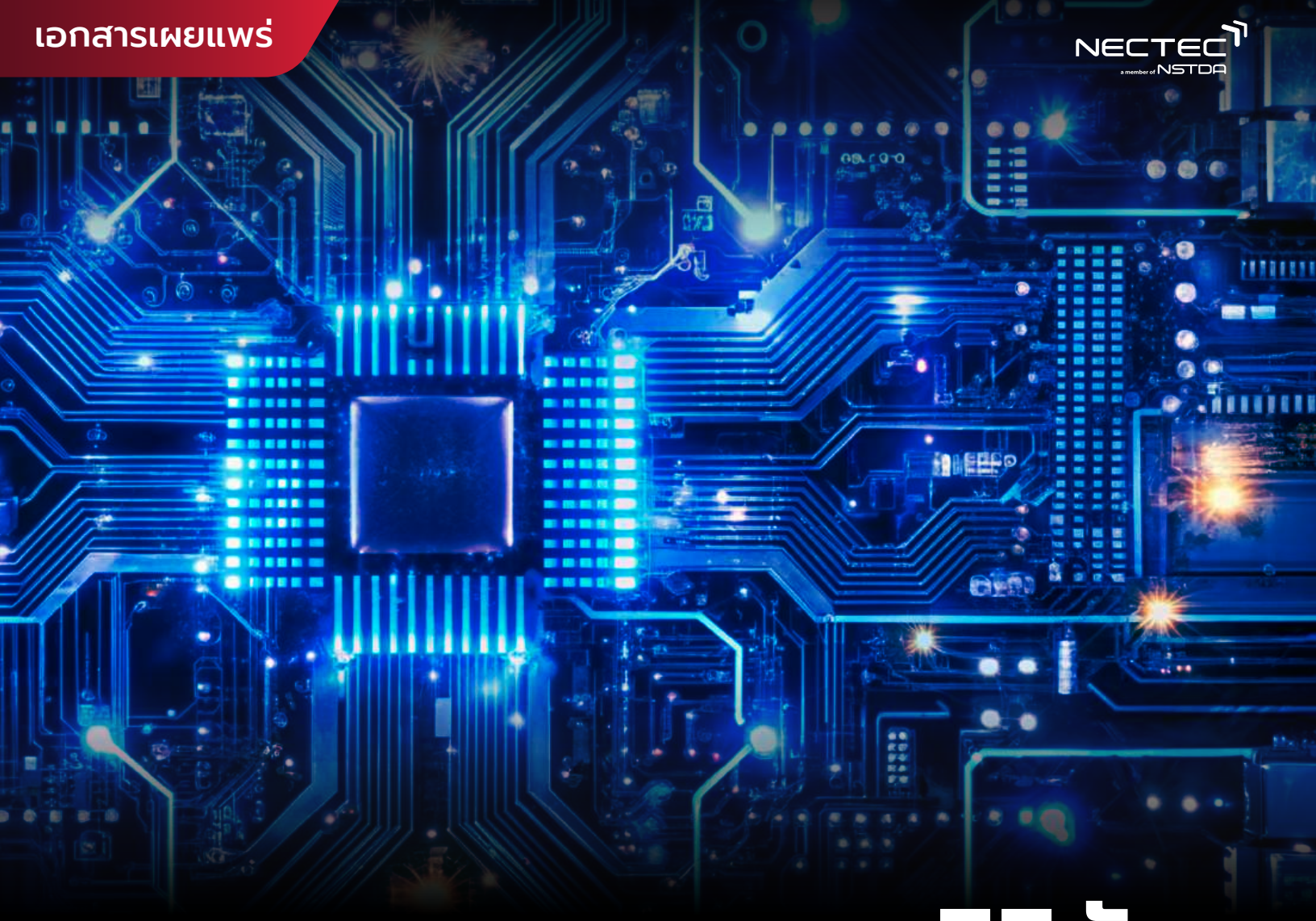
เอกสารเผยแพร่



การออกแบบวงจรดิจิทัล
ด้วยเอฟพีจีเอราคาประหยัด จาก GOWIN

ชำนาญ ปัญญาใส
ทีมระบบไซเบอร์-กายภาพ (CPS)
กลุ่มวิจัยไอโอทีและระบบอัตโนมัติสำหรับงานอุตสาหกรรม (IIARG)
ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

มาเพิ่มทักษะการออกแบบวงจรดิจิตัลด้วยเอฟพีจีเอราคาประหยัดจาก GOWIN

ชำนาญ ปัญญาใส

ทีมระบบไซเบอร์-กายภาพ (CPS)

กลุ่มวิจัยไอโอทีและระบบอัตโนมัติสำหรับงานอุตสาหกรรม (IIARG)

ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

สารบัญ

มาเพิ่มทักษะการออกแบบวงจรดิจิตัลด้วยเอฟพีจีเอราคาประหยัดจาก GOWIN

บทนำ

FPGA (Field-Programmable Gate Array) คืออุปกรณ์วงจรรวมที่สามารถกำหนดการทำงานได้ตามความต้องการของผู้ใช้งาน โดยใช้ภาษาการออกแบบวงจรขั้นสูง เช่น VHDL, Verilog หรือ SystemVerilog ซึ่ง FPGA มักจะถูกใช้ในการออกแบบและพัฒนาระบบดิจิตัลที่มีความซับซ้อนและมีความยืดหยุ่นสูง ผู้ผลิต FPGA ที่เป็นที่รู้จักมากมักมาจากประเทศฝั่งตะวันตก เช่น AMD (ชื่อเดิมคือ Xilinx) หรือ Intel (ชื่อเดิมคือ Altera)

ประโยชน์ของ FPGA

FPGA (Field-Programmable Gate Array) เป็นอุปกรณ์ที่มีความยืดหยุ่นและสามารถกำหนดการทำงานได้ตามความต้องการของผู้ใช้งาน ซึ่งมีประโยชน์หลายประการ ดังนี้:

1. ความยืดหยุ่นในการออกแบบ

FPGA สามารถโปรแกรมใหม่ได้ตลอดเวลา ทำให้สามารถปรับเปลี่ยนฟังก์ชันการทำงานตามความต้องการที่เปลี่ยนแปลงไปได้ ซึ่งต่างจาก ASIC (Application-Specific Integrated Circuit) ที่มีฟังก์ชันคงที่หลังการผลิต

2. การพัฒนาและทดสอบอย่างรวดเร็ว

เนื่องจาก FPGA สามารถโปรแกรมและทดสอบได้ทันที ทำให้การพัฒนาระบบและการทดสอบฟังก์ชันการทำงานของวงจรทำได้อย่างรวดเร็ว ไม่ต้องรอเวลาผลิตชิปแบบดั้งเดิม

3. ลดต้นทุนการพัฒนาในระยะยาว

การใช้ FPGA ช่วยลดต้นทุนการพัฒนาเนื่องจากไม่ต้องเสียค่าใช้จ่ายในการผลิตชิปใหม่ทุกครั้งที่ต้องการปรับเปลี่ยนฟังก์ชัน อีกทั้งยังลดความเสี่ยงในการพัฒนาเนื่องจากสามารถทดสอบฟังก์ชันการทำงานได้ก่อนผลิตจริง

4. การใช้งานที่หลากหลาย

FPGA สามารถนำไปใช้ในหลากหลายอุตสาหกรรม เช่น การสื่อสาร, การแพทย์, การบิน, และระบบควบคุมอุตสาหกรรม เนื่องจากสามารถออกแบบให้รองรับฟังก์ชันการทำงานที่หลากหลายได้

5. ประสิทธิภาพสูง

FPGA มีการเชื่อมต่อภายในที่รวดเร็วและสามารถประมวลผลข้อมูลแบบขนานได้ ทำให้มีประสิทธิภาพสูงในการประมวลผลสัญญาณและการคำนวณที่ซับซ้อน

6. การพัฒนาแบบเปิดกว้าง
FPGA มีเครื่องมือและซอฟต์แวร์ที่หลากหลายสำหรับการออกแบบและพัฒนา ไม่ว่าจะเป็นเครื่องมือจากผู้ผลิตชิปเองหรือเครื่องมือจากบุคคลที่สาม ทำให้ผู้พัฒนามีทางเลือกมากมายในการทำงาน

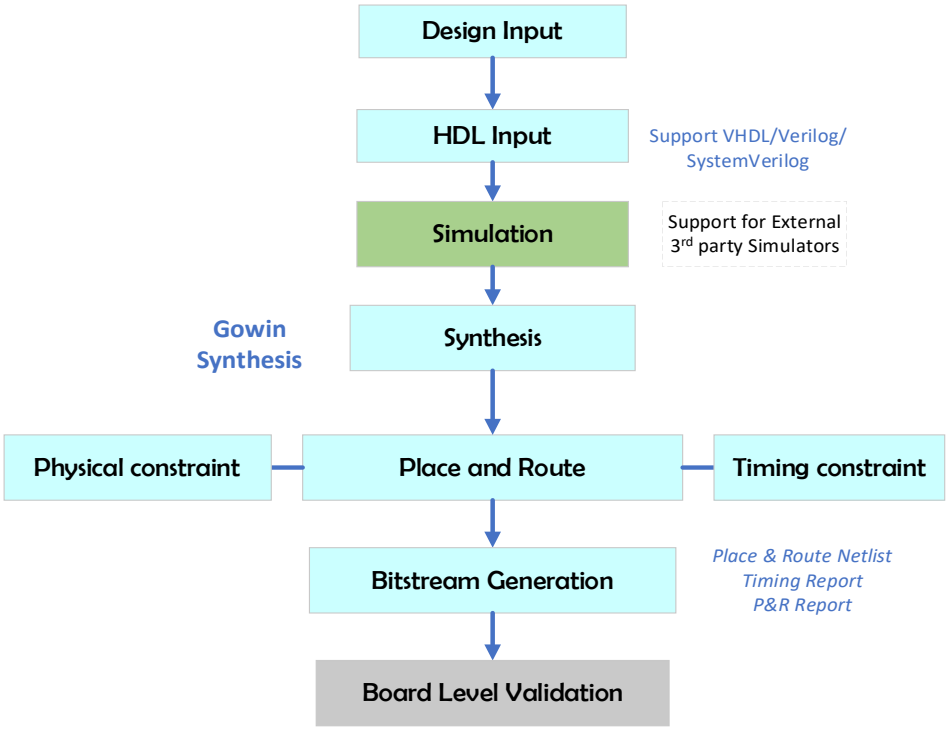
7. การรองรับมาตรฐานใหม่ๆ
FPGA สามารถปรับเปลี่ยนเพื่อรองรับมาตรฐานการสื่อสารหรือโปรโตคอลใหม่ๆ ได้โดยไม่ต้องเปลี่ยนอุปกรณ์ฮาร์ดแวร์ ทำให้สามารถอัพเดตและปรับปรุงระบบได้ง่าย

FPGA GOWIN

GOWIN เป็นบริษัทผู้ผลิต FPGA จากประเทศจีนที่นำเสนอผลิตภัณฑ์ FPGA ราคาประหยัด เหมาะสำหรับการพัฒนาวงจรดิจิตัลหลากหลายประเภท หรือสำหรับผู้ที่เริ่มต้นศึกษาเกี่ยวกับการออกแบบวงจรดิจิตัลด้วย FPGA บทความนี้จะนำเสนอขั้นตอนการออกแบบวงจรด้วย FPGA ของ GOWIN โดยครอบคลุมขั้นตอนตั้งแต่การเลือกชิป FPGA, การเขียนโค้ด HDL, การสังเคราะห์โค้ด, การโปรแกรมชิป FPGA และการทดสอบวงจร

การออกแบบวงจรด้วย FPGA ของ GOWIN จะเริ่มจากการกำหนดสถาปัตยกรรมและฟังก์ชันที่ต้องการในระบบ จากนั้นจะเขียนโค้ด HDL เพื่ออธิบายการทำงานของวงจร ทำการจำลอง (Simulation) เพื่อตรวจสอบความถูกต้องของวงจร และสังเคราะห์โค้ดเป็นเน็ตลิสต์ จากนั้นทำการวางแผนการวางตำแหน่งและการเชื่อมต่อขององค์ประกอบต่างๆ ใน FPGA สร้างไฟล์บิตสตรีมเพื่อใช้ในการโปรแกรมลงบน FPGA และทำการตรวจสอบการทำงานของวงจรในระดับบอร์ด
ด้วยความประหยัดและความยืดหยุ่นของ FPGA จาก GOWIN ทำให้เหมาะสำหรับผู้ที่ต้องการศึกษาและทดลองการออกแบบวงจรดิจิตัลอย่างง่ายดายและไม่ต้องใช้งบประมาณสูง

ขั้นตอนการออกแบบวงจร FPGA ของ GOWIN โดยมีลำดับขั้นตอนคร่าวๆ ดังนี้



รูปที่ 1 ขั้นตอนการออกแบบวงจร FPGA ของ GOWIN

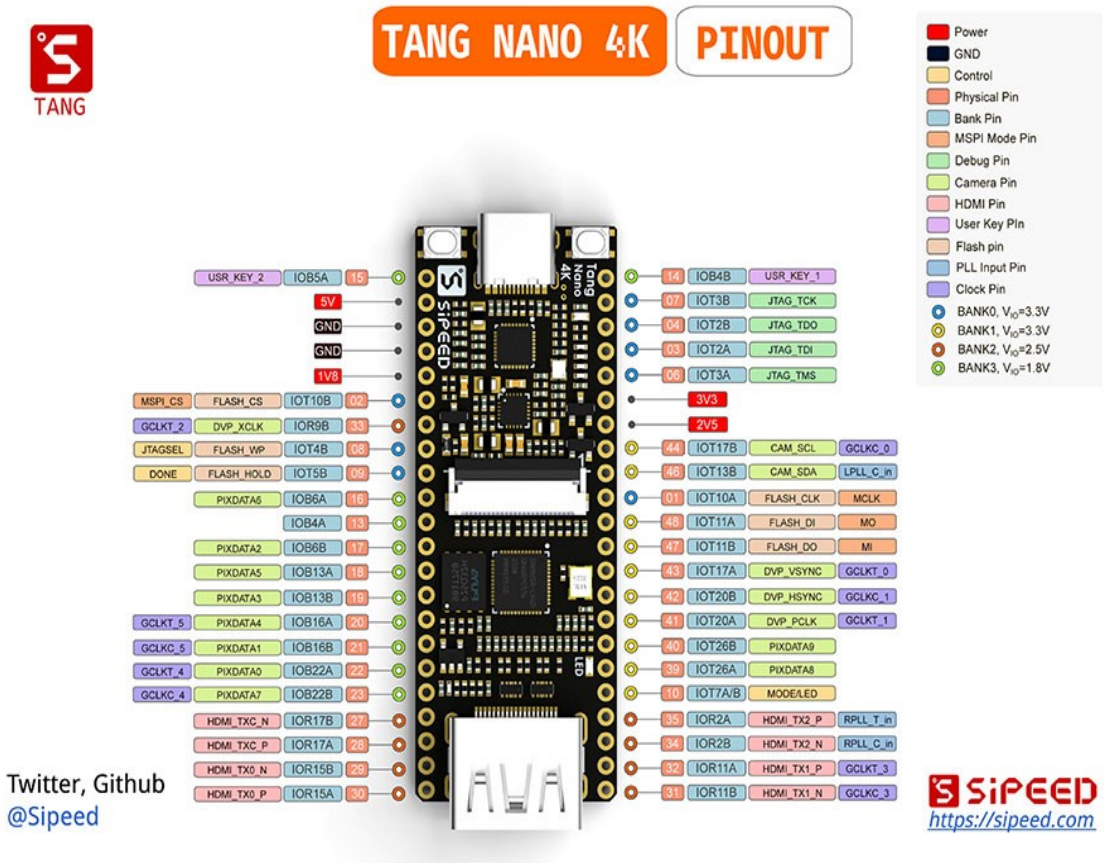
1. **Design Input**:
    - ขั้นตอนแรกของการออกแบบ FPGA คือการกำหนดความต้องการและการวางแผนการออกแบบ โดยจะมีการกำหนดสถาปัตยกรรมและฟังก์ชันที่ต้องการในระบบ
2. **HDL Input**:
    - หลังจากการกำหนดแผนการออกแบบแล้ว ขั้นตอนต่อไปคือการเขียนโค้ด HDL (Hardware Description Language) ซึ่งอาจจะเป็น VHDL, Verilog หรือ SystemVerilog เพื่ออธิบายการทำงานของวงจรที่ต้องการ
3. **Simulation**:
    - เมื่อโค้ด HDL เขียนเสร็จแล้ว จะมีการทำการจำลอง (Simulation) เพื่อตรวจสอบความถูกต้องของการทำงานของวงจร โดยใช้ซอฟต์แวร์จำลอง เช่น ModelSim หรือ QuestaSim
4. **Synthesis**:
    - หากการจำลองผ่านไปด้วยดี โค้ด HDL จะถูกส่งไปยังขั้นตอนการสังเคราะห์ (Synthesis) เพื่อแปลงโค้ดเป็นเน็ตลิสต์ (Netlist) ซึ่งเป็นการอธิบายการเชื่อมต่อของวงจรในระดับลอจิก
5. **Place and Route**:
    - ขั้นตอนนี้จะมีการกำหนดข้อกำหนดทางกายภาพ (Physical Constraints) และข้อกำหนดเวลา (Timing Constraints) เพื่อให้แน่ใจว่าวงจรจะทำงานได้ตามที่ต้องการ จากนั้นจะทำการวางแผนการวางตำแหน่งและการเชื่อมต่อ (Place and Route) ขององค์ประกอบต่างๆ ใน FPGA
6. **Bitstream Generation**:

- หลังจากการวางและการเชื่อมต่อเสร็จสิ้น จะมีการสร้างไฟล์บิตสตรีม (Bitstream) ซึ่งเป็นไฟล์ที่ใช้ในการโปรแกรม FPGA

7. **Board Level Validation**:
   - ขั้นตอนสุดท้ายคือการนำไฟล์บิตสตรีมไปโปรแกรมลงบน FPGA จริง และทำการตรวจสอบการทำงานในระดับบอร์ดเพื่อให้แน่ใจว่าวงจรทำงานได้ตามที่ออกแบบ

บอร์ดพัฒนาเอฟพีจีเอ GOWIN

GOWIN นำเสนอชิป FPGA หลายรุ่น แต่ละรุ่นมีทรัพยากรลอจิก หน่วยความจำ และบล็อก IP ที่แตกต่างกัน ผู้ใช้จึงต้องเลือกชิปให้เหมาะสมกับงาน ตัวอย่างบอร์ดพัฒนาเอฟพีจีเอของ GOWIN ได้แก่ SiPEED Tank Nano 4k ซึ่งใช้ชิป FPGA รุ่น GW1NSR-LV4C



รูปที่ 2 GOWIN FPGA board TANG Nano 4k แสดงตำแหน่งการเชื่อมต่อขาสัญญาณกับเอพพีจีเอ

โครงสร้างภายในของ FPGA GOWIN รุ่น GW1NSR-LV4C

FPGA รุ่น GW1NSR-LV4C จาก GOWIN มีโครงสร้างภายในที่ประกอบด้วยองค์ประกอบหลักหลายส่วน ซึ่งแต่ละส่วนมีหน้าที่และคุณสมบัติที่เฉพาะเจาะจง ดังนี้:

1. Logic Fabric

Logic Fabric เป็นส่วนที่ประกอบด้วย CLU (Configurable Logic Unit) ซึ่งเชื่อมต่อกันผ่าน Interconnect โครงสร้างนี้ช่วยให้ผู้ใช้สามารถกำหนดค่า FPGA เพื่อใช้งาน logic function ที่ต้องการ

- **CLU (Configurable Logic Unit):** แต่ละ CLU ประกอบด้วย LUT (Look-Up Table) 4 ตัว, flip-flop 4 ตัว และ MUX (Multiplexer) 4 ตัว
    - **LUT (Look-Up Table):** สามารถกำหนดค่าเพื่อใช้งาน logic function ใดก็ได้ที่มี input 6 ตัว และ output 1 ตัว
    - **Interconnect:** ช่วยให้ CLU เชื่อมต่อกันในรูปแบบต่างๆ

2. CPU Core

หน่วยประมวลผลแบบฝังตัวที่ใช้ใน FPGA รุ่นนี้คือ 32-bit ARM Cortex M3 ซึ่งช่วยเพิ่มความสามารถในการประมวลผลและการควบคุมวงจร

3. Memory

หน่วยความจำภายใน FPGA ประกอบด้วย Block RAM, CAM และ FIFO ซึ่งสามารถใช้เพื่อจัดเก็บข้อมูลและโค้ด

- **Block RAM:** มีขนาด 1Kb ถึง 32Kb
- **CAM:** มีขนาด 1Kb ถึง 16Kb
- **FIFO:** มีขนาด 8 byte ถึง 4Kb

4. DSP Blocks

ใช้สำหรับการประมวลผลสัญญาณดิจิทัล DSP Block แต่ละตัวมีตัวคูณ 18x18 บิต และ accumulator 48 บิต

- **DSP Block:** สามารถใช้เพื่อใช้งานฟังก์ชั่น DSP ต่างๆ เช่น การกรองสัญญาณ FIR และ IIR, การแปลง FFT และ IFFT

5. I/O Cells

เชื่อมต่อสัญญาณภายใน FPGA กับโลกภายนอก I/O Cell แต่ละตัวรองรับสัญญาณไฟฟ้าแบบ LVCMOS, SSTL, HSTL และ HBM

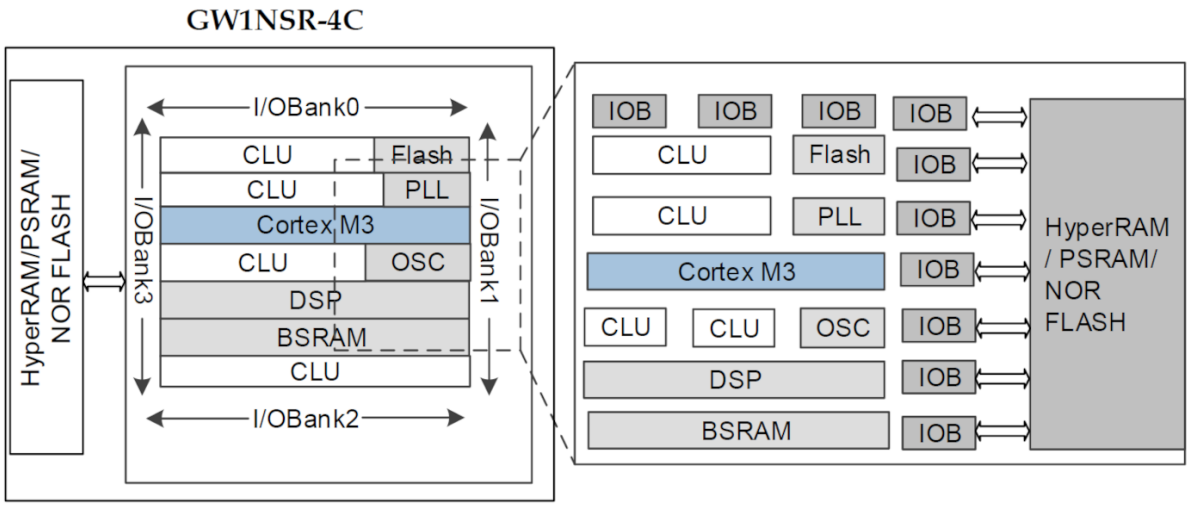
- **I/O Cell:** แต่ละตัวสามารถกำหนดค่าเป็น input, output หรือ bidirectional

6. PLL (Phase Locked Loop)

ใช้สำหรับสร้างสัญญาณนาฬิกาที่มีความเสถียรสูง เพื่อให้วงจรภายใน FPGA ทำงานได้อย่างถูกต้องและแม่นยำ

7. OSC (Oscillator)

วงจรนาฬิกาภายใน FPGA ที่ใช้ในการกำหนดจังหวะการทำงานของวงจรต่างๆ



รูปที่ 3 แสดงแผนผังโครงสร้างสถาปตยกรรมภายในชิป **GW1NSR-LV4C** (ที่มา: Gowin Semiconductor)

- CLU (Configurable Logic Unit)
- CLS (Configurable Logic Section): LUT4 + FF
- PLL (Phase Locked Loop)
- IOB (Input/Output Block)
- BSRAM ( )
- DSP
- OSC

การเขียนโค้ด HDL ด้วยภาษา VHDL หรือ Verilog หรือ SystemVerilog

ทำการเขียนโค้ด VHDL หรือ Verilog หรือ SystemVerilog เพื่อพรรณาฟังก์ชันการทำงานของวงจร ให้ถูกต้องตามหลักไวยกรณ์ของภาษา ทั้งนี้ให้ขั้นตอนนี้เราจะต้องตรวจสอบฟังก์ชั่นการทำงานของวงจรด้วยวิธีทำการจำลองการทำงานทางลอจิก (Logic Simulation)

หมายเหตุ ด้วยซอฟต์แวร์ของ GOWIN ไม่มีส่วนของการทำการจำลองวงจร เราจึงต้องใช้โปรแกรมสำหรับการจำลองวงจร VHDL หรือ Verilog หรือ SystemVerilog จากแหล่งอื่น ซึ่งอาจจะมีค่าใช้จ่ายเพิ่มเติมแล้วแต่กรณี

ตัวอย่างโปรแกรมฟรีสำหรับจำลอง VHDL ได้แก่ GHDL (GNU VHDL), EDA Playground,

ตัวอย่างโปรแกรมฟรีสำหรับจำลอง Verilog ได้แก่ Icarus Verilog, Verilator, CircuitVerse, Xcelium,

ตัวอย่างโปรแกรมฟรีที่รองรับทั้งสองภาษา ได้แก่ ModelSim-Altera Starter Edition, QuestaSim Community Edition, SimulateMe

การสังเคราะห์โค้ด

ใช้ซอฟต์แวร์ EDA ของ GOWIN เพื่อทำการสังเคราะห์โค้ด VHDL หรือ Verilog หรือ SystemVerilog เป็นไฟล์เน็ตลิสต์ (netlist file) ซึ่งจะถูกนำไปทำการ Floorplan และวางตำแหน่งอุปกรณ์ Place & Route ก่อนที่จะสร้างเป็นไฟล์ bitstream สำหรับโปรแกรมลงชิป FPGA

การโปรแกรมชิป FPGA

จากไฟล์ bitstream ที่ได้มา ทำการดาวน์โหลดหรือโปรแกรมลงบนชิปเอฟพีจีเอ ผ่านสาย JTAG

การทดสอบวงจร

ทดสอบการทำงานของวงจรบนบอร์ดเอฟพีจีเอจริง

เครื่องมือที่ใช้

- ซอฟต์แวร์ EDA ของ GOWIN:
    - Gowin Designer: ใช้สำหรับเขียนโค้ด VHDL หรือ Verilog หรือ SystemVerilog
    - Gowin Synthesis: ใช้สำหรับสังเคราะห์โค้ด VHDL หรือ Verilog หรือ SystemVerilog
    - Gowin Programmer: ใช้สำหรับโปรแกรมชิป FPGA
- โปรแกรมเมอร์ JTAG

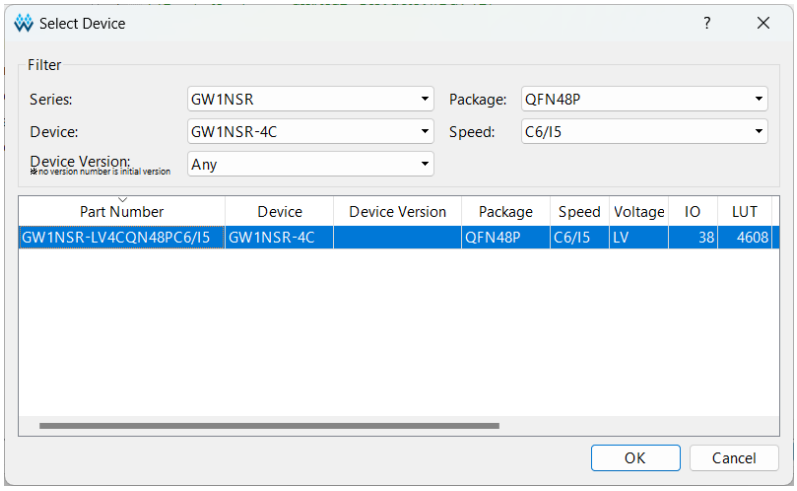
ตัวอย่างการออกแบบ

สมมติว่า ต้องการออกแบบวงจรไฟ LED กะพริบ ทุกๆ 1 วินาที โดยใช้สัญญาณนาฬิกาอ้างอิงของบอร์ด SiPEED Tank Nano 4k ที่ 27 MHz โดยทำการออกบบด้วยภาษา VHDL

1. เลือกชิป FPGA

เปิดโปรแกรม Gowin FPGA Designer แล้วเปิดโปรเจ็กต์ใหม่ New Project ตั้งชื่อโปรเจ็กต์เป็น led1

หลังจากนั้นทำการเลือกชิป FPGA ที่ต้องการใช้งานให้ตรงกับหมายเลขพาร์ต ตัวอย่างเช่น GW1NSR-LV1QN48C6/I5

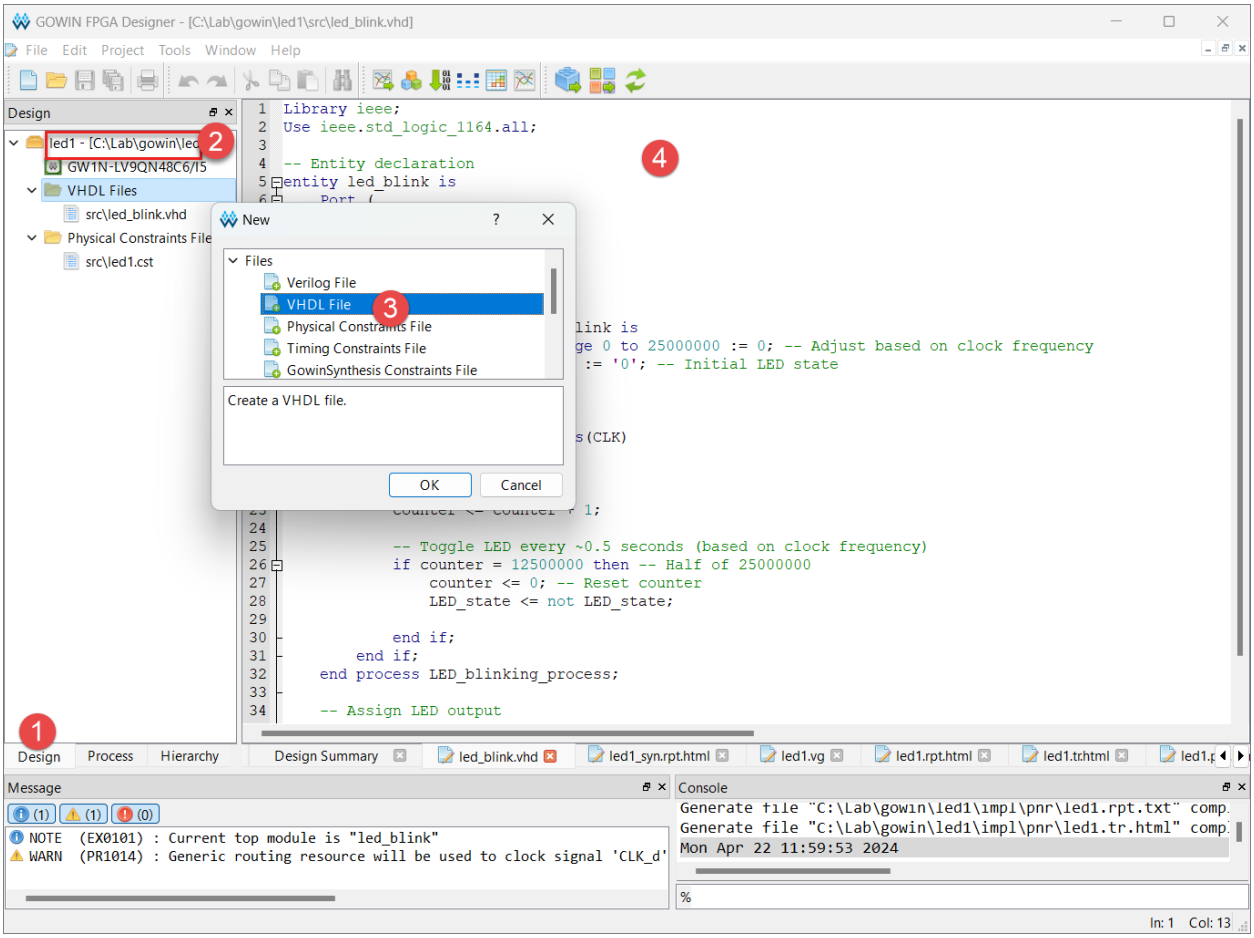


รูปที่ 4

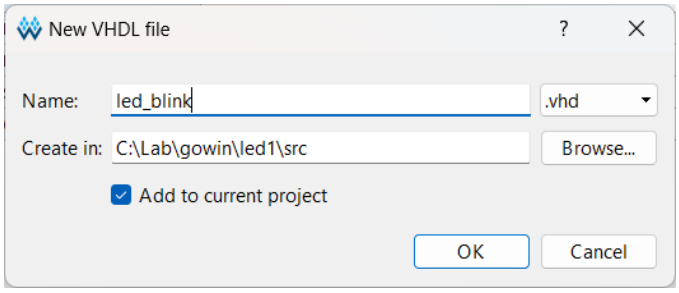
ทำการสร้างไฟล์ HDL โดยการ

1. คลิกแถบ Design
2. คลิกเม้าส์ด้านขวาเพื่อสร้าง Design file ใหม่ (New File..) หรือทำการ Add Files

3. เลือกชนิดไฟล์ ในกรณีนี้เป็น VHDL File ใส่ชื่อไฟล์เป็น led_blink.vhd
4. หน้าวินโดว์สำหรับการสร้างไฟล์



รูปที่ 5



รูปที่ 6

2. เขียนโค้ด VHDL

ตัวอย่างโค้ด VHDL

```
Library ieee;
Use ieee.std_logic_1164.all;

-- Entity declaration
entity led_blink is
    Port (
        clk : in std_logic;
        led : out std_logic
    );
end led_blink;

-- Architecture
architecture Behavioral of led_blink is
    signal counter : integer range 0 to 27000000 := 0; -- Adjust based on clock frequency, i.e. 27MHz
    signal led_state : std_logic := '0'; -- Initial LED state

begin
    -- LED blinking process
    led_blinking_process: process(clk)
    begin
        if rising_edge(clk) then
            -- Increment counter
            counter <= counter + 1;

            -- Toggle LED every ~0.5 seconds (based on clock frequency)
            if counter = 13500000 then -- Half of 27000000
                counter <= 0; -- Reset counter
                led_state <= not led_state;
            end if;
        end if;
    end process led_blinking_process;
```

```
    -- Assign LED output
    led <= led_state;

end Behavioral;
```

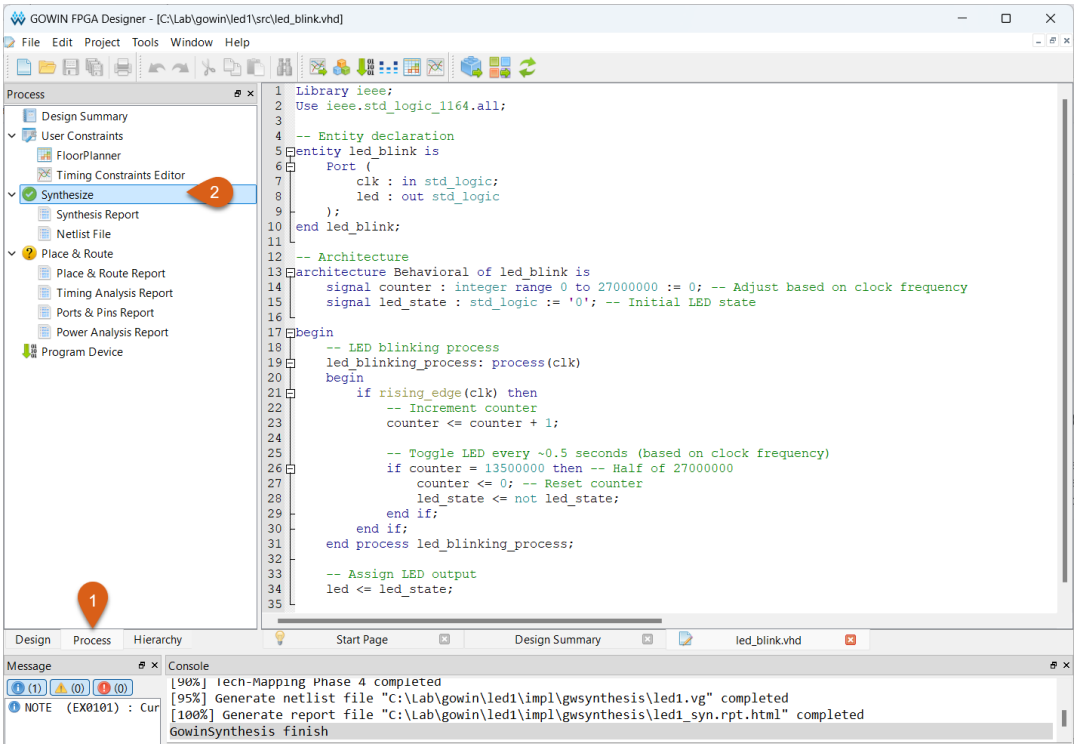
```vhdl
Library ieee;
Use ieee.std_logic_1164.all;

-- Entity declaration
entity led_blink is
    Port (
        clk : in std_logic;
        led : out std_logic
    );
end led_blink;

-- Architecture
architecture Behavioral of led_blink is
    signal counter : integer range 0 to 27000000 := 0; -- Adjust based on clock frequency
    signal led_state : std_logic := '0'; -- Initial LED state

begin
    -- LED blinking process
    led_blinking_process: process(clk)
    begin
        if rising_edge(clk) then
            -- Increment counter
            counter <= counter + 1;

            -- Toggle LED every ~0.5 seconds (based on clock frequency)
            if counter = 13500000 then -- Half of 27000000
                counter <= 0; -- Reset counter
                led_state <= not led_state;
            end if;
        end if;
    end process led_blinking_process;

    -- Assign LED output
    led <= led_state;

end Behavioral;
```

รูปที่ 7

3. สังเคราะห์โค้ด

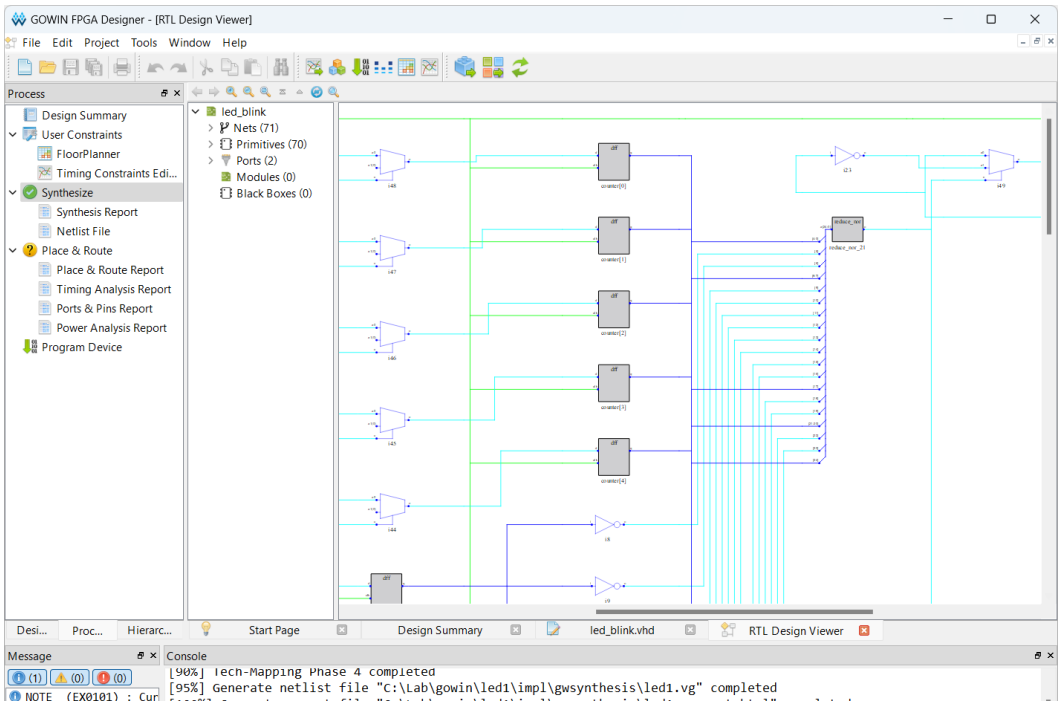
ใช้ Gowin Synthesis สังเคราะห์โค้ด VHDL คลิกที่แท็บ Process 1

รันสังเคราะห์วงจร โดยการดับเบิ้ลคลิกที่ 2 Synthesize เพื่อทำการสังเคราะห์วงจร



รูปที่ 8

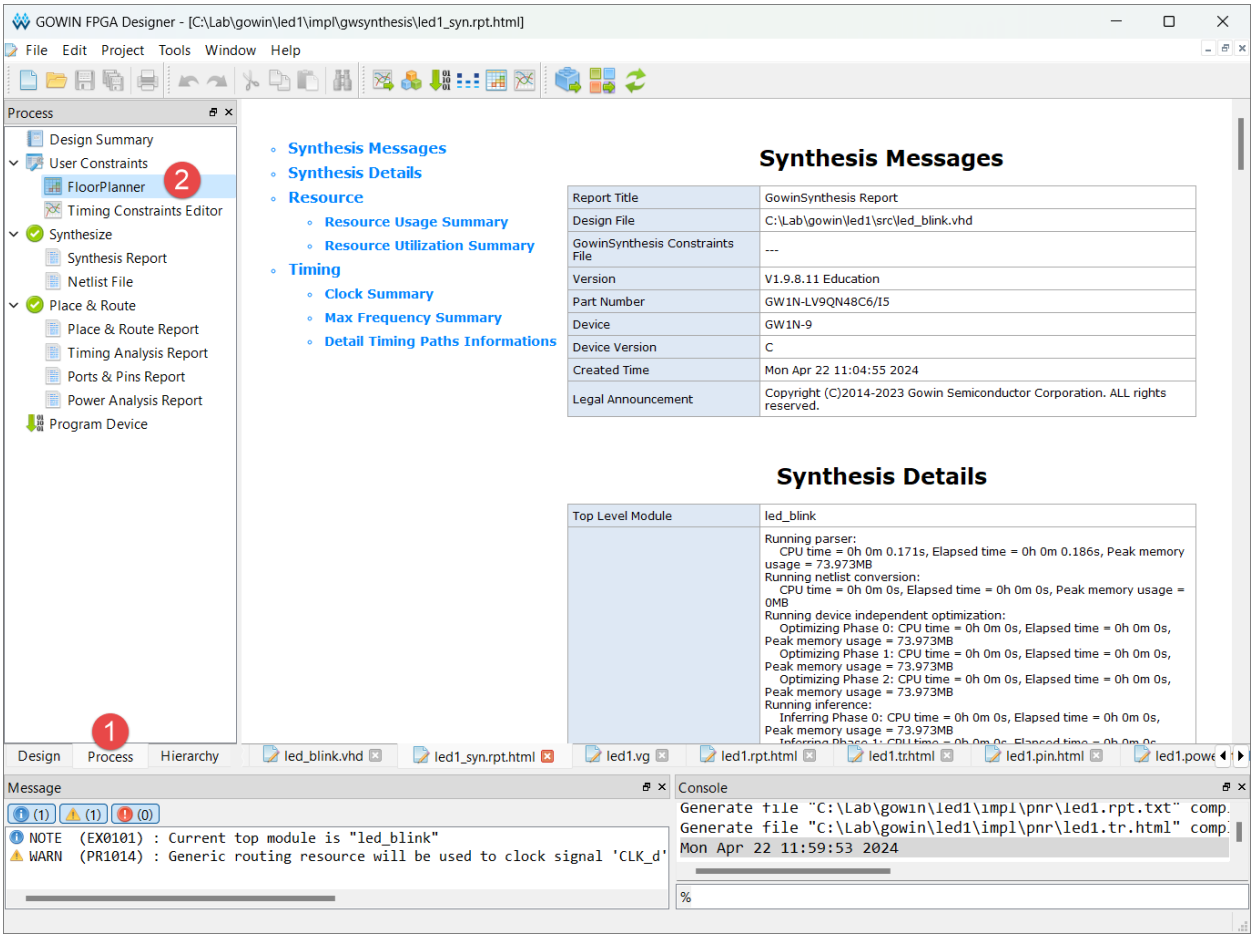
เราสามารถดูตัวอย่างวงจร (schematic) โดย คลิกที่ Tools -> Schematic Viewer -> RTL Design Viewer



รูปที่ 9

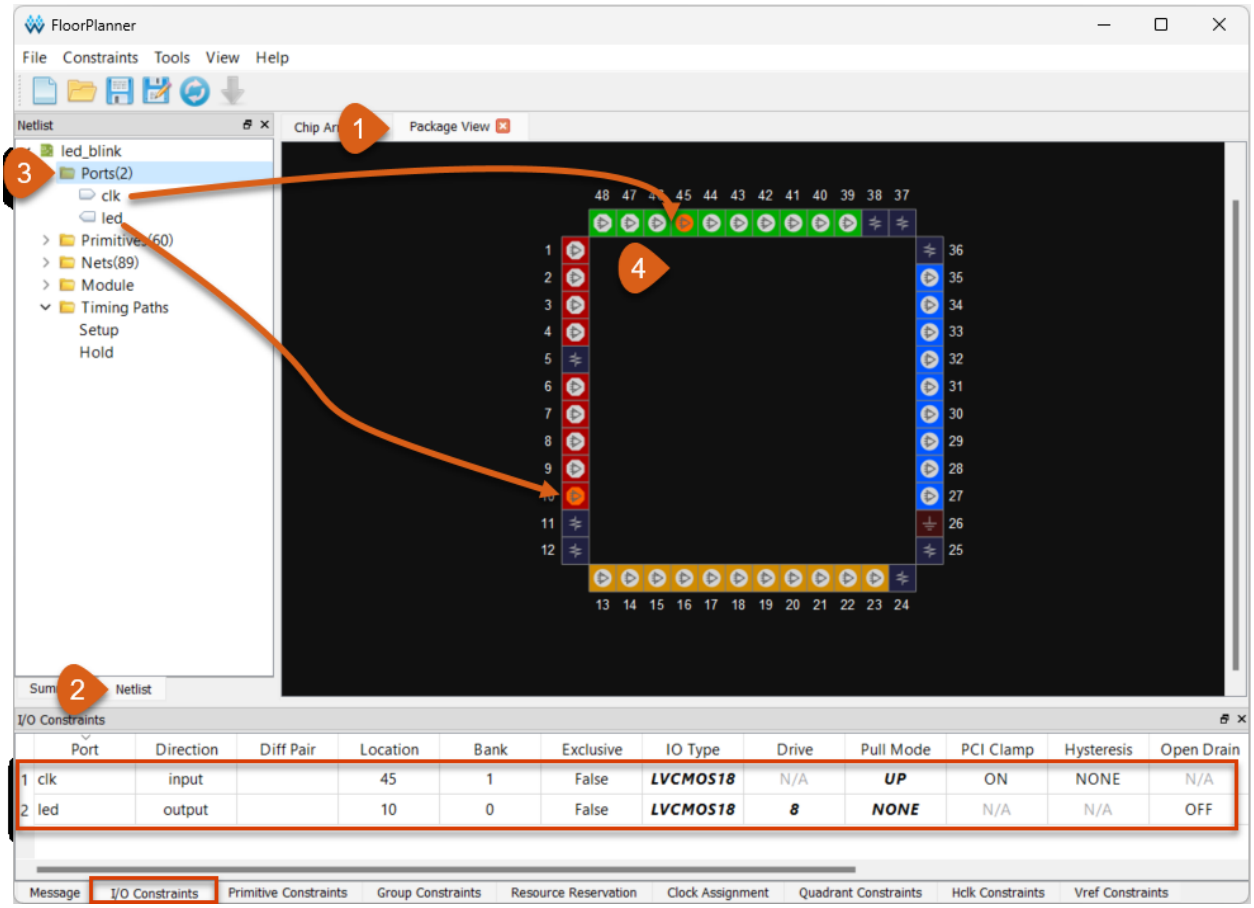
หลังจากทำการสังเคราะห์ให้ทำการกำหนดตำแหน่งขาสัญญาณต่างๆ (pin assignment) ในโปรแกรม Floorplanner

ในหน้าแท็บ Process ภายใต้ User Constraints ทำการดับเบิ้ลคลิกที่ FloorPlanner 2



รูปที่ 10

จะปรากฏวินโดว์ FloorPlanner



รูปที่ 11 หน้าจอ Floorplanner สำหรับการกำหนดตำแหน่งขาสัญญาณต่างๆ I/O Constraints

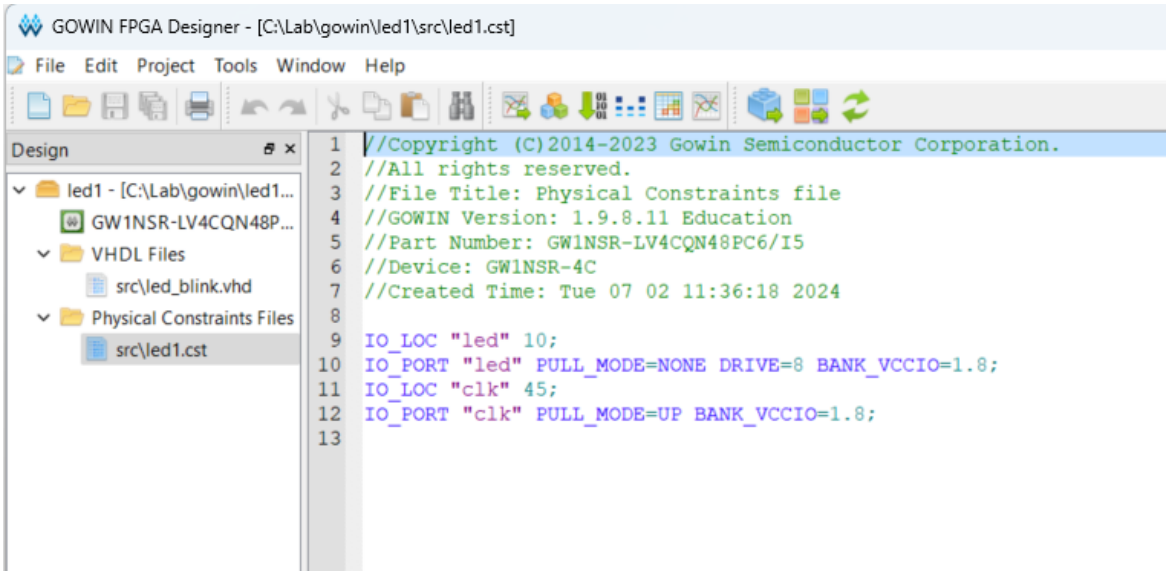
ที่วินโดว์ FloorPlanner คลิกแท็บ Package View 1 เพื่อดูมุมมองตำแหน่งพินขาของเอฟพีจีเอ และแท็บ Netlist 2 ทำการเลือกสัญญาณแต่ละสัญญาณใน Ports 3 ลากมาวางตรงตำแหน่งพินขาของเอฟพีจีเอ 4

ลากสัญญาณ clk วางที่ตำแหน่งพิน 45

ลากสัญญาณ led วางที่ตำแหน่งพิน 10

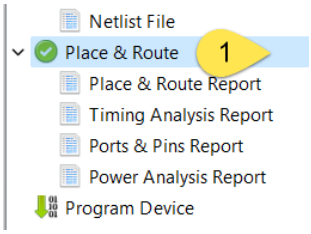
หากคลิกที่แถบ IO Constraints จะแสดงรายละเอียดสัญญาณ Port ต่างๆ ตำแหน่งขา และการเซ็ตค่าต่างๆ เราสามารถเซ็ตค่าต่างๆได้

ทั้งนี้เมื่อทำการบันทึก (Save) ไฟล์จะถูกบันทึกเป็นไฟล์นามสกุล .cst ตัวอย่างเช่น led1.cst อยู่ภายใต้ Physical Constraint File



รูปที่ 12

จากนั้น รัน Place & Route

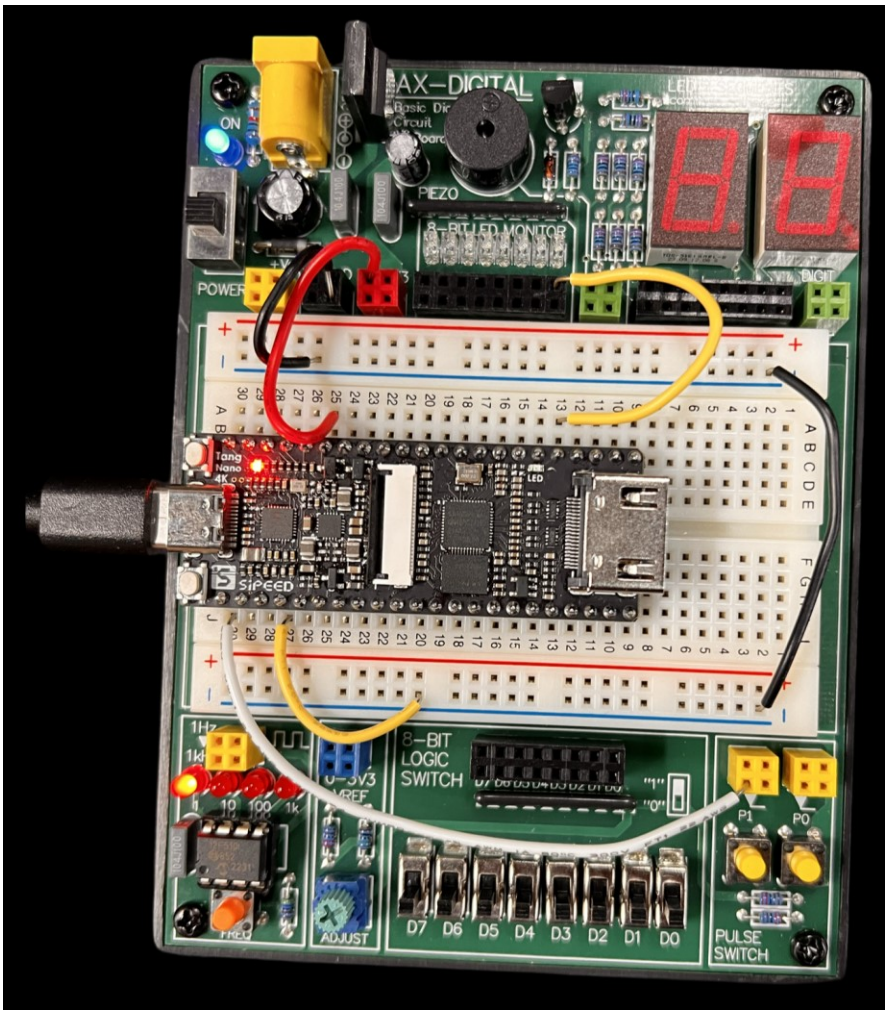


รูปที่ 13

4. โปรแกรมชิป FPGA

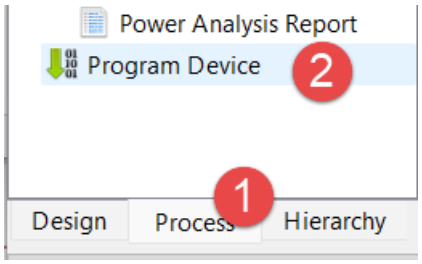
ในขั้นตอน Program Device ซอฟต์แวร์จะทำการสร้างไฟล์ bitstream (ไฟล์ .fs) ใช้โปรแกรม Programmer ในการโปรแกรมไฟล์ bitstream ลงในชิป GW1NSR-LV1QN48C6/I5 ผ่านสาย USB-JTAG โดยสามารถเลือกได้ว่าจะโปรแกรมไฟล์ที่หน่วยความจำแบบ flash หรือหน่วยความจำ SRAM

ทำการเชื่อมต่อสาย USB ระหว่างคอมพิวเตอร์กับบอร์ด Tang Nano 4K เพื่อจ่ายไฟเลี้ยงบอร์ดและเตรียมการสำหรับการโปรแกรมเอฟพีเอ



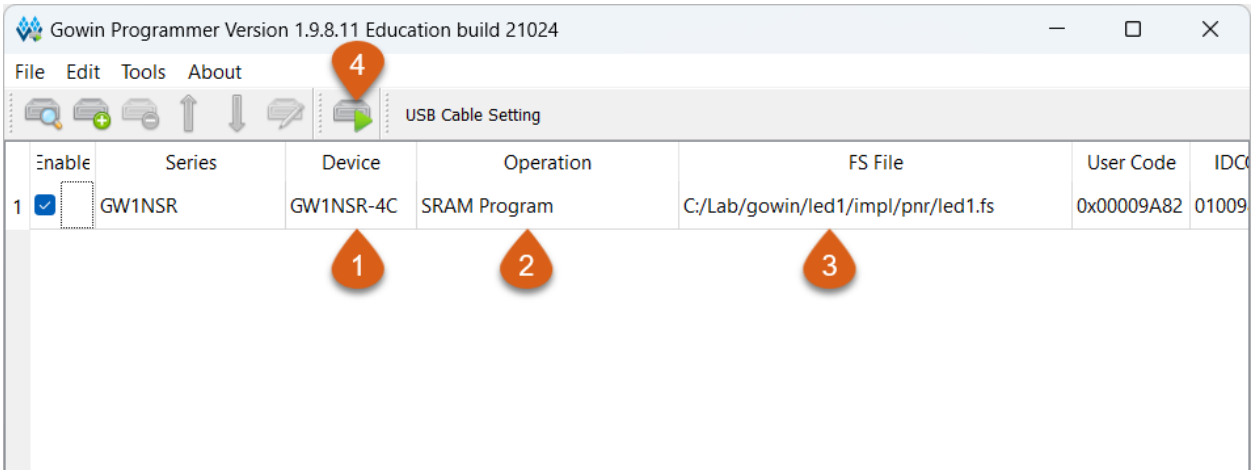
รูปที่ 14

รัน Program Device



รูปที่ 15

จะปรากฏหน้าจอโปรแกรม Gowin Programmer ซึ่งจะทำการสแกนหา Device 1 เราสามารถเซ็ตโหมดของการโปรแกรม 2 เลือกไฟล์ที่ต้องการโปรแกรมที่ 3 แล้วคลิก 4 เพื่อโปรแกรมเอฟพีจีเอ



รูปที่ 16

5. การทดสอบวงจร

ทำการต่อหลอด LED ภายนอกเข้ากับพิน led ของชิป FPGA จ่ายไฟให้ชิป FPGA ทำการโปรแกรมชิป เมื่อการโปรแกรมสิ้นสุดลง จะสังเกตว่า LED มีการกะพริบ

สรุป

บทความนี้ นำเสนอขั้นตอนการออกแบบวงจร FPGA GOWIN คร่าวๆ ผู้ใช้ควรศึกษาเอกสารประกอบซอฟต์แวร์ EDA ของ GOWIN เพิ่มเติม ข้อดีของเอฟพีจีเอของ GOWIN คือมีราคาที่ประหยัดมาก ทำให้ผู้สนใจศึกษาและทดลองการออกแบบวงจรดิจิตัลด้วยเอฟพีจีเอได้ง่ายขึ้น แต่อาจจะมีจุดอ่อนในส่วนของซอฟต์แวร์ที่ยังต้องการพัฒนาให้รวมเอาส่วนของการจำลองวงจร (simulation) เพิ่มเติม จะทำให้การพัฒนาออกแบบวงจรได้สะดวกยิ่งขึ้น

แหล่งข้อมูล

- เว็บไซต์ GOWIN EDA: https://www.gowinsemi.com/ ดาวน์โหลดโปรแกรมเวอร์ชั่นล่าสุด Gowin V1.9.x.x
- ข้อมูลบอร์ด SiPEED Tank Nano 4k: https://wiki.sipeed.com/hardware/en/tang/Tang-Nano-4K/Nano-4K.html
- ตัวอย่างโปรเจ็กต์ : https://wiki.sipeed.com/hardware/en/tang/Tang-Nano-Doc/examples.html
- ชำนาญ ปัญญาใส และวัชรากร หนูทอง (2547) ภาษา VHDL สำหรับการออกแบบวงจรดิจิตอล กรุงเทพฯ: ซีเอ็ดยูเคชั่น

เพิ่มเติม

GOWIN นำเสนอบอร์ดพัฒนา FPGA ราคาประหยัด ผู้ใช้สามารถใช้บอร์ดพัฒนาเพื่อทดสอบวงจร FPGA ก่อนนำไปใช้งานจริง

ตัวอย่างบอร์ดพัฒนา

- Tang Nano 4K: Xkit-FPGA ของบริษัท INEX
- GW1N-LV1Q-EVB:

ตัวอย่างโค้ดวงจรไฟกระพริบ โดยเขียนด้วยภาษา Verilog

ที่มา: https://github.com/sipeed/TangNano-4K-example/blob/main/led_test/project/src/led_test.v

```
module led (
    input sys_clk,
    input sys_rst_n,        // reset input
    output   reg led        //R
);
reg [23:0] counter;
always @(posedge sys_clk or negedge sys_rst_n) begin
    if (!sys_rst_n)
        counter <= 24'd0;
    else if (counter < 24'd1350_0000)       // 0.5s delay
        counter <= counter + 1;
    else
        counter <= 24'd0;
end

always @(posedge sys_clk or negedge sys_rst_n) begin
    if (!sys_rst_n)
        led <= 1'b1;
    else if (counter == 24'd1350_0000)       // 0.5s delay
        led <= ~led;                        // TogglePin
end

endmodule
```

```v
module led (
    input sys_clk,
    input sys_rst_n,        // reset input
    output   reg led        //R
);
 reg [23:0] counter;
always @(posedge sys_clk or negedge sys_rst_n) begin
    if (!sys_rst_n)
        counter <= 24'd0;
    else if (counter < 24'd1350_0000)       // 0.5s delay
        counter <= counter + 1;
    else
        counter <= 24'd0;
end

always @(posedge sys_clk or negedge sys_rst_n) begin
    if (!sys_rst_n)
        led <= 1'b1;
    else if (counter == 24'd1350_0000)       // 0.5s delay
        led <= ~led;                         // TogglePin
end

endmodule
```

```
//Copyright (C)2014-2023 Gowin Semiconductor Corporation.
//All rights reserved.
//File Title: Physical Constraints file
//GOWIN Version: 1.9.8.11 Education
//Part Number: GW1NSR-LV4CQN48PC6/I5
//Device: GW1NSR-4C
//Created Time: Mon 04 29 11:33:49 2024

IO_LOC "led" 10;
IO_PORT "led" PULL_MODE=NONE DRIVE=8 BANK_VCCIO=1.8;
IO_LOC "sys_rst_n" 15;
IO_PORT "sys_rst_n" PULL_MODE=UP BANK_VCCIO=1.8;
IO_LOC "sys_clk" 45;
IO_PORT "sys_clk" PULL_MODE=UP BANK_VCCIO=1.8;

```