**เอกสารประกอบโครงงาน**

**วิชา Advance Digital System Design**

**เรื่อง Fingers DANCE**

**นำเสนอ**

**ผศ. เจริญ วงษ์ชุ่มเย็น**

**จัดทำโดย**

**นาย ชนาวีร์ จันแปงเงิน 54010275 3D/1**

**นาย ชัชชน เจริญชีพ 54010298 3D/1**

**นาย อดิศร คำหว่าง 54011458 3D/2**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**ปีการศึกษา 2556 ภาคเรียนที่ 1**

**ชื่อโครงงาน**

**Fingers DANCE**

**ที่มาและความสำคัญ**

จากการศึกษาและค้นคว้าวิชา Advance Digital System Design ทำให้ทราบว่านอกจากวงจรที่ออกแบบโดยการนำ Gate หรือ Flip Flop มาต่อกันแล้วยังสามารถพัฒนาวงจรด้วยภาษา VHDL ได้อีกด้วย ซึ่งการเขียนด้วยภาษา VHDL นั้นทำได้ง่ายกว่ามาก ทำให้ผู้ศึกษามีความสนใจในภาษา VHDL จึงได้คิดว่าสามารถทำอะไรด้วยภาษา VHDL ได้บ้าง จนได้ความคิดเสนอแนวคิดโครงงานเกมนี้เพื่อฝึกทักษะการเขียนโปรแกรมด้วยภาษา VHDL และการออกแบบวงจรดิจิตอลต่างๆได้

**วัตถุประสงค์**

1. เพื่อศึกษาการเขียนโปรแกรมด้วยภาษา VHDL
2. เพื่อศึกษาการออกแบบวงจรดิจิตอล
3. เพื่อพัฒนาทักษะในการทำงานเป็นทีม
4. เพื่อพัฒนาความรู้ความสามารถในการใช้ภาษา VHDL สร้างสรรค์ผลงานต่างๆ

**วิธีการดำเนินงาน**

วัสดุอุปกรณ์

1. คอมพิวเตอร์
2. FPGA
3. สาย JTAG
4. Software Program ISE WebPACK

แนวทางค้นคว้า

โครงงานนี้เป็นโครงงานเกี่ยวกับการสร้าง**เกม**ลงใน FPGA โดยการเล่นและแสดงผลนั้นจะใช้แค่ I/O ในส่วนบอร์ดของ FPGA เท่านั้น ซึ่งเพียงพอต่อการเล่นแล้ว

ในเกมจะใช้อุปกรณ์ใน FPGA ดังนี้

- ไฟ LED 8 ดวง 2 ชุด

- ปุ่มกดติดปล่อยดับ 3 ตัว

- ตัวแสดงผล 7-segment 4 ตัว

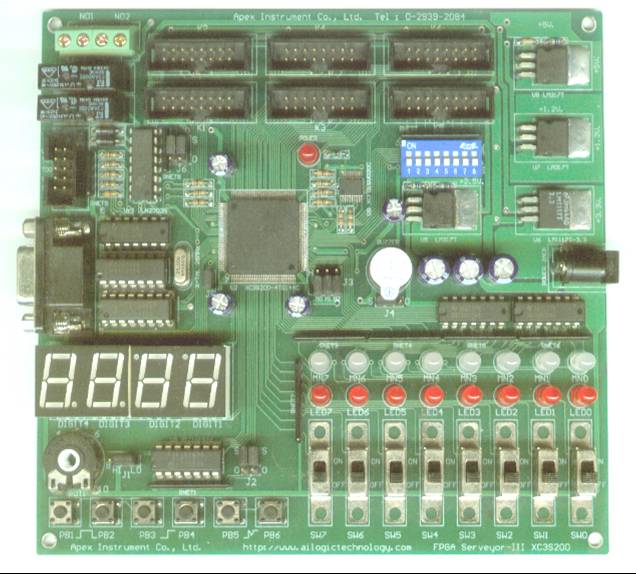
**ผลที่คาดหวัง**

1. ผู้ศึกษามีความรู้และความเข้าใจการเขียนโปรแกรมด้วยภาษา VHDL เพิ่มเติม

2. ผู้ศึกษามีความสามารถในการออกแบบวงจรดิจิตอลเพิ่มเติม

3. สามารถทำงานร่วมกันเป็นทีมได้อย่างดี

**ภาพแสดง interface ที่ติดต่อกับผู้ใช้งาน**

****

5

4

3

2

1

หมายเลข 1 – เป็น 7-segments display 2 หลักแสดงคะแนนของผู้เล่นโดยในแต่ละหลักสามารถแสดงคะแนนได้ถึง 0-f (เลขฐาน 16)

หมายเลข 2 – ทำการนับเวลาเริ่มเกมเมื่อผู้เล่นทำการกดปุ่ม start โดยจะนับเวลาตั้งแต่ 0-7 วินาทีหลังจากนั้นจะค้างไว้ที่ 7 วินาที(เวลาที่ไฟ LED ตัวขวาสุดวิ่งมายังดวงซ้ายสุดเป็นครั้งแรกนั่นเอง)

หมายเลข 3 – เป็นปุ่มกดของผู้เล่นจำนวน 2 ปุ่ม

หมายเลข 4 – เป็นปุ่ม start เพื่อทำการเริ่มเกมหรือกดอีกครั้งเพื่อ reset คะแนนก็ได้

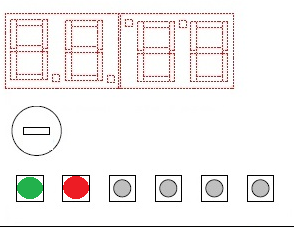
หมายเลข 5 – เป็นไฟวิ่งจำนวนสองแถวเป็นเงื่อนไขที่ผู้เล่นต้องทำตามจึงจะได้คะแนน

**วิธีการเล่น**

1. หลังจากกดปุ่ม start 7-segment หมายเลขสองละเริ่มนับ ไฟจะเริ่มเคลื่อนที่จากซ้ายมือไปขวามือ

****

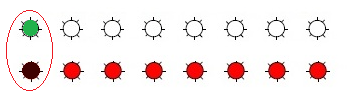
1. เมื่อไฟอยู่ในตำแหน่งนี้ ผู้เล่นสามารถป้อน input (ปุ่มหมายเลข 3) เป็นปุ่มกดติดปล่อยดับจำนวนสองปุ่ม คือปุ่มสีแดงและปุ่มสีเขียว

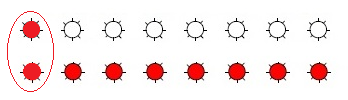
****

1. โดยมีเงื่อนไขในการกดขึ้นอยู่กับสีของไฟ LED ทั้งสองแถว กล่าวคือ ไฟแถวบนคือสีของปุ่ม LED ที่ต้องกด(มีสองสีคือสีเขียวและสีแดง) ไฟแถวล่างแสดงเงื่อนไขตรงกันข้ามกัน(มีสีเดียวคือสีแดง) เช่นไฟแถวบนเป็นสีเขียว และไฟแถวล่างติด ผู้เล่นจะต้องทำตรงกันข้ามกันกับเงื่อนไขของไฟในแถวแรก คือต้องกดสีแดง เป็นต้น

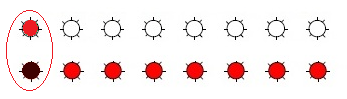
โดยทั่วไปมีอยู่ 4 กรณี

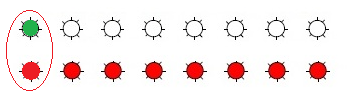
* **กรณีที่ต้องกดปุ่มสีเขียวคือ**

****

****

* **กรณีที่ต้องกดปุ่มสีแดงคือ**



****

1. เมื่อผู้เล่นทำตามเงื่อนไขถูกต้อง จะได้รับคะแนนเพิ่มโดยแสดงในช่อง 7-segment หมายเลข 1
2. ผู้เล่นสามารถเล่นไปได้เรื่อยๆ จนกว่าจะต้องการ
3. หากต้องการเริ่มเล่นใหม่และ reset คะแนนเดิมให้กดปุ่ม start อีกครั้ง

**การออกแบบวงจร**

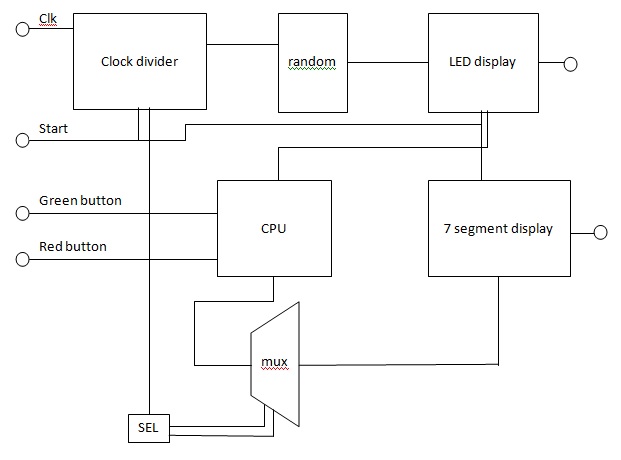
**Input**

1. Clk เป็นสัญญาณนาฬิกาที่จะนำไปใช้ใน module ต่างๆ
2. Start เป็นปุ่มกดเริ่มเกม รวมทั้งยัง reset ค่าต่างๆ ที่แสดง
3. Green button เป็นปุ่มที่ต้องกดเมื่ออยู่ในเงื่อนไขที่ต้องกดปุ่มสีเขียว
4. Red button เป็นปุ่มที่ต้องกดเมื่ออยู่ในเงื่อนไขที่ต้องกดปุ่มสีแดง

**Output**

1. LED display แสดงผลไฟวิ่งจำนวน 8 ดวงสองแถวมีสองสถานะ

2. 7 segment display แสดงคะแนนของผู้เล่นและเวลานับก่อนที่จะสามารถกดปุ่มได้

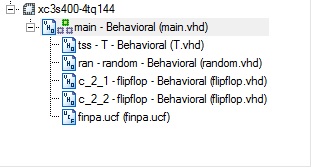
****

**แผนภาพแสดง Module ต่างๆ**

module ต่างๆ ที่จำเป็นมีดังนี้

1. Clock divider ทำการหารสัญญาณนาฬิกาให้เป็นค่าต่างๆ ที่ต้องการแล้วส่งไปให้วงจรอื่นๆ ไปใช้
2. Random ทำการสุ่มค่าแล้วแสดงผลไปยัง LED display
3. LED display ทำการแสดงผลไปยังผู้เล่น แล้วนำสถานะปัจจุบันไปแสดงผลในส่วนประมวลผล (CPU) ร่วมกับ input ที่ผู้เล่นป้อนเข้ามา
4. CPU ทำการประมวลผล input ที่ได้รับมาจาก user เปรียบเทียบกับสถานะของไฟ LED แล้วทำการให้คะแนน
5. MUX ทำการนำค่าที่ได้จาก CPU ส่งไปแสดงผลใน 7-segments
6. 7 segment display แสดงผลคะแนนของ user

VHDL Code ที่ใช้ในโครงงานนี้



* Main.vhd ทำการรวบรวม module ย่อยทั้งหมดเข้าไว้ด้วยกัน รวมทั้งยังจัดการ เตรียม input ที่จำเป็นและการแสดงผลออกทาง 7-segmants อีกด้วย
* T.vhd เป็น T flip flop
* Random.vhd เป็น module ที่ทำการสุ่มและให้ผลลัพธ์ออกมาเป็นเลขขนาด 2 bits
* C\_2\_1 และ c\_2\_1 เป็น T flip flop แบบมีขาเคลียร์ใช้สำหรับ generate ค่าเพื่อนำไปเข้า MUX สำหรับการแสดงผลต่างๆ

############################################### MAIN.vhd ###########################################

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity main is

Port ( inputa : in STD\_LOGIC;

inputb : in STD\_LOGIC;

start : in std\_logic ;

clkin : in std\_logic ;

clkk : in std\_logic ;

ledup : out std\_logic\_vector (7 downto 0) ;

leddown : out std\_logic\_vector ( 7 downto 0 ); -----

common : out std\_logic\_vector (3 downto 0) ;

seg : out std\_logic\_vector (6 downto 0 ) );

end main;

architecture Behavioral of main is

component random

port( input:in std\_logic;

clk:in std\_logic;

output:out std\_logic\_vector(1 downto 0) );

end component;

component T

port ( T : in std\_logic ;

clk: in std\_logic ;

Q : out std\_logic );

end component ;

component flipflop

port ( clk,t : in STD\_LOGIC ;

q : out std\_logic );

end component;

signal t\_inputa,t\_inputb,inputall : std\_logic := '0';

signal checks : std\_logic := '0';

signal checkstart : std\_logic\_vector (2 downto 0) := "000";

signal COUNT,COUNT2,countsc : integer := 0 ;

signal C : integer := 5;

signal countstop : integer := 0;

signal clk1,clk2,clksc,clk05 : std\_logic := '0';

signal t\_ledup : std\_logic\_vector (7 downto 0) := "00000000";

signal t\_leddown : std\_logic\_vector (7 downto 0) := "00000000";

signal SS : std\_logic := '0';

กำหนด signal พื้นฐานที่จำเป็นต้องใช้

signal goran : std\_logic := '0';

signal outt : std\_logic := '0';

signal stop : std\_logic := '0';

signal ranout: std\_logic\_vector (1 downto 0) := "00" ;

signal keepclk : std\_logic := '0';

signal hp : std\_logic\_vector (3 downto 0) := "0101";

signal point,pointa,pointb : std\_logic\_vector (7 downto 0) := "00000000";

signal com : std\_logic\_vector (3 downto 0) := "0000";

signal mux,muxx :std\_logic\_vector (3 downto 0 ) := "0000";

signal c2b : std\_logic\_vector (1 downto 0) := "00" ;

signal but,buta,butb : std\_logic := '0';

signal preinputa,preinputb:std\_logic:='0';

begin

------------------------------------- create clk 1 hz ----------------------------------

process(clkk)

begin

if ( clkk'event and clkk = '1' ) then

if (count >= 12500000) then

count <= 0 ;

clk1 <= not clk1 ;

else

count <= count + 1 ;

end if ;

end if ;

end process;

process(clkk)

begin

if ( clkk'event and clkk = '1' ) then

if (count >= 12500000) then

สร้างสัญญาณ clock ความถี่ขนาด 1 hz

count <= 0 ;

clk1 <= not clk1 ;

else

count <= count + 1 ;

end if ;

end if ;

end process;

----------------------------------- create clk scan --------------------------------------

process(clkk)

สร้างสัญญาณ clock ที่มีความเร็วพอเหมาะที่จะนำไปสแกนผลออกทาง 7 segments

process(clkk)

begin

if ( clkk'event and clkk = '1' ) then

if (countsc >= 30000) then

countsc <= 0 ;

clksc <= not clksc ;

else

countsc <= countsc + 1 ;

end if ;

end if ;

end process;

begin

if ( clkk'event and clkk = '1' ) then

if (countsc >= 30000) then

countsc <= 0 ;

clksc <= not clksc ;

else

countsc <= countsc + 1 ;

end if ;

end if ;

end process;

--------------------------- --------------- led run --------------------------------------------

process(clk1)

begin

if(clk1'event and clk1 = '1' ) then

if (outt = '1' )then

ในส่วนนี้ทำให้ไฟ LED วิ่งไปเรื่อยๆ จนกว่าจะหยุด

if (countstop >= 10) then

countstop <= 0 ;

stop <= '1' ;

else

countstop <= countstop + 1 ;

end if;

end if;

stop <= '0';

end if;

end process;

SS <= start or stop ;

ทำการ port map แล้วทำการสุ่มเทียมจากนั้นนำค่าที่ได้ออกมาจากการวงจรสุ่มใส่กลับเข้าไปในวงจรสุ่มอีกครั้งเพื่อให้ได้ผลลัพธ์ที่หลายหลายและไม่ซ้ำ

tss : T port map (T => '1' ,clk => SS ,Q => outt );

goran <= outt and clk2;

ran : random port map (input => goran ,clk =>clkk , output => ranout );

ran : random port map (input => goran ,clk =>clkk , output => ranout );

process (ranout)

begin

if ranout = "00" then

c<=(c+1\*(c+7))+333;

elsif ranout = "01" then

c<=(c+2) \* (c+3) ;

elsif ranout = "10" then

c<=c+3+c;

elsif ranout = "11" then

c<=(c\*c)-c;

end if ;

if(c>=400000) then

c<=5;

end if;

end process;

ledup(7) <= '0' when ranout = "00" else

'0' when ranout = "01" else

'1' when ranout = "10" else

'1' when ranout = "11" ;

leddown(7)<= '0' when ranout = "00" else

'0' when ranout = "10" else

'1' when ranout = "01" else

'1' when ranout = "11" ;

ledup(7) <= '0' when ranout = "00" else

'0' when ranout = "01" else

ในส่วนนี้เป็นการนำผลลัพธ์ที่ได้จากการสุ่มไปกำหนดค่าไฟ LED ตัวขวามือสุดของทั้งแถวบนและแถวล่าง

'1' when ranout = "10" else

'1' when ranout = "11" ;

leddown(7)<= '0' when ranout = "00" else

'0' when ranout = "10" else

'1' when ranout = "01" else

'1' when ranout = "11" ;

process(clk1)

begin

if(clk1'event and clk1 = '1' ) then

t\_ledup(0) <= t\_ledup(1);

t\_leddown(0) <= t\_leddown(1);

t\_ledup(1) <= t\_ledup(2);

t\_leddown(1) <= t\_leddown(2);

t\_ledup(2) <= t\_ledup(3);

t\_leddown(2) <= t\_leddown(3);

t\_ledup(3) <= t\_ledup(4);

t\_leddown(3) <= t\_leddown(4);

t\_ledup(4) <= t\_ledup(5);

t\_leddown(4) <= t\_leddown(5);

t\_ledup(5) <= t\_ledup(6);

t\_leddown(5) <= t\_leddown(6);

t\_ledup(6) <= t\_ledup(7);

t\_leddown(6) <= t\_leddown(7);

end if;

เป็นส่วนที่ทำให้ไฟ LED ทั้งแถวบนและแถวล่างวิ่งจากขวามือไปยังซ้ายมือโดยขึ้นอยู่กับความเร็ว clock ที่ป้อนเข้ามา

if(clk1'event and clk1 = '1' ) then

t\_ledup(0) <= t\_ledup(1);

t\_leddown(0) <= t\_leddown(1);

t\_ledup(1) <= t\_ledup(2);

t\_leddown(1) <= t\_leddown(2);

t\_ledup(2) <= t\_ledup(3);

t\_leddown(2) <= t\_leddown(3);

t\_ledup(3) <= t\_ledup(4);

t\_leddown(3) <= t\_leddown(4);

t\_ledup(4) <= t\_ledup(5);

t\_leddown(4) <= t\_leddown(5);

t\_ledup(5) <= t\_ledup(6);

t\_leddown(5) <= t\_leddown(6);

t\_ledup(6) <= t\_ledup(7);

t\_leddown(6) <= t\_leddown(7);

end if;

end process ;

ledup <= t\_ledup;

leddown <= t\_leddown;

process(clk1)

begin

if start = '0' then

if clk1'event and clk1 = '1' then

if checkstart >= 7 then

checks <= '1' ;

else

checkstart <= checkstart + 1 ;

end if;

end if;

elsif start = '1' then

checks <= '0' ;

checkstart <= "000";

end if;

end process;

process(clk1)

begin

ทำการนับเวลาเป็นเวลา 7 วินาทีนับตั้งแต่เริ่มกดปุ่ม start หากนับเวลาครบ 7 วินาทีแล้วจึงจะเริ่มป้อน input ได้และมีการคิดคะแนนเกิดขึ้น

if start = '0' then

if clk1'event and clk1 = '1' then

if checkstart >= 7 then

checks <= '1' ;

else

checkstart <= checkstart + 1 ;

end if;

end if;

elsif start = '1' then

checks <= '0' ;

checkstart <= "000";

end if;

end process;

------------------------------------------------ CPU -------------------------------------------------

t\_inputa <= inputa and outt and checks ;

t\_inputb <= inputb and outt and checks ;

ในส่วนนี้เป็นส่วนประมวลผล input ที่ป้อนเข้ามาเทียบกับสถานะของไฟ LED หากตรงกันก็จะทำการเพิ่มคะแนนโดยมีคะแนนสองส่วนด้วยกันคือคะแนนที่มาจากการกดปุ่ม a และ b เมื่อจบ process ก็จะเอาคะแนนจากทั้งสองส่วนมาบวกเข้าด้วยกันแล้วนำไปแสดงผล

process(t\_inputa,t\_inputb,t\_leddown,t\_ledup,point)

begin

if (t\_inputa'event and t\_inputa = '1' ) then

if (t\_ledup(0) = '0' and t\_leddown(0) = '0') then

pointa <= pointa + 1 ;

elsif (t\_ledup(0) = '0' and t\_leddown(0) = '1') then

pointa <= pointa ;

elsif (t\_ledup(0) = '1' and t\_leddown(0) = '0') then

pointa <= pointa ;

elsif (t\_ledup(0) = '1' and t\_leddown(0) = '1') then

pointa <= pointa + 1 ;

end if;

end if;

if (t\_inputa'event and t\_inputa = '1' ) then

if (t\_ledup(0) = '0' and t\_leddown(0) = '0') then

pointa <= pointa + 1 ;

elsif (t\_ledup(0) = '0' and t\_leddown(0) = '1') then

pointa <= pointa ;

elsif (t\_ledup(0) = '1' and t\_leddown(0) = '0') then

pointa <= pointa ;

elsif (t\_ledup(0) = '1' and t\_leddown(0) = '1') then

pointa <= pointa + 1 ;

end if;

end if;

if start = '1' then

pointa <= "00000000";

end if;

end process;

if start = '1' thencoo

ถ้ามีการกดปุ่ม start คะแนนจะถูก reset ทันที

pointa <= "00000000";

end if;

end process;

process(t\_inputa,t\_inputb,t\_leddown,t\_ledup,point)

begin

if (t\_inputb'event and t\_inputb = '1' ) then

if (t\_ledup(0) = '0' and t\_leddown(0) = '0') then

pointb <= pointb ;

elsif (t\_ledup(0) = '0' and t\_leddown(0) = '1') then

pointb <= pointb + 1 ;

elsif (t\_ledup(0) = '1' and t\_leddown(0) = '0') then

pointb <= pointb + 1 ;

elsif (t\_ledup(0) = '1' and t\_leddown(0) = '1') then

pointb <= pointb ;

end if;

end if;

if start = '1' then

pointb <= "00000000";

ทำการ port map flip flop ที่สร้างไว้ในตอนแรกโดยใส่ clock เข้าไป โดยผลลัพธ์ที่ได้นำไปเข้า multiplexor และใช้สแกนขา common เพื่อแสดงผลลัพธ์ออกทาง 7 segnets

end if;

end process;

point <= pointa + pointb ;

------------------------------------------------- 7 - segment ------------------------------------

c\_2\_1 : flipflop port map (clk => clksc , T => '1' , Q => c2b(0) );

c\_2\_2 : flipflop port map (clk => clksc , T => c2b(0) , Q => c2b(1) );

c\_2\_1 : flipflop port map (clk => clksc , T => '1' , Q => c2b(0) );

c\_2\_2 : flipflop port map (clk => clksc , T => c2b(0) , Q => c2b(1) );

muxx <= '0' & checkstart;

------------------------------------------------- MUX & common -----------------------------------

mux <= point( 3 downto 0) when c2b = "00" else

point (7 downto 4) when c2b = "01" else

muxx when c2b = "10" ;

hp when c2b = "11" ;

common <= "1110" when c2b = "00" else

"1101" when c2b = "01" else

"1011" when c2b = "10" else

"0111" when c2b = "11" ;

seg <= "1110001" when mux="1111" else --F

"1111001" when mux="1110" else --E

"1011110" when mux="1101" else --d

"0111001" when mux="1100" else --C

"1111100" when mux="1011" else --b

"1110111" when mux="1010" else --A

"1101111" when mux="1001" else --9

"1111111" when mux="1000" else --8

"0000111" when mux="0111" else --7

"1111101" when mux="0110" else --6

"1101101" when mux="0101" else --5

"1100110" when mux="0100" else --4

"1001111" when mux="0011" else --3

"1011011" when mux="0010" else --2

"0000110" when mux="0001" else --1

"0111111";

seg <= "1110001" when mux="1111" else --F

ส่วนแสดงผลไปยัง 7 segment display

"1111001" when mux="1110" else --E

"1011110" when mux="1101" else --d

"0111001" when mux="1100" else --C

"1111100" when mux="1011" else --b

"1110111" when mux="1010" else --A

"1101111" when mux="1001" else --9

end Behavioral;

############################################## random.vhd ###########################################

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity random is

port( input:in std\_logic;

clk:in std\_logic;

output:out std\_logic\_vector(1 downto 0)

);

end random;

architecture Behavioral of random is

signal x:std\_logic\_vector(1 downto 0):="00";

signal count,count\_t:integer := 0 ;

begin

process(clk)

begin

if(clk'event and clk='1') then

count<=count+1;

end if;

end process;

process(input)

begin

if(input'event and input='1') then

count\_t<=count mod 4;

end if;

end process;

x<=conv\_std\_logic\_vector(count\_t,2);

output<=x;

end Behavioral;

##################################### C-2\_1.vhd, C\_2\_2.vhd ###########################################

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity flipflop is

Port ( clk,t,clear : in STD\_LOGIC ;

q : out std\_logic );

end flipflop;

architecture Behavioral of flipflop is

signal tempq : std\_logic := '0' ;

begin

process(clk,t,clear)

begin

if clear = '1' then

tempq <= '0';

elsif clear = '0' then

if clk'event and clk = '1' then

if t = '1' then

tempq <= not tempq ;

end if;

end if;

end if;

end process;

q <= tempq;

end Behavioral;

################################################ T.vhd #############################################

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity T is

Port ( T : in STD\_LOGIC;

clk : in std\_logic;

Q : out STD\_LOGIC);

end T;

architecture Behavioral of T is

signal q\_temp : std\_logic := '0';

begin

process (clk)

begin

if (clk'event and clk = '1') then

if (T = '1') then

q\_temp <= not q\_temp ;

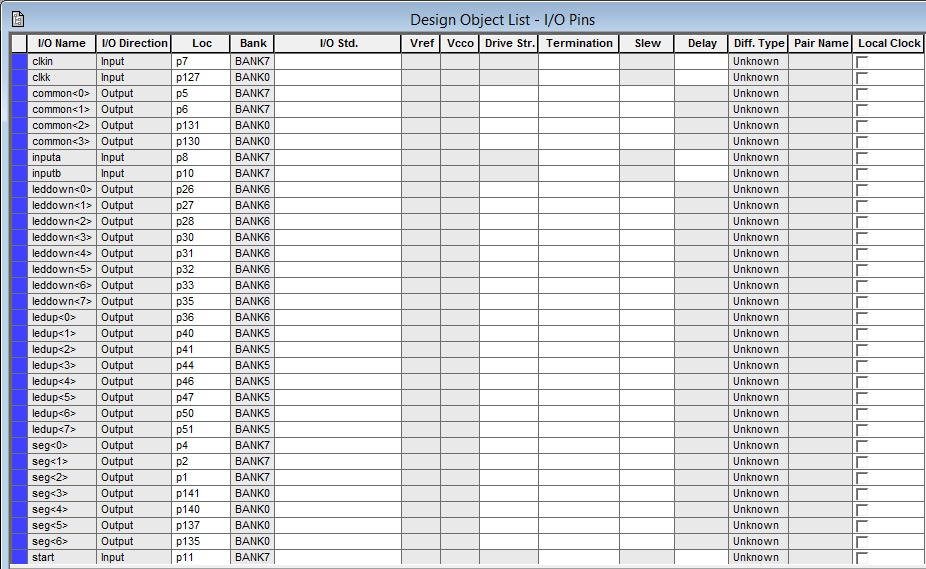
end if ;

end if;

end process ;

Q <= q\_temp ;

end Behavioral;



ภาพแสดงการ mapped I/O pins บน FPGA

**เอกสารอ้างอิง**

ออกแบบไอซีดิจิตอลด้วย FPGA และ CPLD ภาคปฏิบัติ โดยใช้ภาษา VHDL (ISE WebPACK)