REFERENCES

1. Anderson,R. (2001). Security Engineering, Wiley.

2. Stamp, M. (2006). Information Security, Wiley Inter Science.

3. Mee, F.S ,Tong N.Y, and Hiah, S.P, (2004). Systems Security, Pearson Prentice Hall.

4. Blake, I, Seroussi, G, and Smart, N, (2000). Elliptic Curves in Cryptography, Cambridge University Press.

5. Cerven, P, (2003). Crackproof Your Software: Protect Your Software Against Crackers, No Starch Press.

6. Burton, D.M, (1998). Elementary Number Theory, fourth edition, Wm. C. Brown.

7. Main, A, (2007). Application Security: building in security during the development stage, at http://www.cloakware.com/downloads/news/Application_Security_Building_in_Secirity_During_the_Development_stage.pdf, 24 February 2007.

8. Fluhrer, S, Mantin, I, and Shamir, A, (2007). Weakness in the key scheduling algorithm of RC4, at http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf, 24 Feb 2007.

9. Feller, W, (1968). An Introduction to Probability Theory and Its Application, Third Edition, Wiley.

10. Glover, D.B, (1991). Secret Ciphers of the 1876 Presidential Election, Aegian Park Press.

11. Golomb, S.W, (1999). Shift Register Sequences, Aegian Park Press.

12. Manner, J, (2003). Hand Based Biometrics, Biometric Technology Today, pp. 9-11.

13. John, (2003). John the Ripper password cracker, at http://www.openwall.com/john, 25 Feb 2007.

14. Kahn, D, (1996). The Codebreakers: The Story of Secret Wrinting, revised edition, Scribner.

15. Danmarco, K, (2006). Keyless Entry System, at http://www.danmarco.com/home.htm, 25 Feb 2007.

16. Fish, S.D, (2001). Remote Keyless Entry System Overview, at http://www.maxim-ic.com/appnotes.cfm/appnote_number/1774, 26 Feb2007.

Appendix A

Source Code

```
/*Design by Zaini bin Sulaiman
  Final Year Project 2007
  Emergency Dorr Car Entry System
      -to overcome prob where user left key inside the car and grand
theft auto
  Supervisor: Pn. Norhawati
*/
/*Assumptions:
      -when opened by switch, there is no alarm(coz if open car wif no
key,
      there'll b alarm)
      -switch can only open


*/
/*Initial condition of the system
      -there is no key(key=1)
      -manual lock is locked (man_lock=0)
      -system is locked
*/

////////////////////////////////top level design//////////////////////
module version_11
(man_lock,sw,clk,display1,display2,lock,key,disarm_alarm,light);
////////////////////////////////ports declarations//////////////////////

input clk,key,man_lock;
input [3:0]sw;


output [7:0]display1,display2;
output lock,disarm_alarm,light;


reg lock,disarm_alarm,light;
reg [3:0]state;
reg [2:0]swcount;
reg [7:0]display1,display2;


parameter zero=8'b00000011,
              one=8'b10011111,
              two=8'b00100101,
              three=8'b00001101,
              four=8'b10011001,
              five=8'b01001001,
              six=8'b01000001,
              seven=8'b00011111,
              eight=8'b00000001,
```

```
              nine=8'b00001001;
parameter A=4'b0111,
              B=4'b1011,
              C=4'b1101,
              cancel=4'b1110,
              idle=4'b1111;
parameter check_key=1,
              check_init=2,
              next_sw=3,
              next_sw1=4,
              next_sw2=5,
              swcountcheck=6,
              waitlock=7,
              waitmanlock=8;


////////////////////////////////////body/////////////////////////////
always @ (posedge clk)//trigger states at positive edge clock
begin

//////////////////////////external input check routine////////////////
if (!key)
      state<=check_key;
else begin
case (state)
      check_key: begin
                        if (!key)
                              begin
                                    swcount<=3'b0;
                                    display1<=six;
                                    display2<=six;
                              end
                        else
                        begin
                              swcount<=3'b0;
                              state<=check_init;
                        end
                        end

      check_init: begin
                              display1<=one;
                              display2<=one;
                              case (sw)
                                    idle  : begin
                                                      state<=check_key;
                                                end
                                    cancel: begin
                                    //to cancel input n off alarm
                                                      swcount<=3'b000;
                                                      display1<=six;
                                                      display2<=three;
                                                      light<=1'b1;
                                                   disarm_alarm<=1'b1;
                                                      state<=check_key;
                                                end
                                    A           : begin
                                                      swcount[0]<=1'b1;
                                                      swcount[1]<=1'b0;
                                                      swcount[2]<=1'b0;
                                                      state<=next_sw;
                                                end
```

```
                        default: state<=check_key;
                  endcase
            end
next_sw: begin
            display1<=two;
            display2<=one;
            case (sw)
                  idle: state<=next_sw;
                  A     : begin
                              state<=next_sw;
                        end
                  B     : begin
                              swcount[1]<=1'b1;
                              state<=next_sw1;
                        end
                  cancel: begin//to cancel input
                                    swcount<=3'b0;
                                    display1<=six;
                                    display2<=four;
                                    state<=swcountcheck;
                               end
                  default: state<=check_key;
            endcase
         end
next_sw1: begin
            display1<=two;
            display2<=three;
            case (sw)
                  idle: state<=next_sw1;
                  B     : begin
                              state<=next_sw1;
                        end
                  C     : begin
                              swcount[2]<=1'b1;
                              state<=swcountcheck;
                        end
                  cancel: begin
                  //to cancel input n off alarm
                                    swcount<=3'b0;
                                    display1<=six;
                                    display2<=five;
                                    state<=swcountcheck;
                               end
                  default: state<=check_key;
            endcase
          end

swcountcheck: begin
                  display1<=three;
                  display2<=two;
                  if (sw==C)
                        state<=swcountcheck;
                  else
                  begin
                  case (swcount)
                        3'b111: begin

swcount<=3'b0;//clear temp

state<=waitmanlock;
                                          end
```

```
                                    default: begin
                                                      state<=check_key;
                                                end
                              endcase
                           end end
      waitmanlock: begin
                              if (!man_lock)
                                    state<=waitmanlock;
                              //wait till it opens
                              else
                                    state<=waitlock;
                        end

      waitlock: begin
                        display1<=three;
                        display2<=three;
                        case (man_lock)
            //consider lock and manual lock are functioning 2gether
                              1'b0: begin
                                    lock<=1'b0;
                                    light<=1'b1;
                                    disarm_alarm<=1'b1;
                                    state<=check_key;
                                 end

                              1'b1: begin
                                    lock<=1'b1;
                                    swcount<=3'b0;
                                    light<=1'b0;
                                    disarm_alarm<=1'b0;
                                    state<=waitlock;
                                    end
                        endcase
                  end

      default: state<=check_key;




endcase
end end
endmodule
```

Appendix B

RTL Viewer

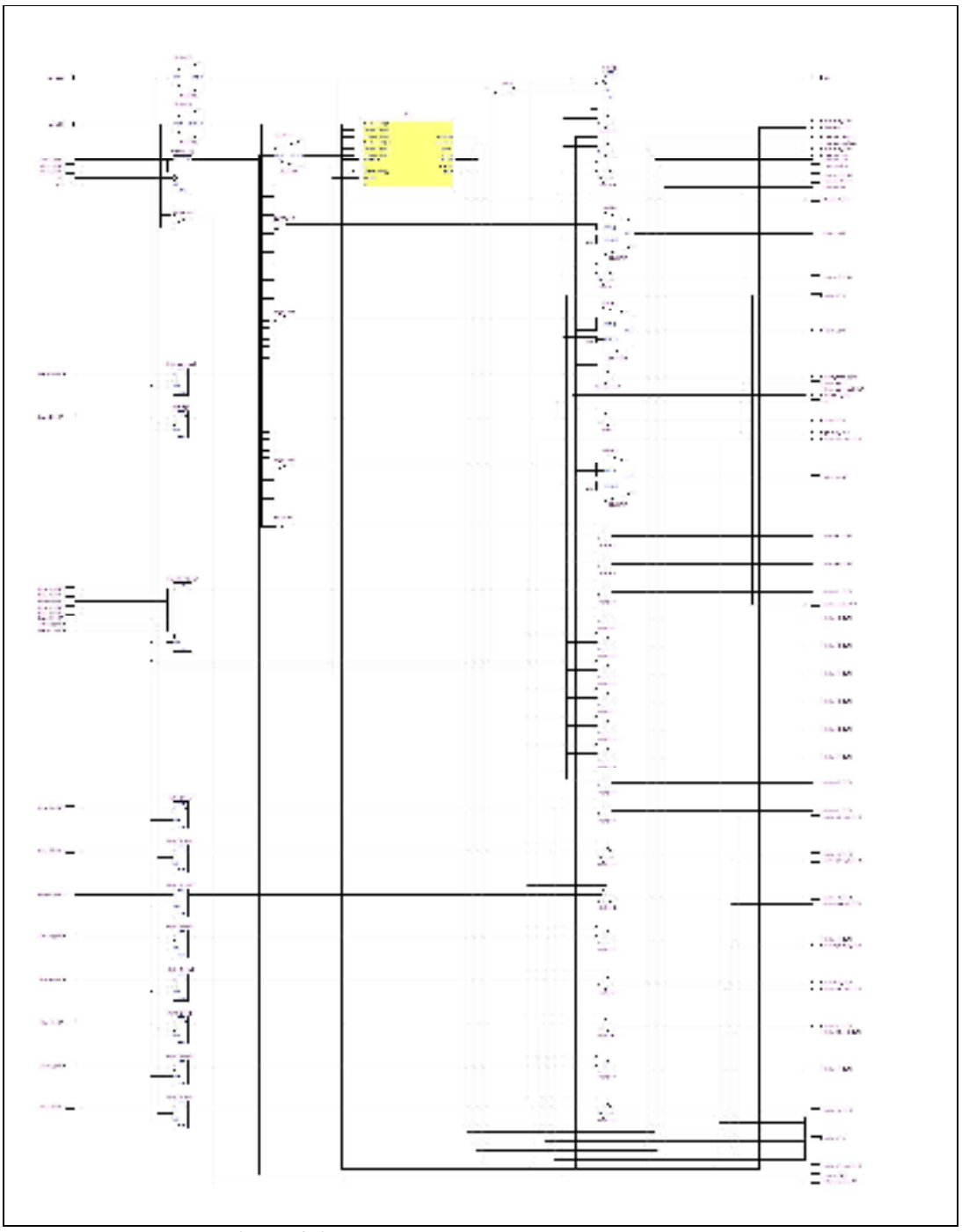

Figure A: RTL View of the Emergency Door Car Entry System

Appendix A

Source Code

```
/*Design by Zaini bin Sulaiman
  Final Year Project 2007
  Keyless Auto Entry System
      -to overcome prob where user left key inside the car and grand
theft auto
  Supervisor: Pn. Norhawati
*/
/*Assumptions:
      -when opened by switch, there is no alarm(coz if open car wif no
key,
      there'll b alarm)
      -switch can only open

      -alarm can be off by key or switch combos
*/
/*Initial condition of the system
      -there is no key(key=1)
      -manual lock is locked (man_lock=0)
      -system is locked
*/

////////////////////////////////////top level
design//////////////////////////////////////
module version_11
(man_lock,sw,clk,display1,display2,lock,key,disarm_alarm,light);
//////////////////////////////////ports
declarations//////////////////////////////////

input clk,key,man_lock;
input [3:0]sw;


output [7:0]display1,display2;
output lock,disarm_alarm,light;


reg lock,disarm_alarm,light;
reg [3:0]state;
reg [2:0]swcount;
reg [7:0]display1,display2;


parameter zero=8'b00000011,
              one=8'b10011111,
              two=8'b00100101,
              three=8'b00001101,
              four=8'b10011001,
              five=8'b01001001,
              six=8'b01000001,
              seven=8'b00011111,
              eight=8'b00000001,
              nine=8'b00001001;
```

```
parameter A=4'b0111,
              B=4'b1011,
              C=4'b1101,
              cancel=4'b1110,
              idle=4'b1111;
parameter check_key=1,
              check_init=2,
              next_sw=3,
              next_sw1=4,
              next_sw2=5,
              swcountcheck=6,
              waitlock=7,
              waitmanlock=8;


//////////////////////////////////////body//////////////////////////////
///////////////
always @ (posedge clk)//trigger states at positive edge clock
begin

//////////////////////////////external input check
routine////////////////////////////////
if (!key)
      state<=check_key;
else begin
case (state)
      check_key: begin
                        if (!key)
                              begin
                                    swcount<=3'b0;
                                    display1<=six;
                                    display2<=six;
                              end
                        else
                        begin
                              swcount<=3'b0;
                              state<=check_init;
                        end
                        end

      check_init: begin
                              display1<=one;
                              display2<=one;
                              case (sw)
                                    idle  : begin
                                                      state<=check_key;
                                                end
                                    cancel: begin//to cancel input n
off alarm
                                                      swcount<=3'b000;
                                                      display1<=six;
                                                      display2<=three;
                                                      light<=1'b1;

      disarm_alarm<=1'b1;
                                                      state<=check_key;
                                                end
```

```
                                    A           : begin
                                                      swcount[0]<=1'b1;
                                                      swcount[1]<=1'b0;
                                                      swcount[2]<=1'b0;
                                                      state<=next_sw;
                                                end
                                    default: state<=check_key;
                              endcase
                        end
      next_sw: begin
                        display1<=two;
                        display2<=one;
                        case (sw)
                              idle: state<=next_sw;
                              A     : begin
                                          state<=next_sw;
                                      end
                              B     : begin
                                          swcount[1]<=1'b1;
                                          state<=next_sw1;
                                      end
                              cancel: begin//to cancel input
                                                swcount<=3'b0;
                                                display1<=six;
                                                display2<=four;
                                                state<=swcountcheck;
                                               end
                              default: state<=check_key;
                        endcase
                  end
      next_sw1: begin
                        display1<=two;
                        display2<=three;
                        case (sw)
                              idle: state<=next_sw1;
                              B     : begin
                                          state<=next_sw1;
                                      end
                              C     : begin
                                          swcount[2]<=1'b1;
                                          state<=swcountcheck;
                                      end
                              cancel: begin//to cancel input n off
alarm
                                                swcount<=3'b0;
                                                display1<=six;
                                                display2<=five;
                                                state<=swcountcheck;
                                               end
                              default: state<=check_key;
                        endcase
                   end

      swcountcheck: begin
                              display1<=three;
                              display2<=two;
                              if (sw==C)
```

```
                                    state<=swcountcheck;
                                else
                                begin
                                case (swcount)
                                    3'b111: begin

      swcount<=3'b0;//clear temp

      state<=waitmanlock;
                                                    end

                                    default: begin
                                                        state<=check_key;
                                                    end
                                endcase
                              end end
      waitmanlock: begin
                                if (!man_lock)
                                    state<=waitmanlock;//wait till it
opens
                                else
                                    state<=waitlock;
                             end

      waitlock: begin
                           display1<=three;
                           display2<=three;
                           case (man_lock)//consider lock and manual lock
are functioning 2gether
                                1'b0: begin
                                    lock<=1'b0;
                                    light<=1'b1;
                                    disarm_alarm<=1'b1;
                                    state<=check_key;
                                   end

                                1'b1: begin
                                    lock<=1'b1;
                                    swcount<=3'b0;
                                    light<=1'b0;
                                    disarm_alarm<=1'b0;
                                    state<=waitlock;
                                    end
                           endcase
                      end

      default: state<=check_key;




endcase
end end
endmodule
```

Appendix B

RTL Viewer

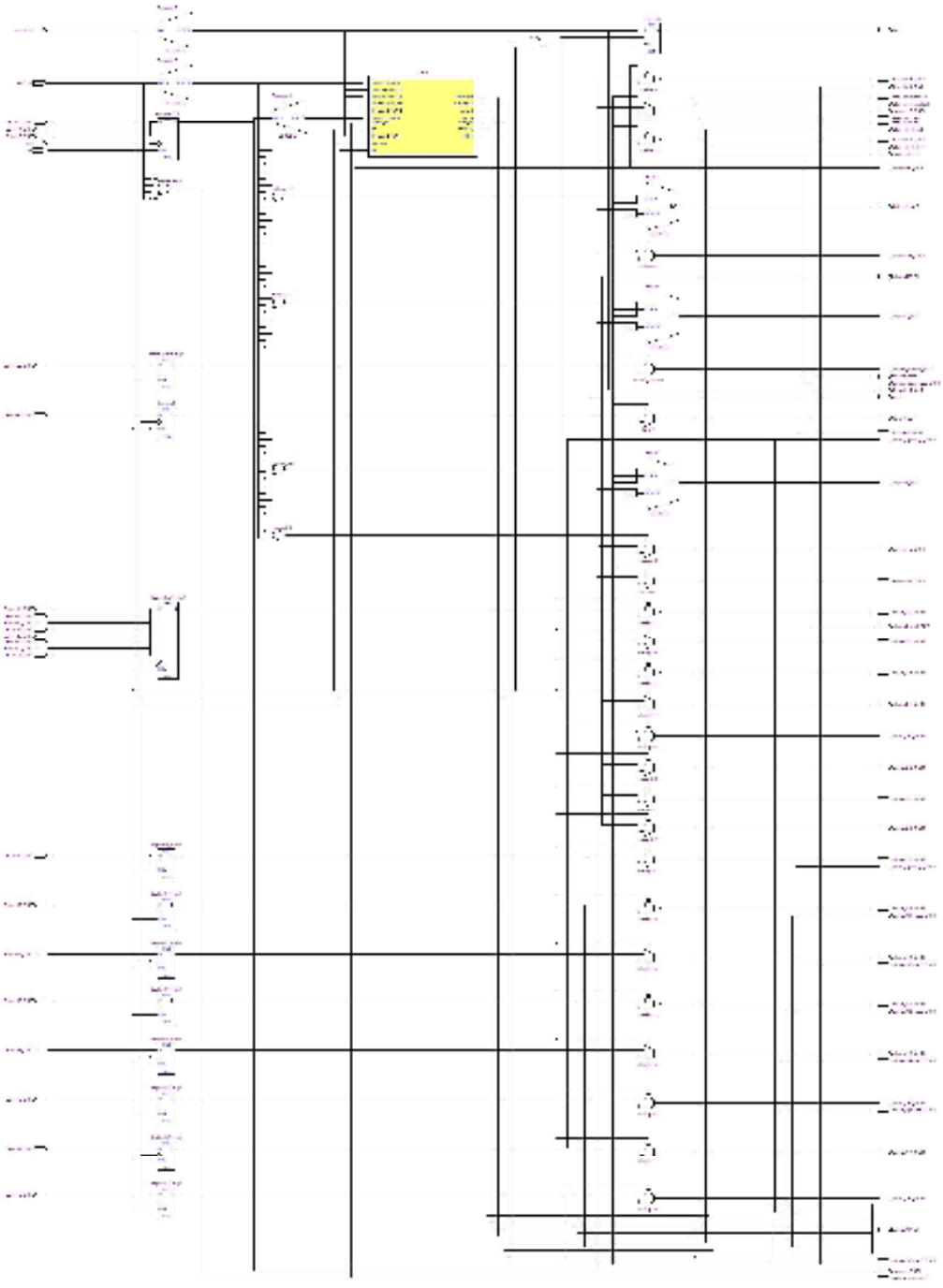

Figure: RTL View of the Emergency Door Car Entry System