รายการอ้างอิง

[1] สายัณห์ ธีรปัญญาวัฒน์, การพัฒนาเครื่องรับเทเลเท็กซต์แสดงผลตัวอักษรไทย/อังกฤษ, บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2536

[2] วันเฉลิม โปรา, แบบจำลองภาษาวีเอชดีแอลของวงจรรวมที่ใช้กับเทเลเท็กซต์, บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2538

[3] โครงการวิจัยเรื่อง เครื่องถอดรหัสเทเลเท็กซต์ เนคเทค-จุฬา, ห้องปฏิบัติการวิจัย ระบบเชิงเลข ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย, 2539

[4] VHDL Modeling, VIEWlogic Customer Training

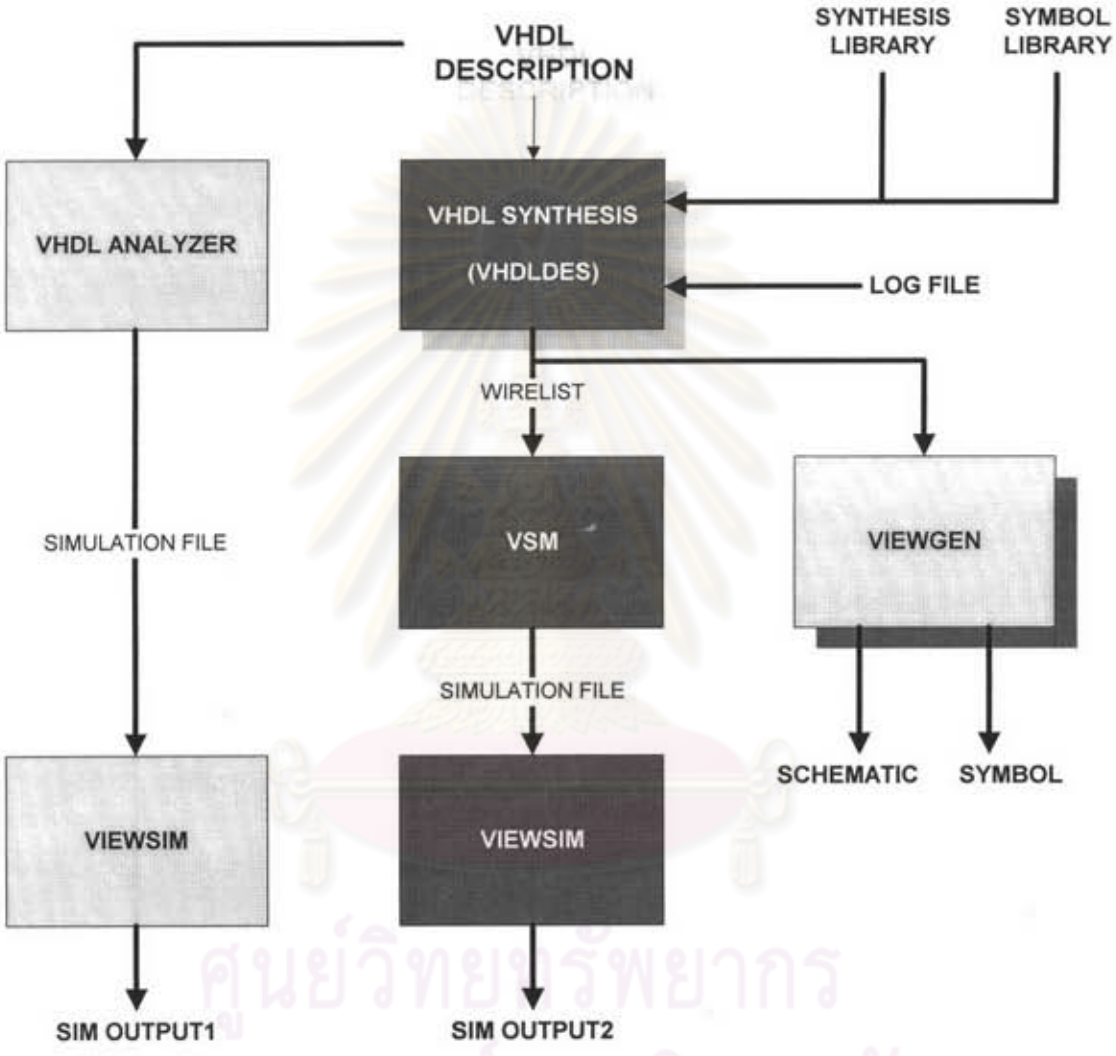
[5] Zainalabedin Navabi, VHDL Analysis and Modeling of Digital System, McGRAW-HILL International Editions, 1993

[6] The Programmable Gate Array Data Book, XILINX, 1992

ภาคผนวก ก

ขั้นตอนในการใช้เครื่องมือต่าง ๆ ในออกแบบที่ใช้แบบจำลองภาษาวีเอชดีแอล

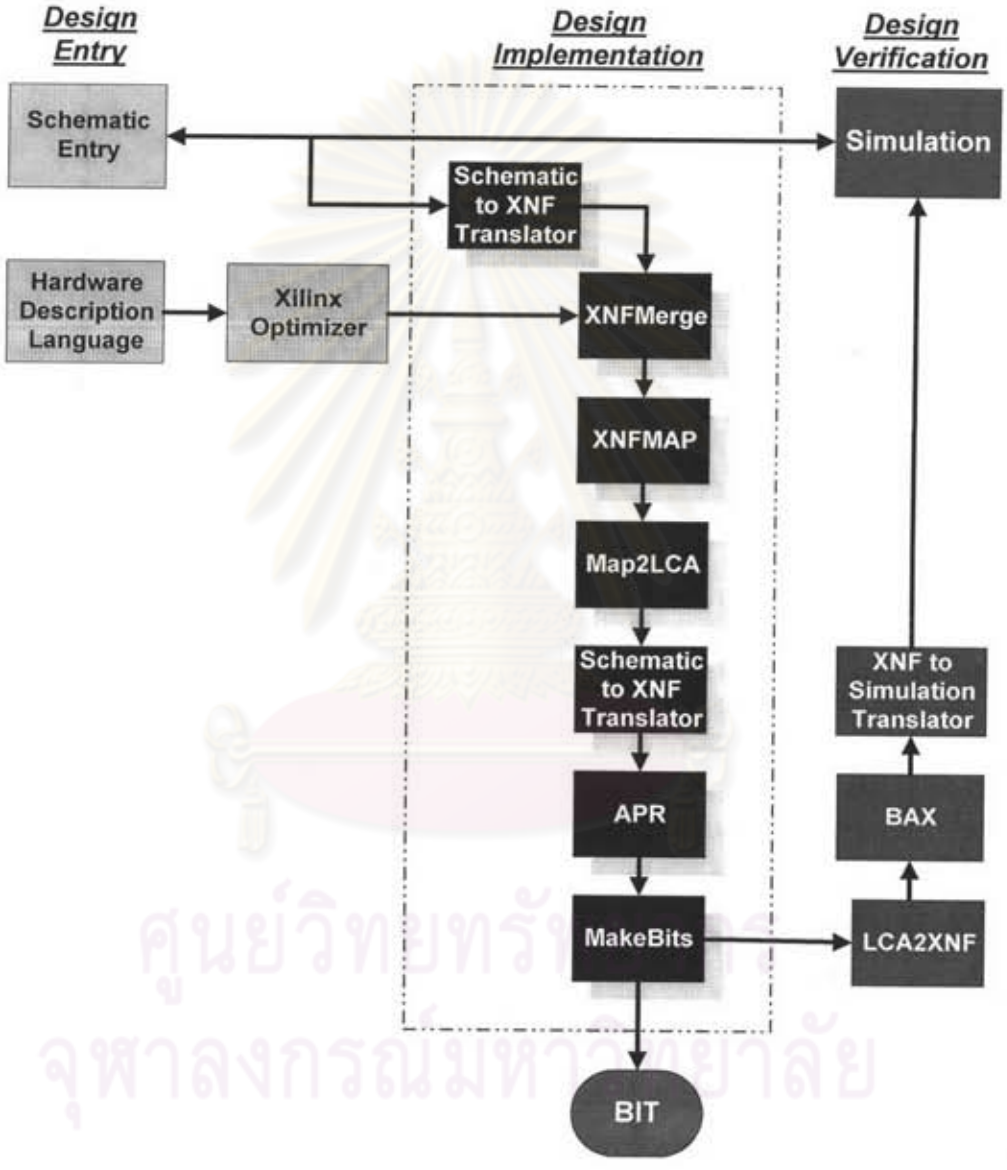
ในการออกแบบวงจรโดยใช้แบบจำลองภาษาวีเอชดีแอลจะมีการใช้ซอฟต์แวร์สองตัวด้วยกัน คือ ซอฟต์แวร์ชื่อ WVPLUS ของบริษัท VIEWlogic และ XACT ของบริษัท XILINX ซึ่งจะขออธิบายตามโฟลว์ ดังที่แสดงในรูปที่ ก.1 และรูปที่ ก.2



รูปที่ ก.1 Design Flow

จากตัวภาษาวีเอชดีแอลที่เขียนขึ้นจะมีเครื่องมือ คือ VHDL Analyzer เป็นตัวตรวจสอบการเขียนว่าถูกต้องตาม syntax ของวีเอชดีแอลหรือไม่ แล้วจึงใช้ ViewSim เป็นเครื่องมือในการทดสอบการจำลองการทำงานของภาษาวีเอชดีแอลที่เขียนขึ้นว่ามีการทำงานของสัญญาณถูกต้องตามที่ออกแบบไว้หรือไม่ หลังจากที่ได้ตัวภาษาวีเอชดีแอลถูกต้องตามที่ต้องการก็จะตัวภาษาวีเอชดีแอลไปออกแบบ schematics และ symbol โดยมีขั้นตอนดังนี้คือ ใช้เครื่องมือที่ชื่อ VHDL Synthesis ซึ่งจะได้

ไฟล์ .wir เกิดขึ้นมา ซึ่งไฟล์นี้จะสามารถนำไปสร้างเป็น schematic โดยใช้เครื่องมือที่ชื่อ ViewGen แต่จากไฟล์ .wir เราสามารถนำไปสร้างเป็นไฟล์ที่ใช้สำหรับจำลองการทำงานได้อีกโดยใช้เครื่องมือชื่อ VSM จะได้ไฟล์ .vsm ซึ่งสามารถนำไปใช้จำลองการทำงานโดยใช้เครื่องมือชื่อ ViewSim อีกครั้งหนึ่งได้



รูปที่ ก.2 Design Implement

จากโฟลว์ในรูปที่ ก.2 เราจะนำ schematic ที่สร้างขึ้นจากขบวนการตามโฟลว์ในรูปที่ ก.1 มาทำการสร้างไฟล์ .bit เพื่อในไปโปรแกรมชิป FPGA โดยเริ่มจากการนำ schematic มาแปลงให้เป็น XNF (Xilinx Netlist File) โดยใช้เครื่องมือ XNF แล้วจึงนำไปผ่านขบวนการ Map โดยใช้เครื่องมือ XNFMap

ซึ่งจะได้ไฟล์ .map เพื่อนำไปสร้างไฟล์ .lca หรือ Logic Cell Array โดยใช้เครื่องมือ Map2LCA (Converts a MAP file to a LCA file) ต่อไปก็จะนำไฟล์ .scp และ .lca ที่ได้ไปทำการรูท (route) โดยใช้เครื่องมือ APR (Automatically places and routes) แล้วจึงนำไฟล์ .lca ที่ได้จากการรูทไปทำการสร้าง bitstream สำหรับชิป FPGA โดยใช้เครื่องมือ MakeBits (Generates FPGA bitstream) จะได้ไฟล์ .bit เป็นไฟล์ที่นำไปใช้โปรแกรมชิป FPGA โดยใช้เครื่องมือ Xchecker สำหรับการโหลดโปรแกรมผ่านสาย Serial port เป็นขั้นตอนสุดท้าย

ภาคผนวก ข

รายละเอียดการเขียนโปรแกรมด้วยภาษาวีเอชดีแอล

```
ENTITY div2 IS
PORT (
      clk13_875  : in  vlbit;
      clk6_9375  : out vlbit
     );
END div2;
ARCHITECTURE behavioral OF div2 IS
SIGNAL  tmp_clk       : vlbit;
BEGIN

      PROCESS
      BEGIN
      WAIT UNTIL PRISING(clk13_875);
      tmp_clk <= NOT(tmp_clk);
      END PROCESS;

      clk6_9375 <= tmp_clk;

END behavioral;
```

```
ENTITY t_versyn IS
PORT (
      composite              : in  vlbit;
      vertical               : in  vlbit;
      tmp_vert_sync          : out vlbit
     );
END t_versyn;
ARCHITECTURE behavioral OF t_versyn IS
SIGNAL   cnt                 : vlbit_1d(4 downto 0);
BEGIN
      PROCESS
      BEGIN
      WAIT UNTIL (PFALLING(composite) OR (vertical = '1'));
      IF vertical = '1' THEN
         cnt <= "00000";
      ELSE
         cnt <= ADDUM(cnt(3 downto 0),"001");
      END IF;
      END PROCESS;

      PROCESS(vertical,cnt)
      BEGIN
      IF vertical = '1' THEN
         tmp_vert_sync <= '1';
      ELSE
         IF cnt(3 downto 0) < B"0001" THEN
            tmp_vert_sync <= '1';
         ELSE
            IF cnt(3 downto 0) <= B"0010" THEN
             tmp_vert_sync <= '0';
            ELSE
             tmp_vert_sync <= '1';
            END IF;
         END IF;
      END IF;
      END PROCESS;

END behavioral;
```

```
ENTITY t_slice IS
PORT (
      vertical          : in  vlbit;
      composite         : in  vlbit;
      clk13_875         : in  vlbit;
      clk6_9375         : in  vlbit;
      selline           : in  vlbit_1d(15 downto 0);
      end_page          : in  vlbit;
      intrpt            : out vlbit;
      sel_rom_ram       : out vlbit;
      text_enable       : out vlbit;
      shift_clk         : out vlbit;
      load_add          : out vlbit
     );
END t_slice;

ARCHITECTURE behavioral OF t_slice IS
SIGNAL     counter                : vlbit_1d(9 downto 0);
SIGNAL     sft_data_enable        : vlbit;
SIGNAL     enable                 : vlbit;
SIGNAL     l1                   : vlbit;
SIGNAL     l2                   : vlbit;
SIGNAL     l3                   : vlbit;
SIGNAL     l4                   : vlbit;
SIGNAL     l5                   : vlbit;
SIGNAL     l6                   : vlbit;
SIGNAL     l7                   : vlbit;
SIGNAL     l8                   : vlbit;
SIGNAL     l9                   : vlbit;
SIGNAL     l10                  : vlbit;
SIGNAL     l11                  : vlbit;
SIGNAL     l12                  : vlbit;
SIGNAL     l13                  : vlbit;
SIGNAL     l14                  : vlbit;
SIGNAL     l15                  : vlbit;
SIGNAL     l16                  : vlbit;
SIGNAL     serial               : vlbit;
SIGNAL     sft_data             : vlbit;
SIGNAL     enable_on            : vlbit;

BEGIN

COUNT_LINE:BLOCK
BEGIN

      PROCESS
      BEGIN
      WAIT UNTIL PFALLING(composite);
      IF (vertical = '0') THEN
          counter <= "0000000000";
      ELSE
          counter <= ADDUM(counter(8 downto 0),"000000001");
      END IF;
      END PROCESS;
```

```
        PROCESS
        BEGIN
        WAIT UNTIL PFALLING(vertical);
           enable <= '1';
        END PROCESS;

        PROCESS
        BEGIN
        WAIT UNTIL PRISING(vertical);
        IF enable = '1' THEN
           enable_on <= '1';
        ELSE
           enable_on <= '0';
        END IF;
        END PROCESS;

END BLOCK COUNT_LINE;

IN_LINE_SELECT:BLOCK
SIGNAL     counter_in_line        : vlbit_1d(9 downto 0);
SIGNAL     slice_enable           : vlbit;
SIGNAL     tmp_intrpt             : vlbit;
SIGNAL  in_line_en                : vlbit;
SIGNAL  tmp_load_add              : vlbit;
SIGNAL  line_en                   : vlbit;

BEGIN

      PROCESS(selline,counter)
      BEGIN
         IF selline(0) = '0' THEN
           l1 <= '0';
         ELSE
            IF counter(8 downto 0) = B"000001000" THEN
               l1 <= '1';
            ELSE
               l1 <= '0';
            END IF;
         END IF;
      END PROCESS;

      PROCESS(selline,counter)
      BEGIN
        IF selline(1) = '0' THEN
           l2 <= '0';
         ELSE
            IF counter(8 downto 0) = B"000001001" THEN
               l2 <= '1';
            ELSE
               l2 <= '0';
            END IF;
         END IF;
      END PROCESS;

      PROCESS(selline,counter)
```

```
BEGIN
   IF selline(2) = '0' THEN
      l3 <= '0';
   ELSE
      IF counter(8 downto 0) = B"000001010" THEN
         l3 <= '1';
      ELSE
         l3 <= '0';
      END IF;
   END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
   IF selline(3) = '0' THEN
      l4 <= '0';
   ELSE
      IF counter(8 downto 0) = B"000001011" THEN
         l4 <= '1';
      ELSE
         l4 <= '0';
      END IF;
   END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
   IF selline(4) = '0' THEN
      l5 <= '0';
   ELSE
      IF counter(8 downto 0) = B"000001100" THEN
         l5 <= '1';
      ELSE
         l5 <= '0';
      END IF;
   END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
   IF selline(5) = '0' THEN
      l6 <= '0';
   ELSE
      IF counter(8 downto 0) = B"000001101" THEN
         l6 <= '1';
      ELSE
         l6 <= '0';
      END IF;
   END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
   IF selline(6) = '0' THEN
     l7 <= '0';
```

```
    ELSE
       IF counter(8 downto 0) = B"000001110" THEN
          l7 <= '1';
       ELSE
          l7 <= '0';
       END IF;
    END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
   IF selline(7) = '0' THEN
      l8 <= '0';
   ELSE
      IF counter(8 downto 0) = B"000001111" THEN
         l8 <= '1';
      ELSE
         l8 <= '0';
      END IF;
   END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
   IF selline(8) = '0' THEN
      l9 <= '0';
   ELSE
      IF counter(8 downto 0) = B"000010000" THEN
         l9 <= '1';
      ELSE
         l9 <= '0';
      END IF;
   END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
   IF selline(9) = '0' THEN
      l10 <= '0';
   ELSE
      IF counter(8 downto 0) = B"000010001" THEN
         l10 <= '1';
      ELSE
         l10 <= '0';
      END IF;
   END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
   IF selline(10) = '0' THEN
      l11 <= '0';
   ELSE
      IF counter(8 downto 0) = B"000010010" THEN
         l11 <= '1';
```

```
        ELSE
            l11 <= '0';
        END IF;
    END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
    IF selline(11) = '0' THEN
        l12 <= '0';
    ELSE
        IF counter(8 downto 0) = B"000010011" THEN
            l12 <= '1';
        ELSE
            l12 <= '0';
        END IF;
    END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
    IF selline(12) = '0' THEN
        l13 <= '0';
    ELSE
        IF counter(8 downto 0) = B"000010100" THEN
            l13 <= '1';
        ELSE
            l13 <= '0';
        END IF;
    END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
    IF selline(13) = '0' THEN
        l14 <= '0';
    ELSE
        IF counter(8 downto 0) = B"000010101" THEN
            l14 <= '1';
        ELSE
            l14 <= '0';
        END IF;
    END IF;
END PROCESS;

PROCESS(selline,counter)
BEGIN
    IF selline(14) = '0' THEN
        l15 <= '0';
    ELSE
        IF counter(8 downto 0) = B"000010110" THEN
            l15 <= '1';
        ELSE
            l15 <= '0';
        END IF;
```

```
      END IF;
   END PROCESS;

   PROCESS(selline,counter)
   BEGIN
      IF selline(15) = '0' THEN
         l16 <= '0';
      ELSE
         IF counter(8 downto 0) = B"000010111" THEN
            l16 <= '1';
         ELSE
            l16 <= '0';
         END IF;
      END IF;
   END PROCESS;

line_en <= l1 OR l2 OR l3 OR l4 OR l5 OR l6 OR l7 OR l8 OR l8
OR l9 OR l10 OR l11 OR l12 OR l13 OR l14 OR l15 OR l16;

sft_data <= line_en AND enable_on ;

   PROCESS(counter,enable_on)
   BEGIN
   IF enable_on = '1' THEN
      IF counter(8 downto 0) >= B"000001000" THEN
         IF counter(8 downto 0) <= B"000010111" THEN
            tmp_intrpt <= '1';
         ELSE
            tmp_intrpt <= '0';
         END IF;
      ELSE
         tmp_intrpt <= '0';
      END IF;
   ELSE
      tmp_intrpt <= '0';
   END IF;
   END PROCESS;

intrpt <= tmp_intrpt;

slice_enable <= (sft_data AND composite);

   PROCESS
   BEGIN
   WAIT UNTIL PRISING(clk6_9375);
   IF slice_enable = '0' THEN
      counter_in_line <= "0000000000";
   ELSE
      counter_in_line <= ADDUM(counter_in_line(8 downto 0),
"000000001");
   END IF;
   END PROCESS;

   PROCESS(counter_in_line)
   BEGIN
```

```
      IF counter_in_line(8 downto 0) >= B"000000010" THEN --2
         IF counter_in_line(8 downto 0) <= B"000011111" THEN --
31
            tmp_load_add <= '1';
         ELSE
            tmp_load_add <= '0';
         END IF;
      ELSE
         tmp_load_add <= '0';
      END IF;
      END PROCESS;

      PROCESS --(counter_in_line,slice_enable,end_page)
      BEGIN
      WAIT UNTIL PFALLING(clk13_875);
      IF end_page = '1' THEN
         IF tmp_load_add = '1' THEN
            load_add <= '1';
         ELSE
            load_add <= '0';
         END IF;
      ELSE
         load_add <= '0';
      END IF;
      END PROCESS;

      PROCESS(end_page,counter_in_line)
      BEGIN
      IF end_page = '1' THEN
         IF counter_in_line(8 downto 0) = B"000000001" THEN
            sel_rom_ram <= '1' ;
         ELSE
            sel_rom_ram <= '0' ;
         END IF;
      ELSE
         sel_rom_ram <= '0';
      END IF;
      END PROCESS;

      PROCESS(counter_in_line)
      BEGIN                                          --32
      IF counter_in_line(8 downto 0) >= B"000100000" THEN
         IF counter_in_line(8 downto 0) <= B"110000111" --391
            in_line_en <= '1';
         ELSE
            in_line_en <= '0';
         END IF;
      ELSE
         in_line_en <= '0';
      END IF;
      END PROCESS;

      PROCESS
      BEGIN
      WAIT UNTIL PFALLING(clk13_875);
```

```
      IF in_line_en = '1' THEN --serial = '1'
         IF clk6_9375 = '1' THEN
            shift_clk <= '1';
         ELSE
            shift_clk <= '0';
         END IF;
      ELSE
         shift_clk <= '0';
      END IF;
      END PROCESS;

text_enable <= slice_enable;

END BLOCK IN_LINE_SELECT;

END behavioral;
```

```
ENTITY t_mult IS
PORT (
      text_enable           : in  vlbit;
      empty                 : in  vlbit;
      clk                   : in  vlbit;
      load_or_shift         : out vlbit;
      add_addrs             : out vlbit
      );
END t_mult;

ARCHITECTURE PRE_PISO OF t_mult IS
SIGNAL   load_sft_cnt      : vlbit_1d(3 downto 0);
SIGNAL   enable_load       : vlbit;
SIGNAL   cnt_load          : vlbit_1d(6 downto 0);
SIGNAL   tmp_load_shift    : vlbit;
SIGNAL   enable_add        : vlbit;

BEGIN

LOAD_SHIFT:BLOCK
BEGIN
      PROCESS
      BEGIN
      WAIT UNTIL (PRISING(clk) OR (text_enable = '0'));
      IF text_enable = '0' THEN
         load_sft_cnt <= "0000";
      ELSE
         load_sft_cnt <= ADDUM(load_sft_cnt(2 downto 0),"001");
      END IF;
      END PROCESS;

      PROCESS
      BEGIN
      WAIT UNTIL (PRISING(tmp_load_shift) OR (text_enable =
'0'));
      IF text_enable = '0' THEN
         cnt_load <= "0000000";
      ELSE
         cnt_load <= ADDUM(cnt_load(5 downto 0),"0001");
      END IF;
      END PROCESS;

      PROCESS(cnt_load)
      BEGIN
         IF cnt_load(5 downto 0) <= B"101111" THEN --47
            enable_load <= '1';
         ELSE
            enable_load <= '0';
         END IF;
      END PROCESS;

      PROCESS(load_sft_cnt,enable_load)
      BEGIN
         IF enable_load = '1' THEN
            IF load_sft_cnt(2 downto 0) = B"001" THEN
```

```
            tmp_load_shift <= '1';
         ELSE
            tmp_load_shift <= '0';
         END IF;
      ELSE
         tmp_load_shift <= '0';
      END IF;
   END PROCESS;

   load_or_shift <= tmp_load_shift;

   PROCESS(cnt_load)
   BEGIN
   IF cnt_load(5 downto 0) >= B"000100" THEN --4
      enable_add <= '1';
   ELSE
      enable_add <= '0';
   END IF;
   END PROCESS;

   PROCESS(load_sft_cnt,enable_load,enable_add,empty)
   BEGIN
   IF enable_load = '1' THEN
      IF empty = '0' THEN
         IF enable_add = '1' THEN
            IF load_sft_cnt(2 downto 0) = B"011" THEN  --3
               add_addrs <= '1';
            ELSE
               add_addrs <= '0';
            END IF;
         ELSE
            add_addrs <= '0';
         END IF;
      ELSE
         add_addrs <= '0';
      END IF;
   ELSE
      add_addrs <= '0';
   END IF;
   END PROCESS;
END BLOCK LOAD_SHIFT;

END PRE_PISO;
```

```
ENTITY t_cntlin IS
PORT (
      clk_line_cnt      : in  vlbit;
      end_page          : out vlbit
     );
END t_cntlin;
ARCHITECTURE layout OF t_cntlin IS
SIGNAL   data           : vlbit_1d (5 downto 0);
SIGNAL   tmp_clr        : vlbit;
BEGIN

      PROCESS
      BEGIN
      WAIT UNTIL (PRISING(clk_line_cnt) OR (tmp_clr = '1'));
      IF tmp_clr = '1' THEN
         data <= "000000";
      ELSE
         data <= ADDUM(data(4 downto 0),"00001");
      END IF;
      END PROCESS;

      PROCESS(data)
      BEGIN
      IF data(4 downto 0) = B"00001" THEN
         end_page <= '1';
      ELSE
         end_page <= '0';
      END IF;
      END PROCESS;

      PROCESS(data)
      BEGIN
      IF data(4 downto 0) = B"11001" THEN --25
         tmp_clr <= '1';
      ELSE
         tmp_clr <= '0';
      END IF;
      END PROCESS;

END layout;
```

```
ENTITY t_addrs2 IS
PORT  (
      load              : in  vlbit;
      clk6_9375         : in  vlbit;
      clk_cnt_addrs     : in  vlbit;
      empty_data        : in  vlbit;
      init_up           : in  vlbit;
      update            : in  vlbit;
      address           : out vlbit_1d(19 downto 0)
      );
END t_addrs2;
ARCHITECTURE behavioral OF t_addrs2 IS
SIGNAL     init_addrs : vlbit_1d(20 downto 0);
SIGNAL     tmp_addrs  : vlbit_1d(20 downto 0);

BEGIN
      PROCESS
      BEGIN
      WAIT UNTIL PFALLING(clk6_9375);
      IF load = '1' THEN
         IF update = '1' THEN
            IF empty_data = '1' THEN
               init_addrs <= "00000000000000000000";
            ELSE
               init_addrs <= ADDUM(tmp_addrs(19 downto 0),
"0001");
            END IF;
         END IF;
      END IF;
      END PROCESS;

      PROCESS
      BEGIN
      WAIT UNTIL PFALLING(clk6_9375);
      IF clk_cnt_addrs = '1' THEN
         IF init_up = '1' THEN
            IF empty_data = '1' THEN
               tmp_addrs <= "00000000000000000000";
            ELSE
               tmp_addrs(19 downto 0) <= init_addrs(19 downto
0);
            END IF;
         ELSE
            tmp_addrs <= ADDUM(tmp_addrs(19 downto 0),"0001");
         END IF;
      END IF;
      END PROCESS;

      address(19 downto 16) <= NOT(tmp_addrs(19 downto 16));
      address(15 downto 0)  <= tmp_addrs(15 downto 0);

END behavioral;
```

```
ENTITY t_addrs3 IS
PORT  (
      load              : in  vlbit;
      clk6_9375         : in  vlbit;
      data              : in  vlbit_1d(7 downto 0);
      clk_cnt_addrs     : out vlbit;
      load_addrs        : out vlbit;
      update            : out vlbit;
      sel_addrs         : out vlbit;
      init_up           : out vlbit;
      empty_data        : out vlbit;
      reg_addrs         : out vlbit_1d(19 downto 0)
      );
END t_addrs3;
ARCHITECTURE behavioral OF t_addrs3 IS
SIGNAL  cnt             : vlbit_1d(5 downto 0);
SIGNAL  init            : vlbit;
SIGNAL  tmp_clk_rd      : vlbit;
SIGNAL  tmp_reg_addrs : vlbit_1d(19 downto 0);
SIGNAL  cntone          : vlbit;
SIGNAL  cnttwo          : vlbit;
SIGNAL  cntthree        : vlbit;

BEGIN

      PROCESS
      BEGIN
      WAIT UNTIL PRISING(clk6_9375);
          IF load = '0' THEN
             cnt <= "000000";
          ELSE
             cnt <= ADDUM(cnt(4 downto 0),"0001");
          END IF;
      END PROCESS;

      PROCESS(cnt)
      BEGIN
           IF cnt(4 downto 0) = B"00010" THEN    --2
            load_addrs <= '0';
           ELSE
           IF cnt(4 downto 0) = B"01010" THEN    --10
            load_addrs <= '0';
          ELSE
             IF cnt(4 downto 0) = B"10010" THEN       --18
                load_addrs <= '0';
             ELSE
                IF cnt(4 downto 0) = B"11100" THEN    --28
                 load_addrs <= '0';
                ELSE
                 load_addrs <= '1';
                END IF;
             END IF;
          END IF;
           END IF;
     END PROCESS;
```

```
PROCESS(cnt)
BEGIN
IF cnt(4 downto 0) = B"00001" THEN          --1
   clk_cnt_addrs <= '1';
ELSE
   IF cnt(4 downto 0) = B"01001" THEN       --9
      clk_cnt_addrs <= '1';
   ELSE
      IF cnt(4 downto 0) = B"10001" THEN         --17
         clk_cnt_addrs <= '1';
      ELSE
         clk_cnt_addrs <= '0';
      END IF;
   END IF;
END IF;
END PROCESS;

PROCESS(cnt)
BEGIN
IF cnt(4 downto 0) = B"10100" THEN          --20
   update <= '1';
ELSE
   update <= '0';
END IF;
END PROCESS;

PROCESS(cnt)
BEGIN
IF cnt(4 downto 0) = B"00001" THEN          --1
   init_up <= '1';
ELSE
   init_up <= '0';
END IF;
END PROCESS;

PROCESS(cnt)
BEGIN
IF cnt(4 downto 0) < B"11001" THEN          --25
   sel_addrs <= '1';
ELSE
   sel_addrs <= '0';
END IF;
END PROCESS;

PROCESS(cnt)
BEGIN
IF cnt(4 downto 0) = B"00111" THEN          --7
   cntone <= '1';
ELSE
   cntone <= '0';
END IF;
END PROCESS;

PROCESS(cnt)
BEGIN
```

```
        IF cnt(4 downto 0) = B"01111" THEN          --15
           cnttwo <= '1';
        ELSE
           cnttwo <= '0';
        END IF;
        END PROCESS;

        PROCESS(cnt)
        BEGIN
        IF cnt(4 downto 0) = B"10111" THEN          --23
           cntthree <= '1';
        ELSE
           cntthree <= '0';
        END IF;
        END PROCESS;

        PROCESS
        BEGIN
        WAIT UNTIL PFALLING(clk6_9375);
        IF cntone = '1' THEN
           tmp_reg_addrs(7 downto 0) <= data(7 downto 0);
        ELSE
           IF cnttwo = '1' THEN
              tmp_reg_addrs(15 downto 8) <= data(7 downto 0);
           ELSE
               IF cntthree = '1' THEN
                  tmp_reg_addrs(19 downto 16) <= data(3 downto 0);
               END IF;
           END IF;
        END IF;
        END PROCESS;

        reg_addrs <= tmp_reg_addrs;

        PROCESS(tmp_reg_addrs,load)
        BEGIN
        IF load = '1' THEN
           IF tmp_reg_addrs(15 downto 0) = B"0000000000000000"
THEN
              IF tmp_reg_addrs(19 downto 16) = B"0000" THEN
                 empty_data <= '1';
              ELSE
                 empty_data <= '0';
              END IF;
           ELSE
              empty_data <= '0';
           END IF;
        END IF;
        END PROCESS;

END behavioral;
```

```
ENTITY t_addrs4 IS
PORT  (
      load              : in  vlbit;
      sel_addrs         : in  vlbit;
      tmp_addrs         : in  vlbit_1d(19 downto 0);
      reg_addrs         : in  vlbit_1d(19 downto 0);
      address           : out vlbit_1d(19 downto 0)
       );
END t_addrs4;
ARCHITECTURE behavioral OF t_addrs4 IS
BEGIN

      PROCESS(load,sel_addrs,tmp_addrs,reg_addrs)
      BEGIN
         IF load = '0' THEN
            address <= "00000000000000000000";
         ELSE
            IF sel_addrs = '1' THEN
               address <= tmp_addrs(19 downto 0);
            ELSE
               address <= reg_addrs;
            END IF;
         END IF;
      END PROCESS;

END behavioral;
```

```
ENTITY t_ldaddr IS
PORT (
     clk              : in  vlbit;
     load_addrs       : in  vlbit;
     add_addrs        : in  vlbit;
     addrs_in         : in  vlbit_1d(19 downto 0);
     addrs_out        : out vlbit_1d(19 downto 0)
     );
END t_ldaddr;
ARCHITECTURE behavioral OF t_ldaddr IS
SIGNAL   tmp_addrs    : vlbit_1d(20 downto 0);

BEGIN

     PROCESS
     BEGIN
     WAIT UNTIL PRISING(clk);
     IF load_addrs = '0' THEN
        tmp_addrs(20) <= '0';
        tmp_addrs(19 downto 0) <= addrs_in;
     ELSE
        IF add_addrs = '1' THEN
           tmp_addrs <= ADDUM(tmp_addrs(19 downto 0),"0001");
        END IF;
     END IF;
     END PROCESS;

     addrs_out <= tmp_addrs(19 downto 0);

END behavioral;
```

```
ENTITY t_piso IS
PORT (
      clk              : in  vlbit;
      empty            : in  vlbit;
      load_shift       : in  vlbit;
      text_enable      : in  vlbit;
      data_in          : in  vlbit_1d(7 downto 0);
      serial_data_out  : out vlbit
      );
END t_piso;

ARCHITECTURE behavioral OF t_piso IS
SIGNAL  byte_num       : vlbit_1d(6 downto 0);
SIGNAL  reg_data       : vlbit_1d(8 downto 0);

BEGIN

      PROCESS
      BEGIN
      WAIT UNTIL (PFALLING(clk) OR (text_enable = '0'));
      IF text_enable = '0' THEN
         byte_num <= "0000000";
      ELSE
         IF load_shift = '1' THEN
            byte_num <= ADDUM(byte_num(5 downto 0),"000001");
         END IF;
      END IF;
      END PROCESS;

SHIFT_DATA:BLOCK
SIGNAL tmp_reg         : vlbit;
SIGNAL bytezero        : vlbit;
SIGNAL byteone         : vlbit;
SIGNAL bytetwo         : vlbit;

BEGIN

      PROCESS(byte_num)
      BEGIN
      IF byte_num(5 downto 0) = B"000000" THEN
         bytezero <= '1';
      ELSE
         bytezero <= '0';
      END IF;
      END PROCESS;

      PROCESS(byte_num)
      BEGIN
      IF byte_num(5 downto 0) = B"000001" THEN
         byteone <= '1';
      ELSE
         byteone <= '0';
      END IF;
      END PROCESS;
      PROCESS(byte_num)
```

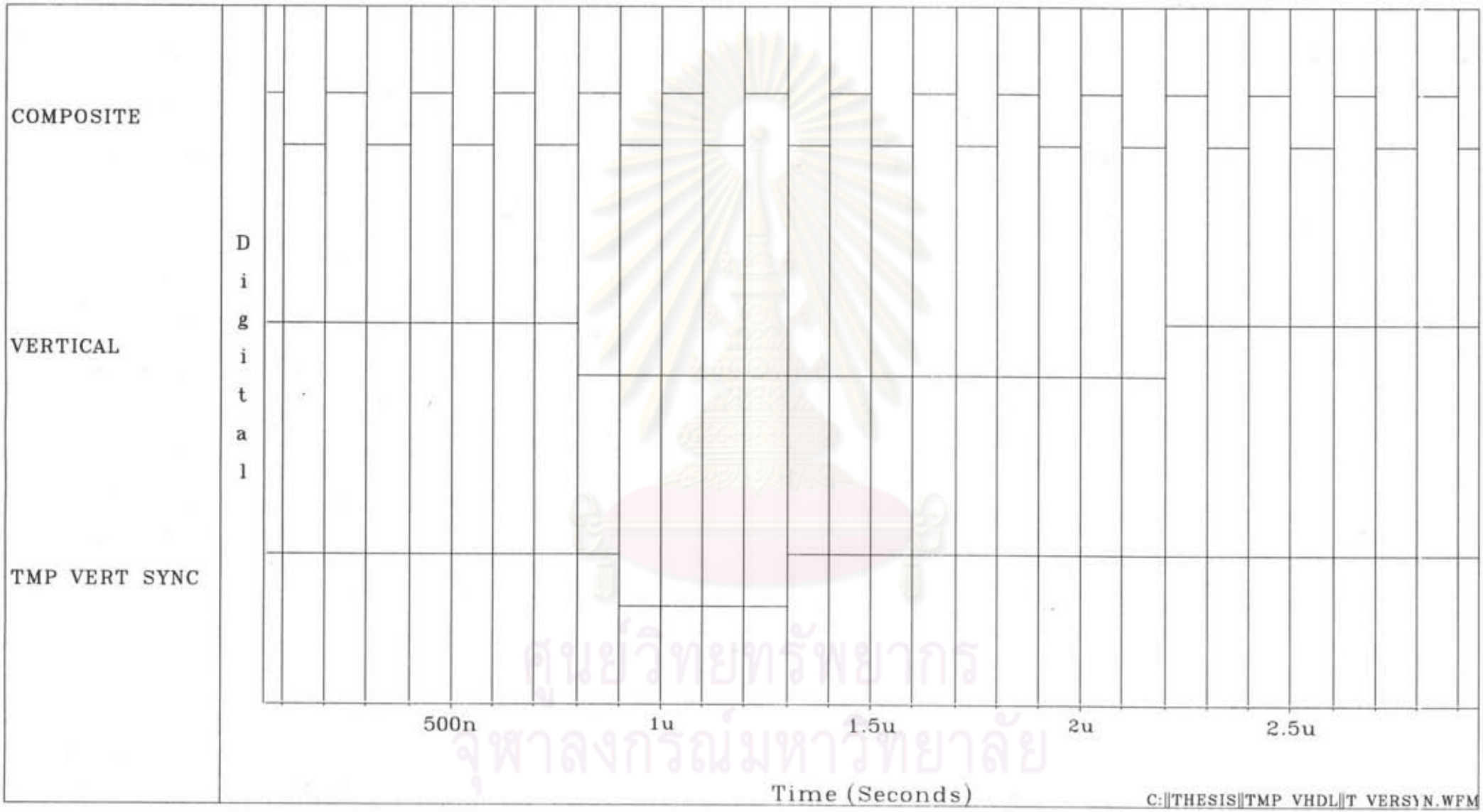
```
BEGIN
IF byte_num(5 downto 0) = B"000010" THEN
   bytetwo <= '1';
ELSE
   bytetwo <= '0';
END IF;
END PROCESS;

PROCESS
BEGIN
WAIT UNTIL (PFALLING(clk) OR (text_enable = '0'));
IF (text_enable = '0') THEN
   reg_data(7 downto 0) <= "00000000";
   tmp_reg              <= '0';
ELSE
   IF load_shift = '1' THEN
      IF bytezero = '1' THEN--"000001"
         reg_data(7) <= '0';
         reg_data(6) <= NOT(tmp_reg);
         reg_data(5) <= '0';
         reg_data(4) <= NOT(tmp_reg);
         reg_data(3) <= '0';
         reg_data(2) <= NOT(tmp_reg);
         reg_data(1) <= '0';
         reg_data(0) <= NOT(tmp_reg);
      ELSE
         IF byteone = '1' THEN--"000010"
            reg_data(7) <= '0';
            reg_data(6) <= NOT(tmp_reg);
            reg_data(5) <= '0';
            reg_data(4) <= NOT(tmp_reg);
            reg_data(3) <= '0';
            reg_data(2) <= NOT(tmp_reg);
            reg_data(1) <= '0';
            reg_data(0) <= NOT(tmp_reg);
         ELSE
            IF bytetwo = '1' THEN--"000011"
               reg_data(7) <= '0';
               reg_data(6) <= '0';
               reg_data(5) <= NOT(tmp_reg);
               reg_data(4) <= '0';
               reg_data(3) <= '0';
               reg_data(2) <= NOT(tmp_reg);
               reg_data(1) <= NOT(tmp_reg);
               reg_data(0) <= NOT(tmp_reg);
            ELSE
               reg_data(7 downto 0) <= data_in;
            END IF;
         END IF;
      END IF;
   ELSE  -- shift = '0'
      reg_data(7 downto 0) <= '0' & reg_data(7 downto 1);
   END IF;
END IF;
END PROCESS;
```

```
      PROCESS(empty,reg_data)
      BEGIN
      IF empty = '1' THEN
         serial_data_out <= '0';
      ELSE
         serial_data_out <= reg_data(0);
      END IF;
      END PROCESS;
END BLOCK SHIFT_DATA;

END behavioral;
```

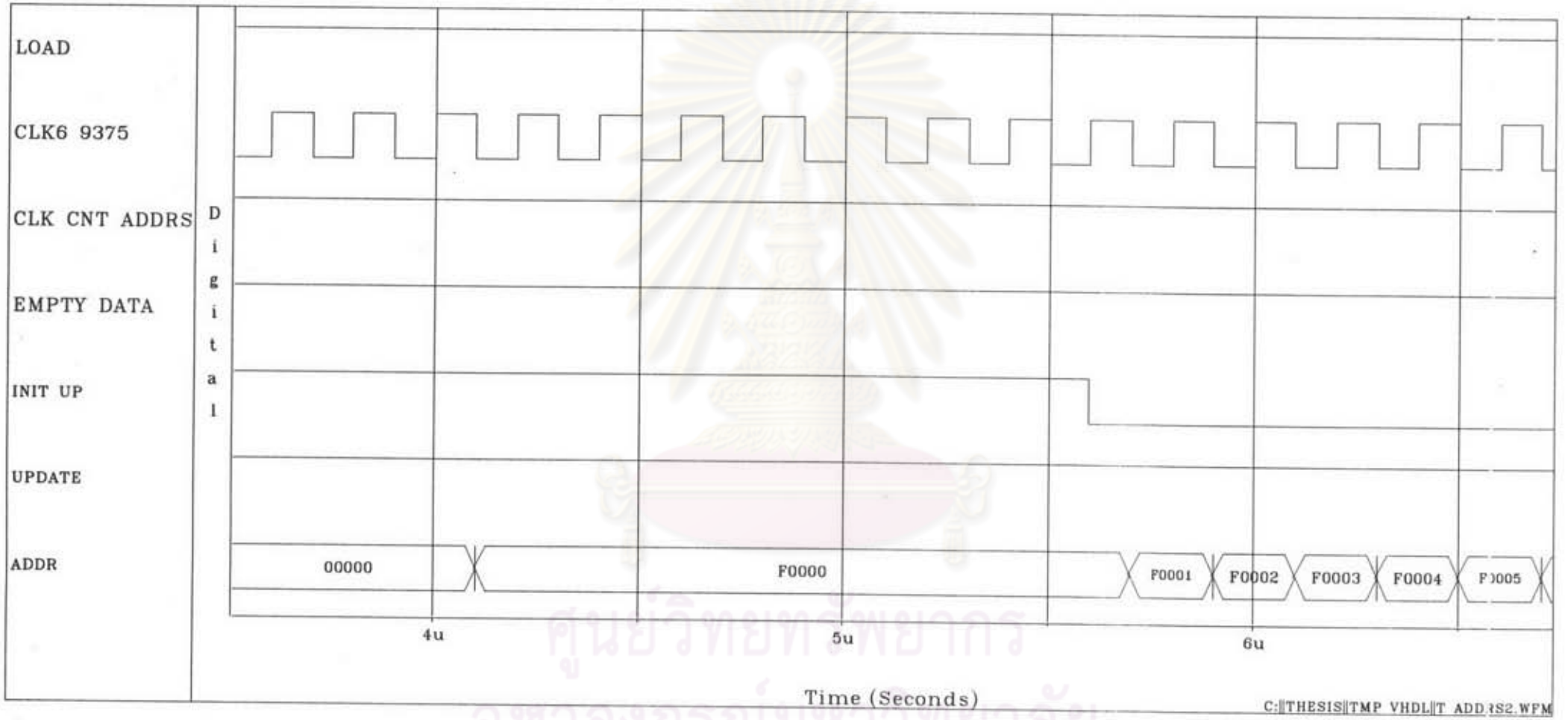
```
ENTITY t_latch IS
PORT  (
      ALE         : in  vlbit;
      addrs_in    : in  vlbit_1d(19 downto 0);
      ADDRS       : out vlbit_1d(19 downto 0)
       );
END t_latch;
ARCHITECTURE behavioral OF t_latch IS
SIGNAL      tmp_addrs  : vlbit_1d(7 downto 0);

BEGIN

      PROCESS
      BEGIN
      WAIT UNTIL PFALLING(ALE);
      tmp_addrs(7 downto 0) <= addrs_in(7 downto 0);
      END PROCESS;

      ADDRS( 7 downto 0) <= tmp_addrs(7 downto 0);
      ADDRS(19 downto 8) <= addrs_in(19 downto 8);

END behavioral;
```

ภาคผนวก ค

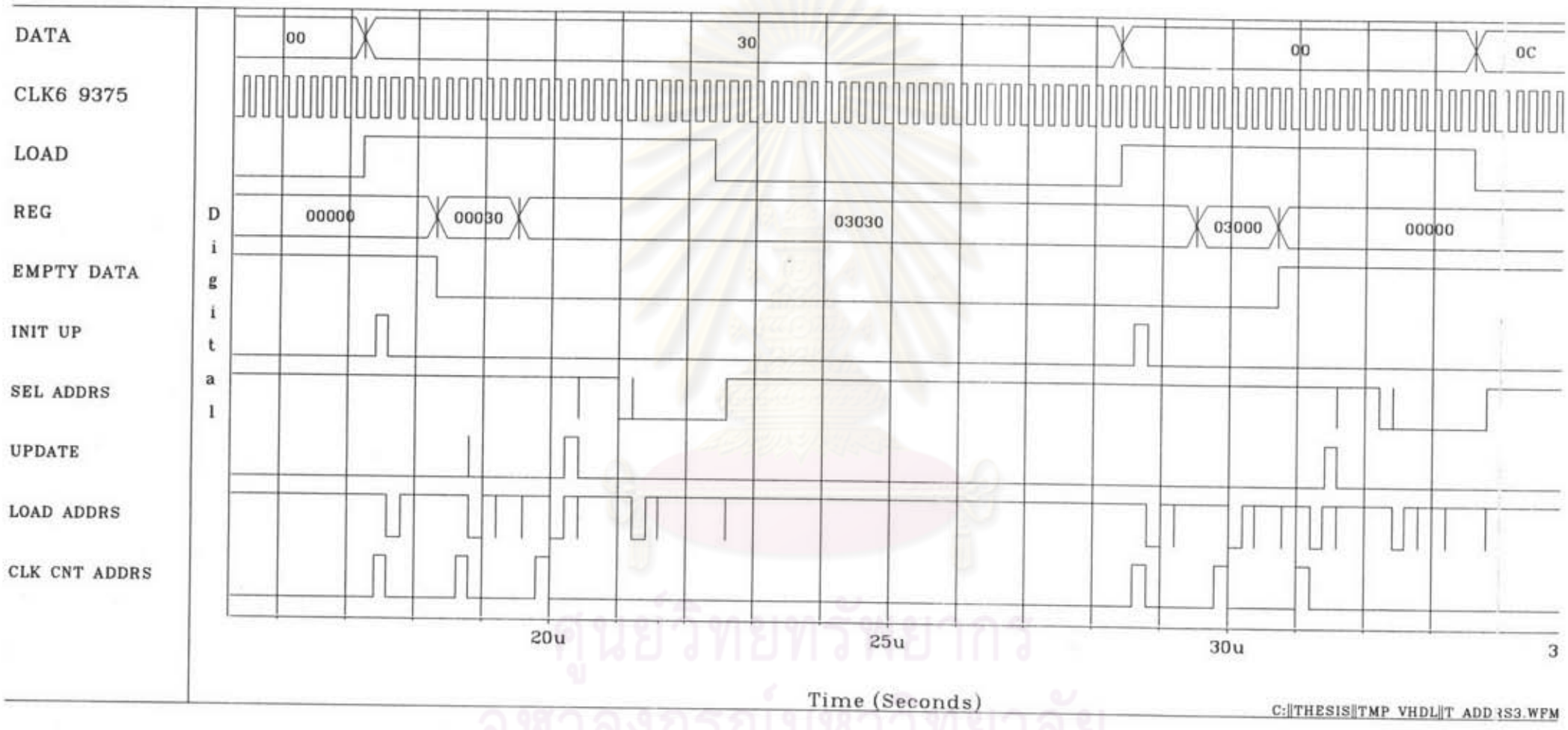
ตัวอย่างผลของการจำลองการทำงานและวงจรเชิงเลข ที่ออกแบบโดยใช้วีเอชดีแอล



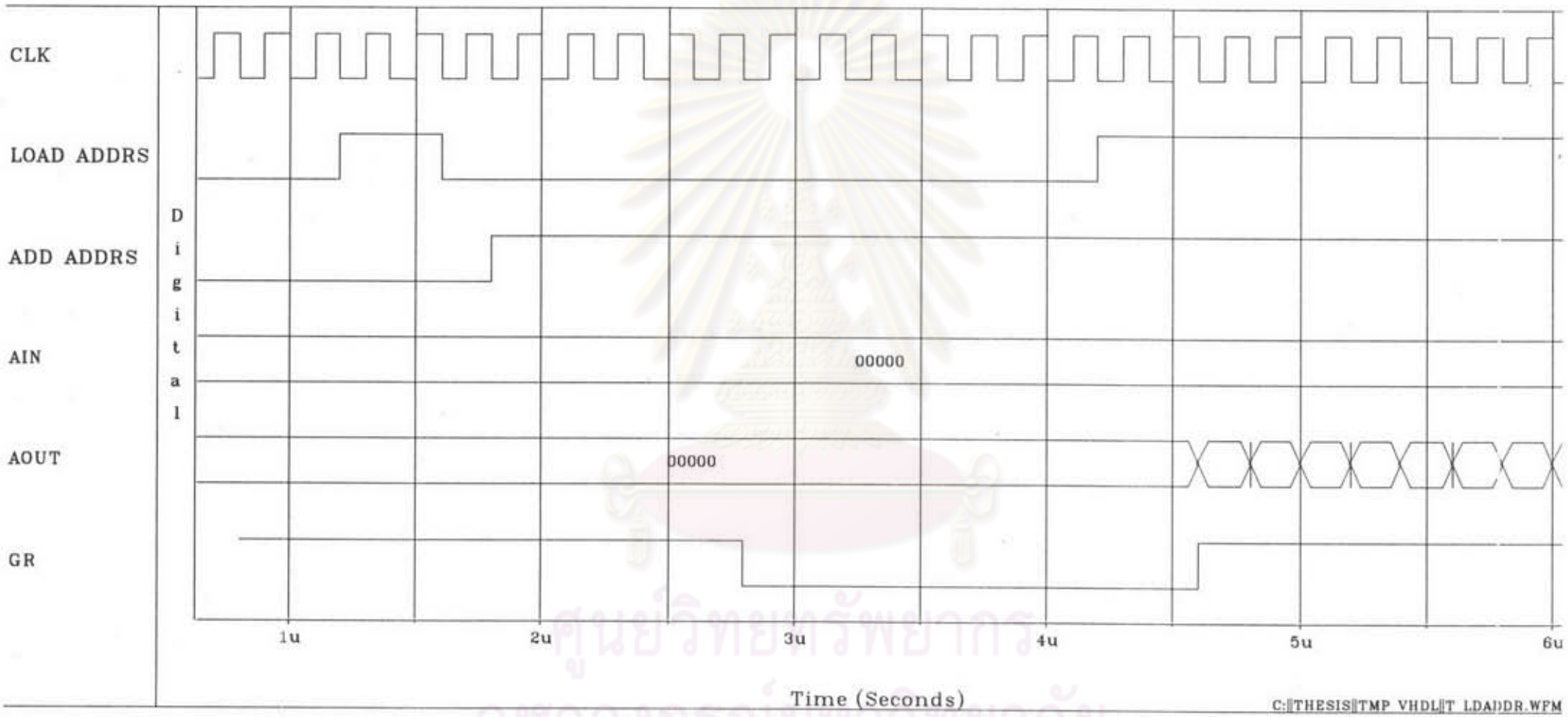
simulation file T_VERSYN.VHD



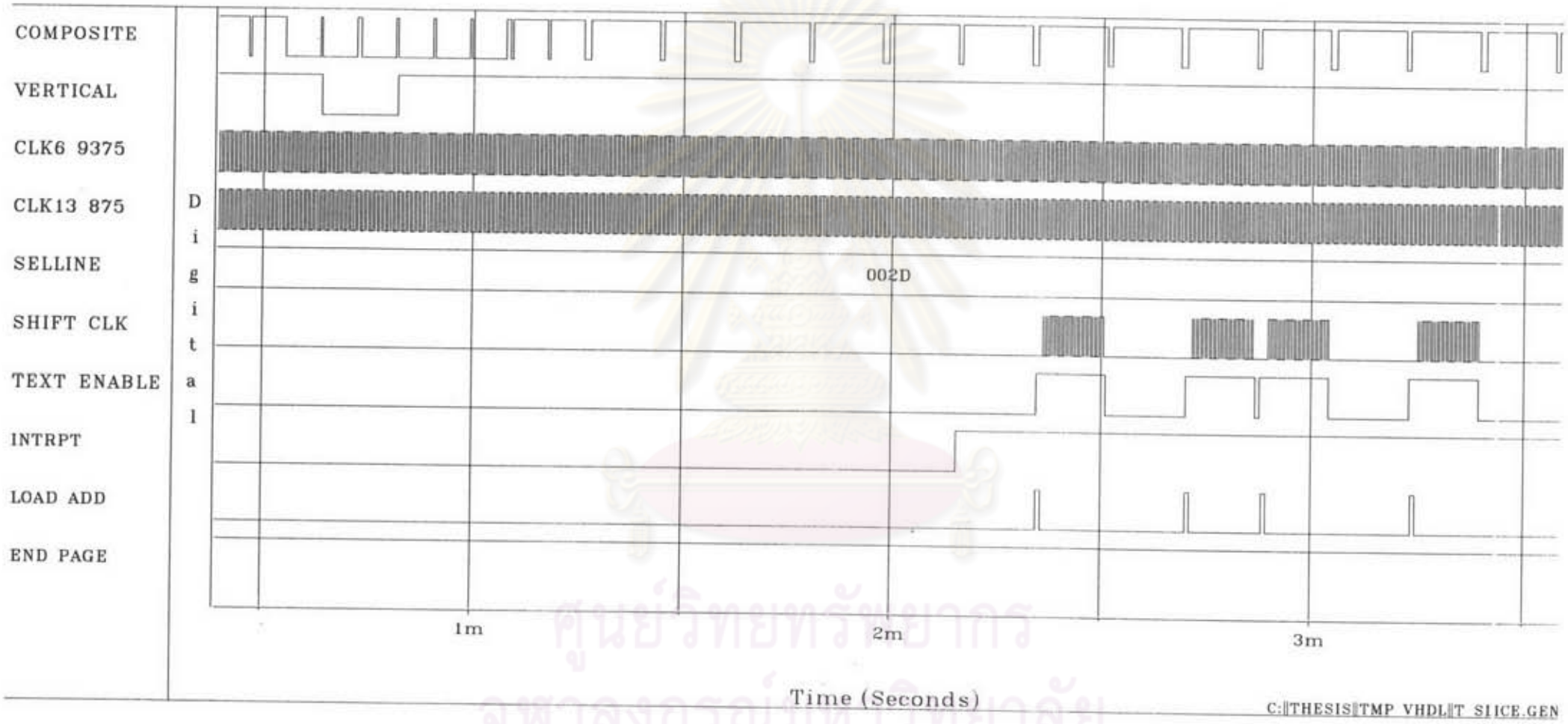
simulation file T_ADDRS2.VHD



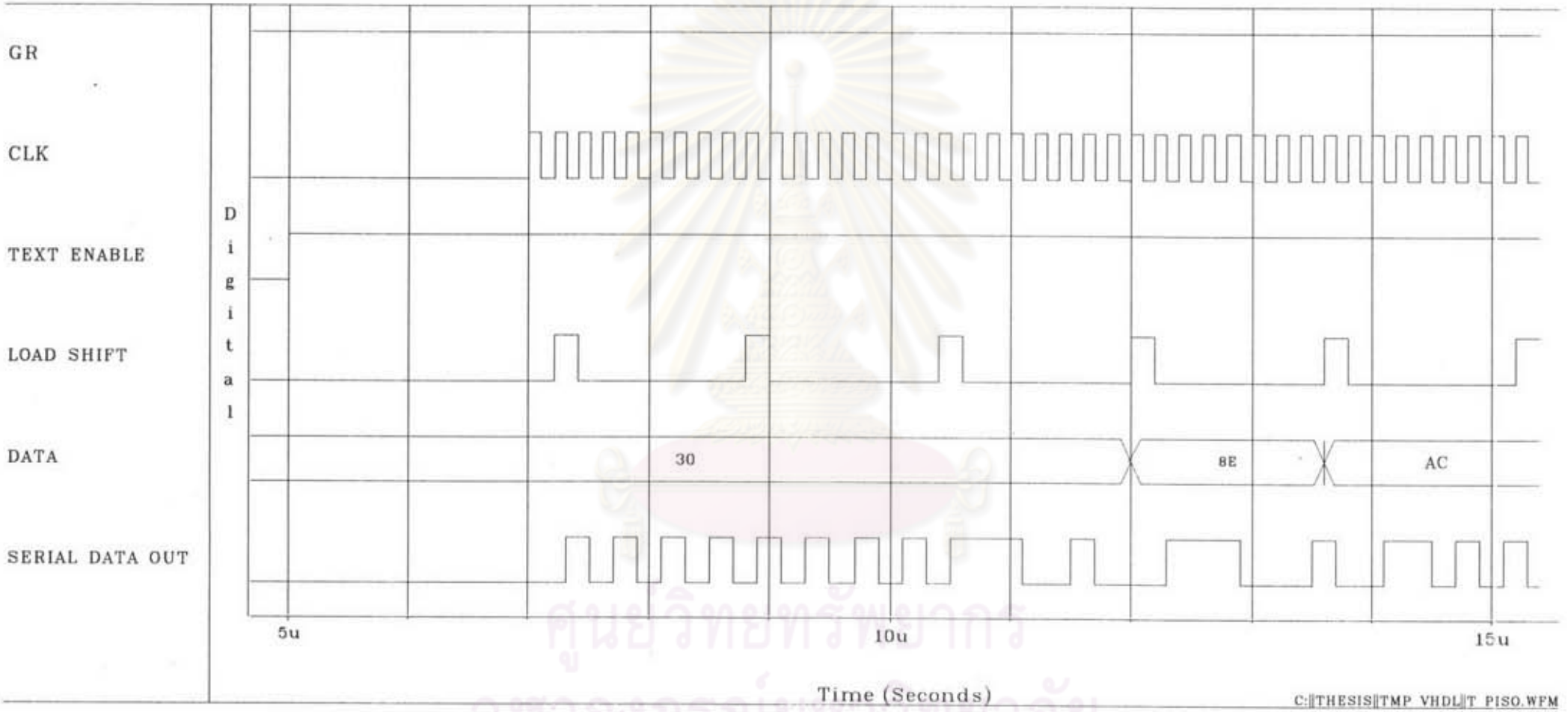
simulation file T_ADDRS3.VHD



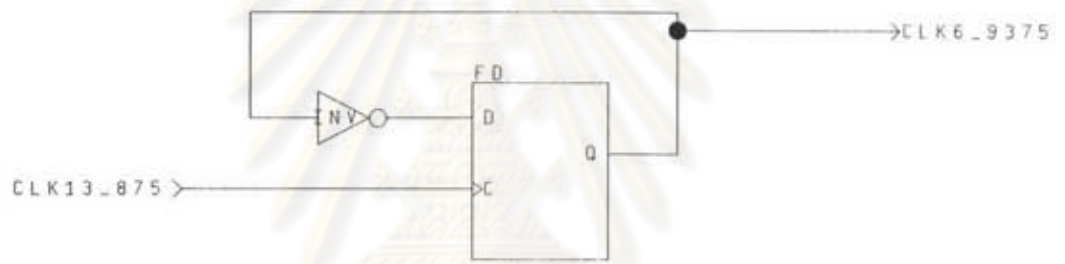
simulation file T_LDADDR.VHD



simulation file T_SLICE.VHD



simulation file T_PISO.VHD

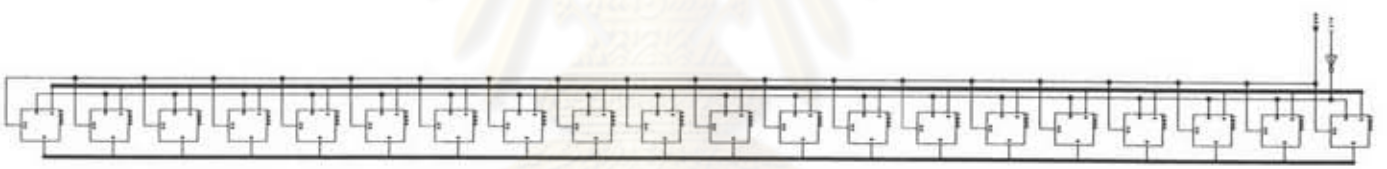


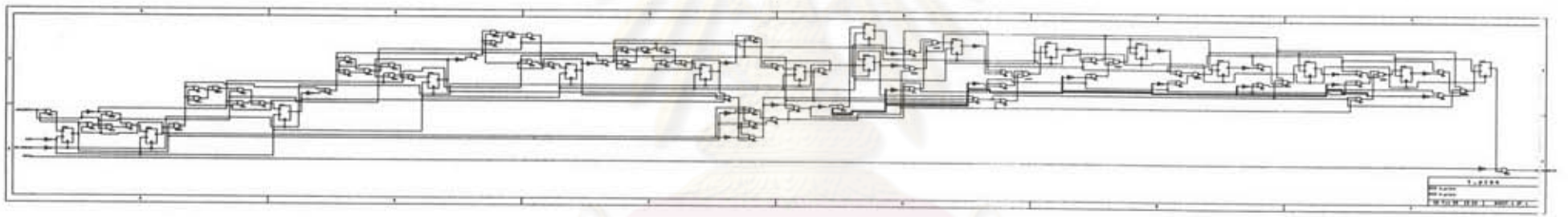
div2

WIR: div2

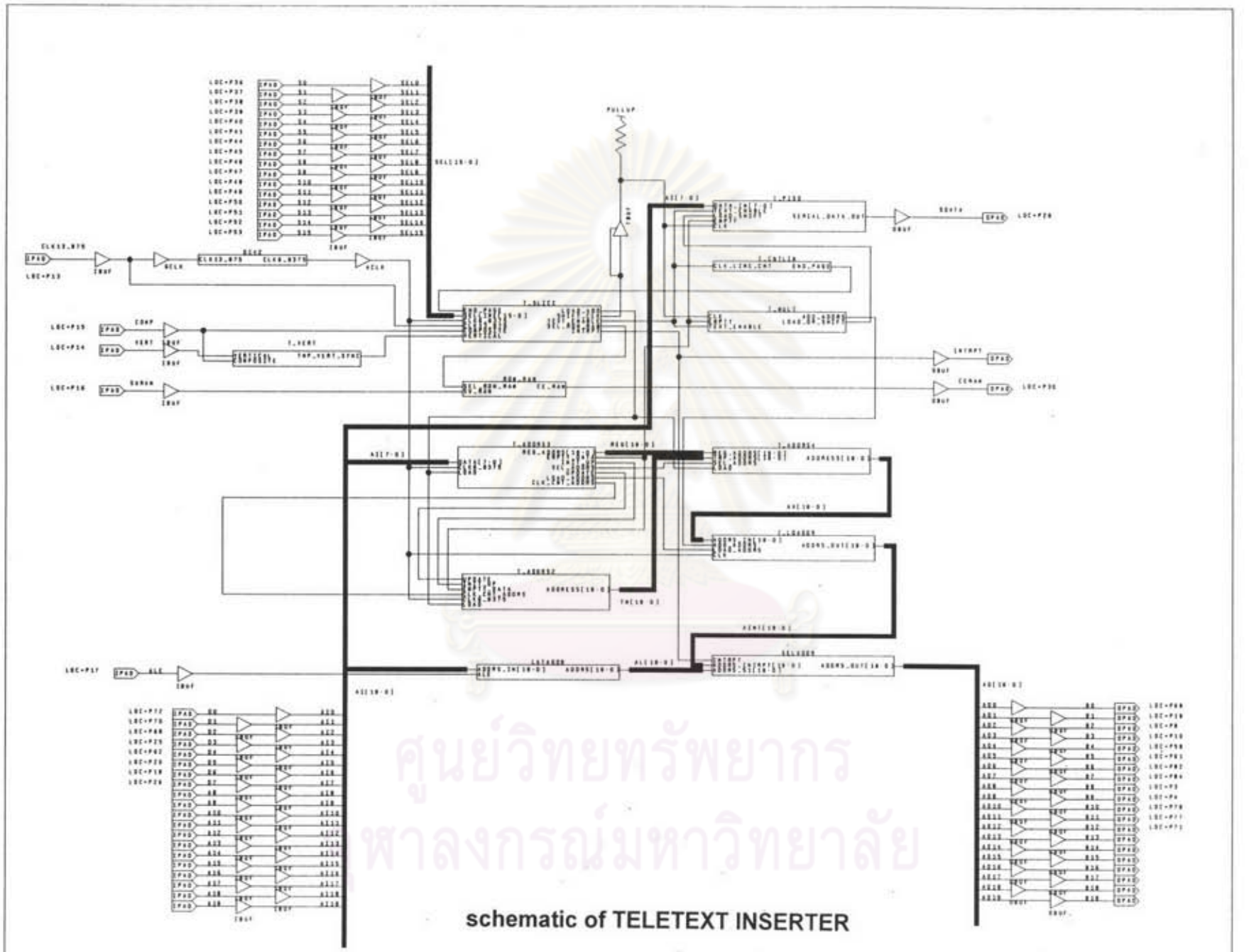
SCH: div2

8 Feb 96 9 49 SHEET 1 OF 1





schematic of file T_PISO.VHD



schematic of TELETEXT INSERTER

ภาคผนวก ง

รายละเอียดโปรแกรมสำหรับไมโครคอนโทรลเลอร์

```
MYBUF           EQU     R0
CNT_ADDRS       EQU     R1
MSB_ADDRS       EQU     2FH
BITVAR          EQU     2EH
A16             BIT     MSB_ADDRS.0
A17             BIT     MSB_ADDRS.1
A18             BIT     MSB_ADDRS.2
A19             BIT     MSB_ADDRS.3
ADDRS_NEED      BIT     BITVAR.0
DATA_NEED       BIT     BITVAR.1
RTS             BIT     P1.7
;
                ORG     0000H
                AJMP     INITIAL
;
                ORG     0003H   ; Interrupt Source IE0
                AJMP     INTRPT1
;
                ORG     000BH   ; Interrupt Source TF0
                AJMP     INTRPT2
;
                ORG     0023H
                AJMP     RX
;---------------------------------------------------------------
                ORG     0100H

INTRPT1:        MOV     TL0,#32H    ; Time Cal. from FFFF
                                    ; minus val.
                MOV     TH0,#0C2H
                SETB    TR0
                CLR     RTS
                RETI

INTRPT2:        SETB    P1.5
                CLR     TR0
                SETB    RTS
                RETI

RX:             CLR     EA
                JNB     RI,RX
                MOV     A,SBUF
                CLR     RI
                MOV     MYBUF,A
                CJNE    MYBUF,#0FFH,START
                MOV     MYBUF,#00H
START:          CJNE    MYBUF,#55H,DISABLE    ;DATA_FOUND
                SETB    ADDRS_NEED            ;MYBUF equal #55
H
                MOV     CNT_ADDRS,#00H
                JMP    RETURN_RX         ;In fact it is RETURN
INTRPT
DISABLE:        CJNE    MYBUF,#27H,DATA_FOUND
                CLR     DATA_NEED            ;MYBUF equal #27H
                JMP    RETURN_RX
DATA_FOUND:     JNB     ADDRS_NEED,DATA_TELE
```

```
                CJNE    CNT_ADDRS,#00H,ADDRS_1  ;
A,#00H,ADDRS_1
                MOV     DPL,MYBUF
                INC     CNT_ADDRS
                JMP     RETURN_RX
ADDRS_1:        CJNE    CNT_ADDRS,#01H,ADDRS_2  ;
A,#01H,ADDRS_2
                MOV     DPH,MYBUF
                INC     CNT_ADDRS
                JMP     RETURN_RX
ADDRS_2:        CJNE    CNT_ADDRS,#02H,DATA_TELE ; A,#02H,NEXT
;ERROR_ADDRS
                MOV     MSB_ADDRS,MYBUF
                INC     CNT_ADDRS
                CLR     ADDRS_NEED
                SETB    DATA_NEED
                JNB     A16,SETA16
                SETB    P1.0
NEXTA17:        JNB     A17,SETA17
                SETB    P1.1
NEXTA18:        JNB     A18,SETA18
                SETB    P1.2
NEXTA19:        JNB     A19,SETA19
                SETB    P1.3
NEXT:           JMP     RETURN_RX
SETA16:         CLR     P1.0
                JMP     NEXTA17
SETA17:         CLR     P1.1
                JMP     NEXTA18
SETA18:         CLR     P1.2
                JMP     NEXTA19
SETA19:         CLR     P1.3
                JMP     RETURN_RX

;******* MOVE DATA TO RAM ***********
DATA_TELE:      JNB     DATA_NEED,RETURN_RX
                MOV     A,MYBUF
                MOVX    @DPTR,A
                INC     DPTR

RETURN_RX:      SETB    EA
RETINT:         RETI

;----------------This part is main progam -------------------
INITIAL:        MOV     IP,#00000011B
                MOV     IE,#10010011B
                MOV     TMOD,#00100001B
                MOV     TCON,#00000001B
                MOV     SCON,#01010000B
                MOV     TH1,#0FDH
                MOV     CNT_ADDRS,#00H
                SETB    TR1

MAIN:           NOP
                NOP
```

```
                LJMP    MAIN
;***************************************************************
                END
```

ประวัติผู้เขียน

นายชาญณรงค์ อ่างทอง เกิดเมื่อวันที่ 14 มีนาคม พ.ศ. 2514 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมอิเล็กทรอนิกส์ จากคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2535 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิตสาขาวิศวกรรมไฟฟ้า (แขนงวิชาระบบเชิงเลข) ที่คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ซึ่งในระหว่างการศึกษาระดับมหาบัณฑิตนี้ได้รับทุนผู้ช่วยสอนจากภาควิชาวิศวกรรมไฟฟ้า