodology. Mentor Graphics Design Arhitect was utilized for creating top-level and transistor-level schematics. ICStation was used for the layout construction, Accusim was used for analog simulation and Quicksim was used for the digital simulation and functional verification of the design. Delays of the datapath components were generated at each stage using the capacitance extraction tools of Mentor Graphics and added to all critical points. Each component of the datapath was then simulated and functionally verified using Accusim and Quicksim. The whole datapath was verified using the same tools.

The layout of the control unit, on-chip data memory, multiplier, and the serial interface were created using the Verilog Hardware Description Language and Epoch design compiler from Cascade Design Automation. The Verilog modules for the components were functionally verified using Signalscan. After the generation of the layouts of all components, the controller, datapath, and memory were automatically routed and floor-planned using Epoch design compiler. All parts created by the Epoch design compiler were functionally verified using QuicksimPro.

The whole chip was extensively simulated and functionally verified using QuicksimPro. Input and output pads were added from Epoch's cell library and simulations were done on the final schematic with the input and output pads. The simulations were done for each instruction as well as for several complicated test programs. From the simulations, the optimal operable frequency of the chip was determined to be 12.5 MHz.

5. TESTING

5.1 Verification

As mentioned in the previous section, the microcontroller was extensively simulated at each stage, component by component and as a whole using both analog and digital simulation tools. Furthermore, specific choices were made during the layout process to enhance testability before and after fabrication. These choices are described in the next subsection.

5.2 Design for Testability

Design for testability was an important concern in the design of our microcontroller. We have adopted a systematic approach to ease the testing of our chip. Scan design method was used to separate the memory modules from combinational modules during testing. Each register in the microcontroller was laid out as a scannable register using multiplexers to select between the scan chain input and normal input. General purpose registers and special purpose registers are exceptions to this, because their outputs can easily be determined by reading from them. Program counter, instruction register (which acts as the single pipeline latch) and program status register (which stores the state of the machine) were all included in the single scan chain. The inclusion of the scan chain inside the chip increases the number of pins by three and increases the critical path delay of the circuit, but considering the importance of testing of microprocessors, it is a crucial tradeoff that is necessary to make. An important advantage of having a scan chain on the microcontroller is the fact that it makes the test pattern generation relatively easy using automatic test pattern generation.

5.3 Final Testing

As the proposed microcontroller is yet to be fabricated, the final testing has not been done on the chip. However, due to the design for testability and small number of pins, we expect the final functional verification of the chip to be fairly straightforward after fabrication.

In the test mode, specific bit patterns (test vectors) will be fed to the scan chain input of the chip. Results coming out of the scan chain output will be monitored for correctness and analysis. Automatic test pattern generation equipment is planned to be used for test vector generation for the testing of the combinational blocks of the microcontroller.

Upon the fabrication of the microcontroller, we plan to test it using an HP82000 IC Development System. For the generation of the test vectors to be used by the HP82000, we plan to use the TDS Software System by Fluence Technology.

Eventually, after the complete functional verification following the fabrication, we plan to connect the microcontroller to a compatible card-reader, keypad and LED's. The real testing will be completed when it is verified with all the input output interfaces.

6. CONCLUSION

In this paper, a low-cost low-power microcontroller for a stand-alone security access control system is presented. This system is proposed to be a convenient alternative to other systems for consumers that need moderate to high level security without any networking considerations and at a low cost. It is crucial to reiterate that the level of security provided by such a system heavily depends on the security of the encryption algorithm used in the encryption/decryption process. Therefore, a good selection of the encryption algorithm, similar to the one proposed in this paper is essential for the reliability of the security system.

7. REFERENCES

[1] Van Der Lubbe, Jan C. A., Basic Methods of Cryptography, Trans: Steve Gee, Cambridge University Press,p.62, (1998)

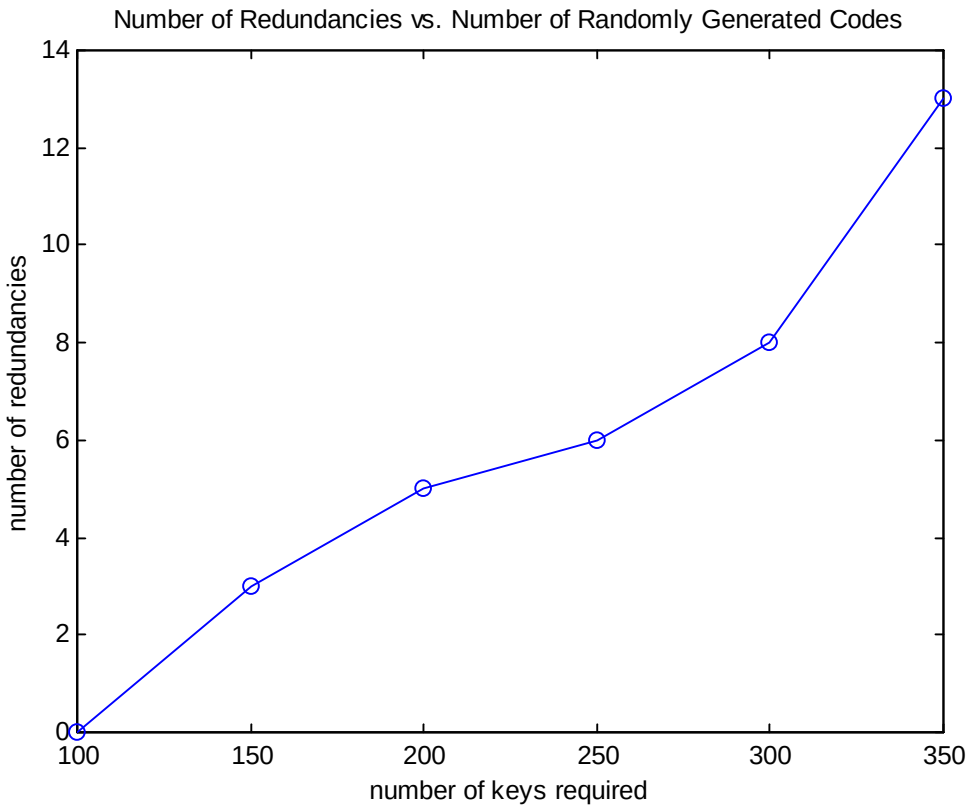
[2] Rabaey, J. M., *Digital Integrated Circuits: A Design Perspective*, Prentice Hall, Upper saddle River, NJ, p.235 (1996).

Appendix A.

We implemented a MATLAB program that can be used to determine the codes and assign the passwords for a given number of users. Codes are randomly generated until the required number of codes, each of which map to a unique password key, are obtained.

The program also plots the relationship between number of redundancies and requested number of codes. Each code corresponds to a user

Figure 1.



SAMPLE MATLAB PROGRAM

```
clear
factors=fliplr(pow2([0:15]));
count=1;
track=1;
passwords =[];
limit(1)=100;
iteration=1;
as(1,:)=[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1];%0x000a
as(2,:)=[0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0];%0x00a0
as(3,:)=[0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0];%0x0a00
as(4,:)=[1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0];%0xa000

while(track-count<=10)

   while (count <=limit(iteration))

       initial_left=(normrnd(0,1,1,16)>=0);
       initial_right=(normrnd(0,1,1,16)>=0);

       left(1,:)=initial_left;
       right(1,:)=initial_right;
       num(1,:)=[left(1,:) , right(1,:)];

       for i=2:5
          left(i,:)=right(i-1,:);
          xor_res = bitxor(right(i-1,:),as(i-1,:));

          xor_result_dec=sum( factors.* xor_res);
          left_dec=sum(factors .* left(i-1,:));
          i;
          dec2bin(left_dec) - num2str(0);

         result_dec=xor_result_dec+left_dec;
         if(result_dec <= pow2(16))
       right(i,:)= [zeros(1,16-ceil(log2(result_dec))) (dec2bin(result_dec) - num2str(0))];

         else

             dec2bin(result_dec ) - num2str(0);
          while(result_dec>pow2(16))
           result_dec=result_dec-pow2(floor(log2(result_dec)));%this effectively takes just 16
bits
             % which is effectively the case in our adder;
           end

           dec2bin(result_dec) - num2str(0);
```

```
            right(i,:)=[zeros(1,16-ceil(log2(result_dec))) (dec2bin(result_dec) - num2str(0))];
        end

        end
        final = [left(5,1:2:16) right(5,1:2:16)];
        final_dec=sum(factors .* [zeros(1,2) final(1:14)]);

        while (final_dec>pow2(13))
        final_dec=final_dec-pow2(floor(log2(final_dec)));
        end

        final_dec;


    if(~isempty(passwords))
        present=0;
        for q=1:length(passwords)
            if(final_dec==passwords(q))
                present=1;
            end
        end

        if(present==0)
            passwords(count,:) = final_dec;
            codes(count,:)=[initial_left initial_right];
            count=count+1;
        end

    else
    passwords(count,:)= final_dec;
       codes(count,:)=[initial_left initial_right];
       count=count+1;
    end

    [left(5,:) right(5,:)];
    track=track+1;
end
tracks(iteration)=track;
counts(iteration)=count;

limit(iteration+1)=limit(iteration)+50;
iteration=iteration+1;
end
%display the codes
passwords
codes;
```