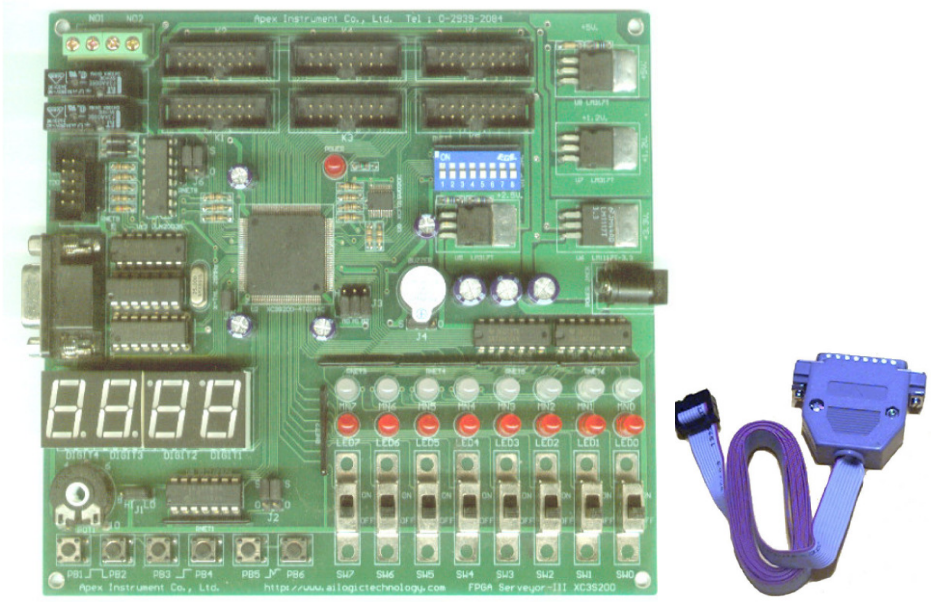
Lab I: Introduction to FPGA design using Verilog

1. แนะนำการใช้งาน FPGA และภาษา Verilog

1.1 แนะนำบอร์ด FPGA ที่ใช้ในการทดลอง



รูปที่ 1 บอร์ดทดลอง FPGA (XC3S200)

บอร์ดทดลองที่ใช้ในการทดลองเป็นบอร์ดรุ่น Surveyor-III ของบริษัท APEX Instrument ที่ใช้ชิป FPGA ของบริษัท Xilinx รุ่น XC3S200 ดังรูปที่ 1 โดยบอร์ดนี้มีความจุของวงจรมากถึง 200,000 เกตและใช้ platform flash PROM สำหรับเก็บข้อมูลของวงจร ซึ่งสามารถโปรแกรมวงจรลงสู่ flash PROM นี้ผ่านทางสายดาวน์โหลดแบบ JTAG โดยบอร์ดที่ใช้งานมีคุณลักษณะทั่วไปดังนี้

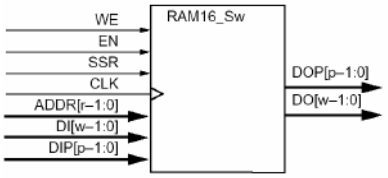
- FPGA รุ่น XC3S200-4TQ144C (*package แบบ surface mount ทั้ง 4 ด้าน จำนวนขาทั้งสิ้น 144 ขา*)
- Platform flash PROM เบอร์ XCF01SVO20C
- 7 Segment 4 หลัก
- DIP switch 8 ดวง
- LED 3 สถานะ 8 ดวง
- Buzzer 1 ตัว
- clock generator 1-100 Hz และ 1-10 KHz
- Relay 220V/3A 2 ตัว
- 25 MHz Oscillator

คุณลักษณะเด่นของ FPGA รุ่นนี้คือ

- มีความจุวงจรถึง 200 เกต
- หน่วยความจำ 18Kb clock RAMs จำนวน 12 ชุด รวม 216K bits
- วงจรคูณ 18 x 18 ผลลัพธ์ที่ได้เป็น 36 บิต มีจำนวน 12 ชุด
- Digital Clock Manager (DCM) จำนวน 4 ชุด
- Digital Controlled Impedance (DCI)

1.1.1 หน่วยความจำ 18 Kb block RAM

เป็นหน่วยความจำความเร็วสูงประมาณ 200 MHz มีด้วยกัน 12 ชุดโดยแต่ละชุดทำงานได้ตามรูปที่2 และ 3



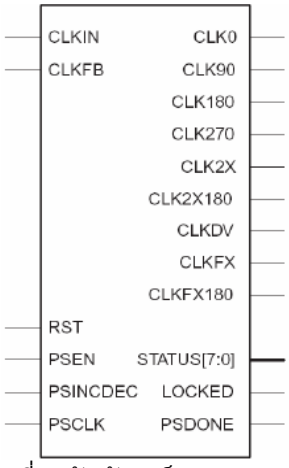
รูปที่ 2 แสดงขนาดของ RAM แบบ Single Port

Organization	Memory Depth	Data Width	Parity Width
512x36	512	32	4
1Kx18	1024	16	2
2Kx9	2048	8	1
4Kx4	4096	4	-
8Kx2	8192	2	-
16Kx1	16384	1	-

รูปที่ 3 RAM แบบ Single Port ขนาดต่างๆที่สร้างจาก Block RAM แต่ละชุด

1.1.2 Digital Clock Manager

Digital Clock Manager (DCM) เป็นวงจรที่มีความสำคัญมากที่ช่วยจัดการเกี่ยวกับสัญญาณนาฬิกา ซึ่งมีอยู่ในชิพจำนวน 4 ชุด วงจรชุดนี้ช่วยให้สามารถสร้างความถี่ต่างๆที่หลากหลายได้จากออสซิเลเตอร์เพียงตัวเดียวจากภายนอก ซึ่งจะสามารถทำการ synchronize กับสัญญาณนาฬิกาจากออสซิเลเตอร์ได้อีกด้วย



รูปที่ 4 สัญลักษณ์ของวงจร DCM

DCM จะทำงานในหน้าที่ต่อไปนี้

- หารความถี่ (Clock divider) ซึ่งเป็นวงจรที่ให้ความถี่ผลลัพธ์เท่ากับความถี่ที่ป้อนให้หารด้วยตัวเลขดังต่อไปนี้ 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15 หรือ 16

- สร้างความถี่สองเท่า (Double Clock Frequency)

- Digital Frequency Synthesizer (DFS) เป็นวงจรที่สามารถกำหนดให้ความถี่ผลลัพธ์เท่ากับผลคูณของความถี่ที่ป้อนให้กับอัตราส่วน M/D โดยที่ M = 2 ถึง 32 และ D = 1 ถึง 32 ตัวอย่างเช่น Fin = 25 MHz ถ้าต้องการ Fout = 200 MHz ทำได้โดยเลือก M = 8, D = 1 วงจร DFS นี้นำไปใช้งานเช่น การสร้างวงจรเปลี่ยนจากการส่งข้อมูลแบบขนานเป็นอนุกรม ซึ่งต้องสร้างสัญญาณนาฬิกาสูงกว่าของเดิมเช่น 10 – 11 เท่า หรืองานอื่นที่ต้องการใช้วงจร Frequency Synthesizer

- Delay-Locked Loop (DLL) เป็นวงจรใช้แก้ปัญหาการเลื่อนเฟสในวงจรให้กลับมาตรงตามเฟสที่ต้องการ

- Quadrant Phase Shift เป็นวงจรเลื่อนเฟส 90, 180 และ 270 องศาตามลำดับ

- Fine Phase Shift เป็นวงจรใช้ในการเลื่อนเฟสอย่างละเอียด มีความละเอียดอยู่ที่ 1/256 เท่าของคาบความถี่

1.1.3 Digitally Controlled Impedance (DCI)

ใช้ป้องกันสัญญาณสะท้อนใน PCB โดยการควบคุมเอาท์พุตอิมพีแดนซ์ที่เหมาะสม

(รายละเอียดของการใช้บอร์ดดูได้จากคู่มือบอร์ดท้ายการทดลอง)

1.2 แนะนำภาษา Verilog

โครงสร้างของ Verilog คือ

- modules
- ports

```
module <module_name> (<portlist>);
                  .
                  .        // module components
                  .
endmodule
```

```
module top;
  type1 childA(ports...); // "ports..." indicates a port list
  type2 childB(ports...); // which will be explained later
endmodule

module type1(ports...);
  type3 leaf1(ports...);
  type3 leaf2(ports...);
endmodule

module type2(ports...);
  type3 leaf3(ports...);
  type1 node1(ports...);
endmodule

module type3(ports...);
  // this module does not instantiate any other modules
endmodule
```

ตัวอย่างโปรแกรมโค้ด Verilog

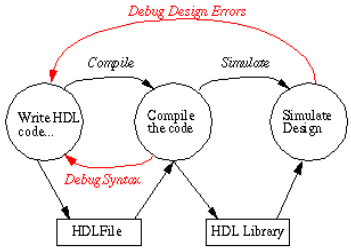
วงจรบวกเลขฐานสองขนาด 8 บิต

```
module adder(A, B, SUM);
input [7:0] A;
input [7:0] B;
output [7:0] SUM;

assign SUM = A + B;
endmodule
```

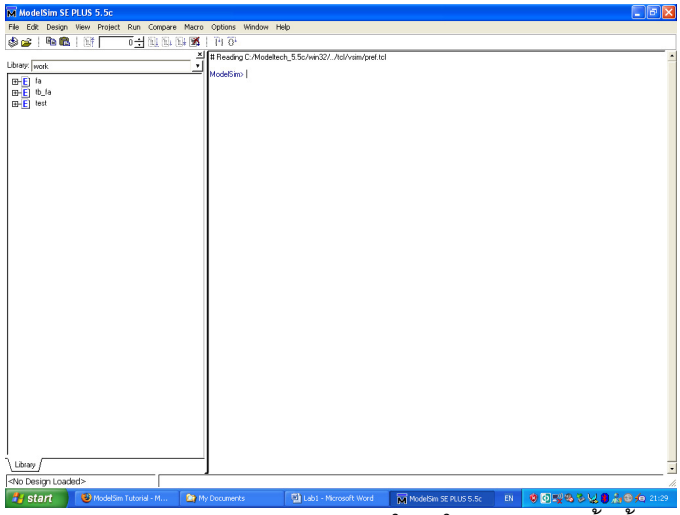
วงจร Latch ทำการเก็บค่าเมื่อสัญญาณทริกที่ขอบขาขึ้น

```
module latch (G, D, Q);
input G, D;
output Q;
reg Q;

always @(G or D)
begin
if (G)
       Q <= D;
end
endmodule
```

2. แนะนำการใช้งานโปรแกรมช่วยจำลองการทำงาน Modelsim



เรียกใช้ ModelSim จาก Icon  ModelSim SE 5.5c.lnk โดยที่หน้าต่างแรกของ ModelSim คือ



โค้ดที่ได้จากการออกแบบจะต้องทำการ compiler ใส่ลงใน library ดังนั้นขั้นตอนแรกสุดก่อนเริ่มทำการจำลองการทำงานคือจะต้องสร้าง library สำหรับงานชุดนั้นขึ้นมาก่อนซึ่ง library ก็คือการสร้าง directory เพื่อเก็บโมดูลของวงจรที่สร้างเรียบร้อยแล้ว ดังนั้นเราสามารถทำการเรียกใช้โมดูลที่มีอยู่ใน library ได้อีก (*สามารถสร้าง library ได้โดยการเลือกที่เมนู Design ->Create a new library*)

เมื่อทำการสร้าง library เรียบร้อยแล้วจะสามารถเริ่ม compile ได้โดยการเลือกที่เมนู *Design -> Compile จากนั้นให้เลือก library ที่สร้างขึ้นมาใหม่ พร้อมทั้งเลือกไฟล์โค้ดที่ต้องการจะ compile*

การเริ่มจำลองการทำงานจะสามารถทำได้หลังจากที่การ Compile เสร็จสิ้นโดยไม่มีข้อผิดพลาด ทำได้โดยการเลือกไปที่ *Design -> Load Design* จากนั้นก็เลือกชื่อของวงจรที่ได้ทำการออกแบบ (ชื่อนี้จะสอดคล้องกับชื่อของ Module ในโค้ด)

3. แนะนำการใช้งานซอฟต์แวร์ช่วยออกแบบ Xilinx (ISE Xilinx or Xilinx Webpack)

การออกแบบวงจรดิจิตอลด้วย FPGA เริ่มต้นจากขั้นตอนของการทำ System Implementation จนกระทั่งโปรแกรมวงจรลงบน FPGA ทั้งหมดนี้จะทำบนซอฟต์แวร์ช่วยออกแบบ (EDA Tools) ของบริษัท Xilinx เนื่องจากบอร์ดทดลองใช้ FPGA ของบริษัทนี้ ซึ่งปัจจุบันล่าสุดมีอยู่ 2 แบบคือ ISE Webpack (สามารถดาวน์โหลดได้ฟรี) และ ISE Foundation (ต้องซื้อ) ซึ่งในขณะนี้ได้พัฒนาเป็นเวอร์ชั่นที่ 8.2 โดยทำงานบนระบบปฏิบัติการดังนี้

Platform Type	Version Number
Windows®	2000 Pro SP2, SP3, SP4 XP Pro SP1, SP2
Japanese Windows®	2000 Pro SP2, SP3, SP4 XP Pro SP1, SP2
Chinese Windows®	2000 Pro SP2, SP3, SP4 XP Pro SP1 XP Pro SP2
Korean Windows®	2000 Pro SP2, SP3, SP4 XP Pro SP1 XP Pro SP2
Linux®	Red Hat® Enterprise WS 3.0 32-bit/64-bit Red Hat Enterprise WS 4.0 32-bit/64-bit (Limited Support)
Solaris®	8, 9

สำหรับ ISE Foundation

Platform and System Requirements

IBM PC or Compatible

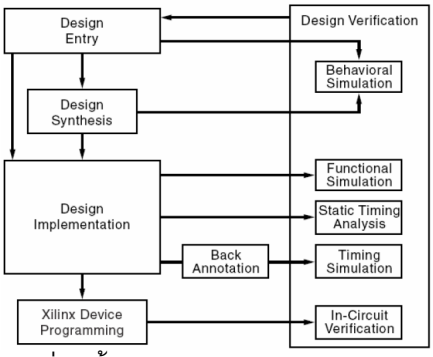
- Windows XP, Windows 2000 w/SP2 (Chinese, Korean, US, Japanese)

UNIX

- Red Hat Enterprise Linux 3

สำหรับ ISE WebPack

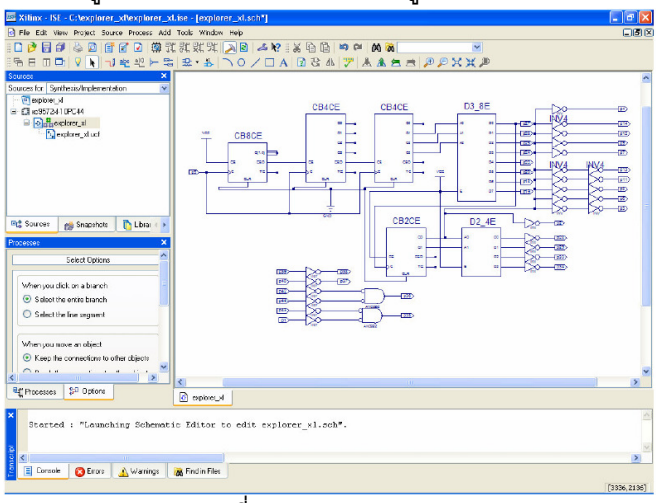
ขั้นตอนของการออกแบบวงจรดิจิตอลด้วย FPGA สามารถสรุปได้ดังรูปที่ 5



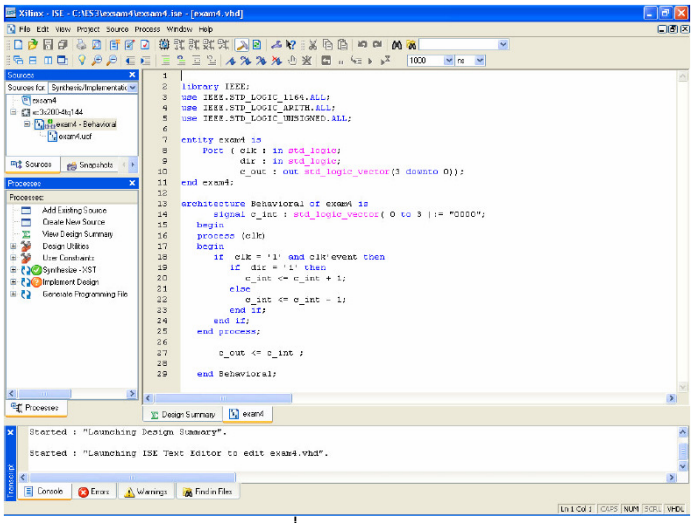
รูปที่ 5 ขั้นตอนการออกแบบวงจรดิจิตอล

3.1 การออกแบบวงจร (Design Entry)

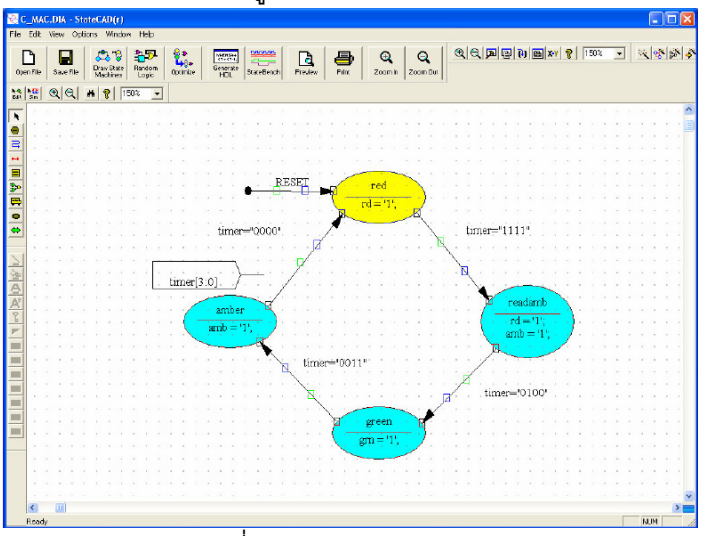
เป็นการนำเอาวงจรที่ต้องการมาออกแบบด้วยวิธีการต่าง ๆเช่น วิธีการวาดผังวงจร (Schematic) ดังรูปที่ 6 ออกแบบด้วยภาษาระดับสูง HDL (สามารถใช้งานทั้งภาษา VHDL และ Verilog) ดังรูปที่ 7 และการออกแบบด้วยวิธีการเขียนแผนภูมิสถานะ (State Diagram) ดังรูปที่ 8



รูปที่ 6 Schematic Entry



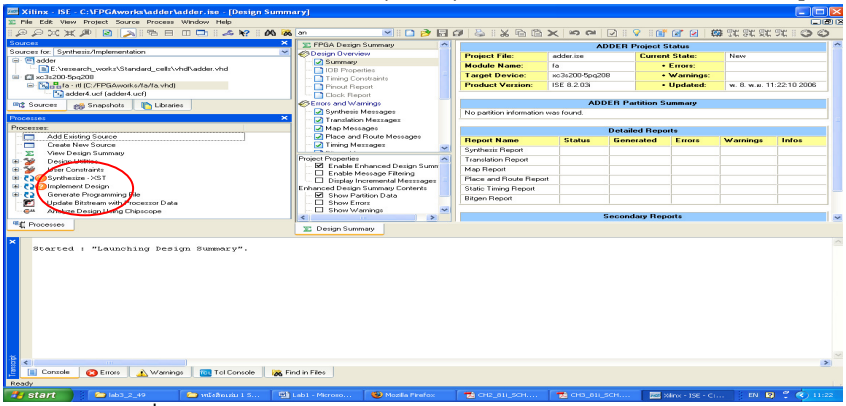
รูปที่ 7 HDL Entry



รูปที่ 8 State Diagram Entry

3.2 การสังเคราะห์วงจร

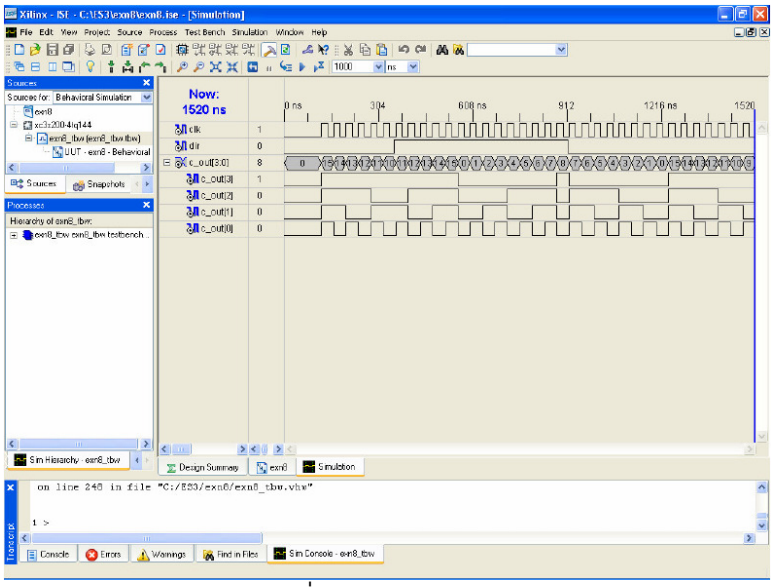
เป็นการแปลงโค้ดที่ได้ออกแบบวงจรด้วยภาษาระดับสูงเป็นวงจรในระดับเกต (Gate-level) โดยใช้ Xilinx Synthesis Tool (XST) โดยวงจรจะถูกเขียนอยู่ในรูปแบบของ text file ที่เรียกว่า netlist ที่เป็นการระบุรายการอุปกรณ์ (Component) ในระดับเกตและระบุรายการที่สายสัญญาณ (Net) แต่ละเส้นที่เชื่อมต่อถึงกัน ไฟล์ netlist อาจอยู่ในรูปแบบของ Xilinx netlist format (XNF) หรือ EDIF (อี-ดิฟ) ที่เป็นมาตรฐานของภาคอุตสาหกรรม ซึ่งการสังเคราะห์วงจรจะต้องระบุชนิด รุ่นของ FPGA ที่ต้องการใช้งานดังรูปที่9



รูปที่ 9 หน้าต่างแสดงเมนูของการเลือกทำการสังเคราะห์วงจร

3.3 การตรวจสอบวงจร

ในที่นี้ใช้ ISE Simulator ที่มีกับ ISE Foundation และ WebPack รุ่น 8.1 ขึ้นไปดังรูปที่ 10 ซึ่งในที่นี้อาจจะใช้งาน ModelSim แทนก็ได้



รูปที่ 10 ISE Simulator

โดยการจำลองการทำงานจะแบ่งออกได้เป็น 3 ระดับ

- Behavioral simulation เป็นการจำลองเฉพาะพฤติกรรมของวงจรโดยยังไม่คิดรวมถึงโครงสร้างของวงจรภายใน ซึ่งโดยปกติแล้วโค้ดแบบนี้จะไม่สามารถนำไปทำงานสังเคราะห์วงจรได้

- Functional simulation เป็นการเขียนโค้ดในระดับของ RTL (Register Transfer Level) ซึ่งสามารถนำไปทำงานสังเคราะห์วงจรได้

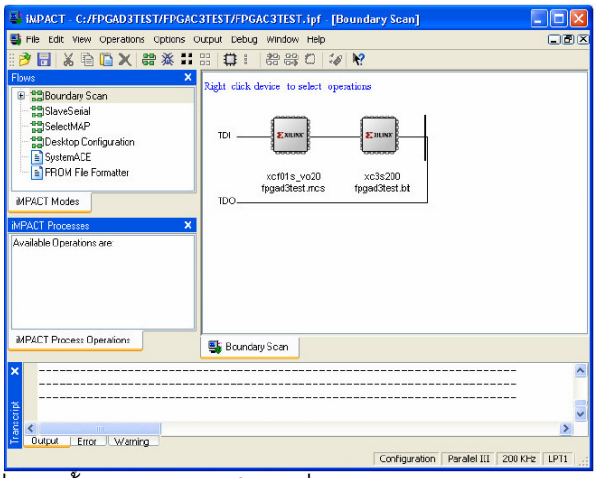
- Timing simulation เป็นการจำลองการทำงานเพื่อตรวจสอบค่าหน่วงของเวลาที่จะเกิดขึ้นจริงในขณะทำงาน ดังนั้นจึงนับได้ว่าเป็นการจำลองการทำงานที่ใกล้เคียงกับ Hardware จริงมากที่สุด

3.4 Design Implementation

เป็นการแปลง (Translate) ไฟล์ netlist มาทำการ optimize วงจร แล้วทำการ map วงจรที่ได้ลงสู่ FPGA ที่ได้เลือก แล้วทำการลองวาง (Place) บน Hardware แล้วจึงทำการเชื่อมต่อแต่ละส่วนเข้าด้วยกัน (Route)

3.5 การโปรแกรมข้อมูลวงจรลงชิพ

สามารถทำได้โดยการนำไฟล์สกุล .bit (Format ของ FPGA) ดังรูปที่ 11



รูปที่ 11 ขั้นตอนการดาวน์โหลดที่จอคอมพิวเตอร์ลงบนชิพ FPGA

4. การทดลอง

4.1 การออกแบบวงจรนับ (Counter)

ออกแบบวงจรนับขนาด 4 บิตที่มีการเคลียร์ค่าแบบ asynchronous จงวาดรูป block diagram ของวงจร

รายการ	ลายเซ็นต์	วัน / เดือน / ปี
# check 1- Block Diagram ของ asynchronous counter 4 บิต		
# check 2- ผลการทำงานของวงจรนับขนาด 4 บิตบน ModelSim หรือ ISESimulator		

4.2 การออกแบบวงจรถอดรหัส (Decoder)

ออกแบบวงจรถอดรหัส 3 to 8

รายการ	ลายเซ็นต์	วัน / เดือน / ปี
# check 1- Block Diagram ของวงจรถอดรหัส 3 to 8		
# check 2- ผลการจำลองการทำงานของวงจรถอดรหัสที่ได้ออกแบบไว้บน ModelSim หรือ ISESimulator		

5. โจทย์ปัญหาการออกแบบวงจรอย่างง่าย

5.1 จงออกแบบวงจรนับ 0 – 9, A-F และแสดงค่าออกทาง 7-segment

รายการ	ลายเซ็นต์	วัน / เดือน / ปี
# check 1- Block Diagram ของระบบที่ได้ออกแบบ		
# check 2- การทำงานบนบอร์ด Spartan ถูกต้อง		

5.2 จงออกแบบวงจรที่สามารถเลือกนับขึ้นหรือนับลง 0-9, A-F พร้อมทั้งแสดงค่าออกทาง 7-segment

รายการ	ลายเซ็นต์	วัน / เดือน / ปี
# check 1- Block Diagram ของระบบที่ได้ออกแบบ		
# check 2- การทำงานบนบอร์ด Spartan ถูกต้อง		