FPGA GoWin čip GW1N-LV1QN48C6/I5 za hubičku

Ako je známe, mám rád lacné veci a rôzne kity, ktoré po krátkom hraní končia ako lapače prachu na poličke.
Preto ma potešilo objavenie veľmi lacnej dosky s osadeným FPGA čipom TangNano.

Tang Nano vyrába Seeed studio a je dostupná za smiešnych 4.90 USD priamo od Seeed studia, alebo dá sa nájsť aj u rýchleho Aliho.

Samozrejme, že som si ju objednal. Už hodne dávno.

Táto doska doteraz iba lapala prach a tak bol najvyšší čas pripojiť ju a zistiť či vôbec funguje.

Po pripojení k USB, sa v boarde spustí demo aplikácia - blikanie RGB LEDkou (zelená - modrá - červená) v pol sekundových intervaloch.
V operačnom systéme pribude:
USB Serial Converter A a USB Serial Converter B (0403:6010)

Sťahujem

- Gowin_V1.9.7.03Beta_win.zip (verzia pre Windows / verzia pre Linux)

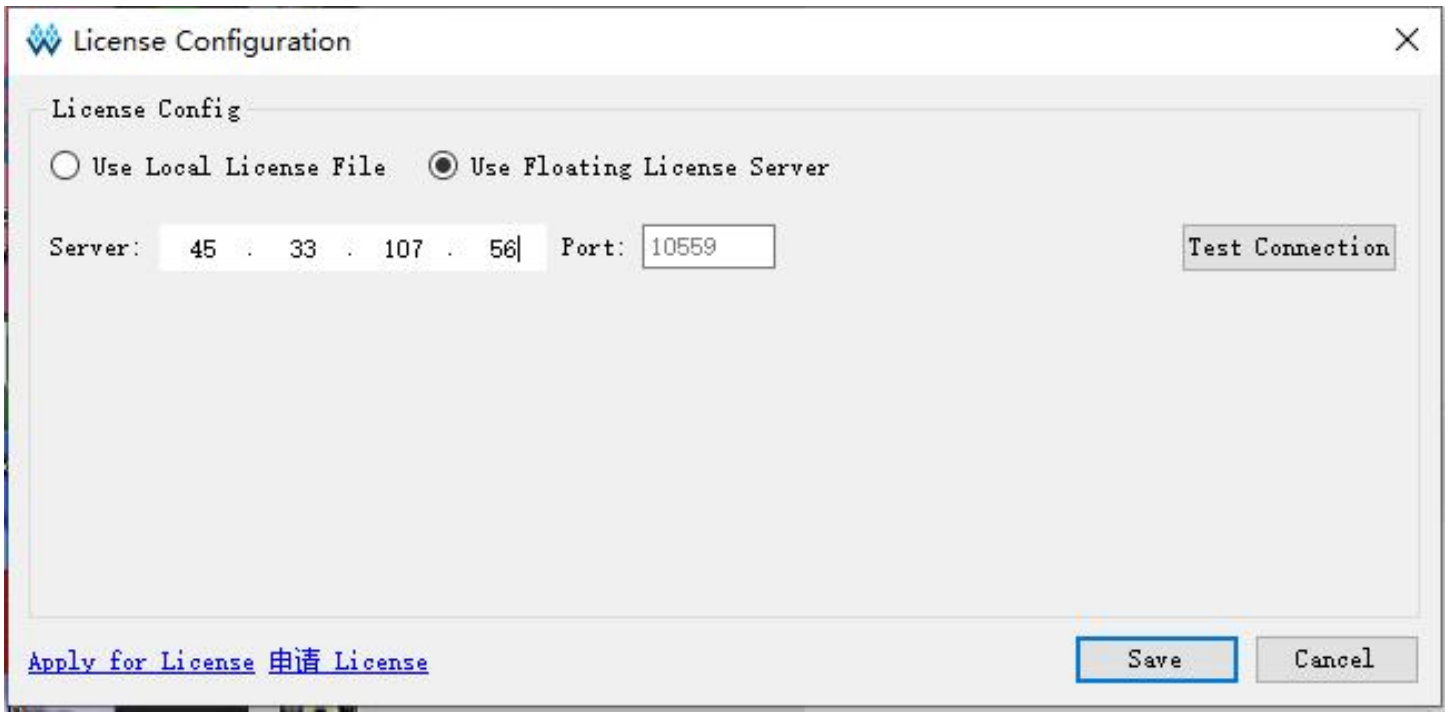
- example projekt RGB LED blink

Inštalujem:

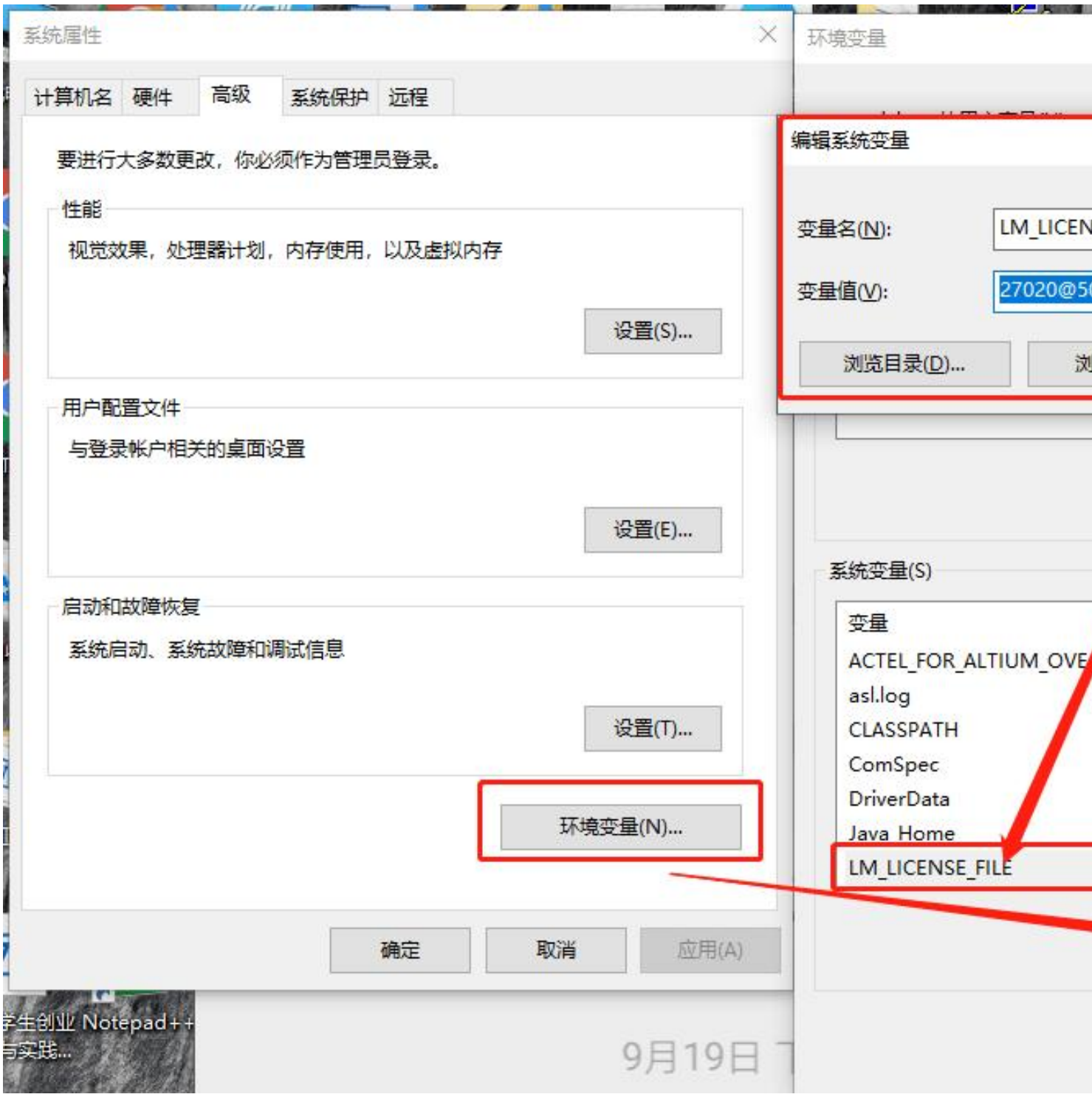
- Gowin_V1.9.7.03Beta_win.exe - obsahuje všetko potrebné pre tvorbu, syntézu a nakoniec aj naprogramovanie funkcie do súčiastky.

- rozbalujem example projekt do svojich dokumentov (stačí iba podadresár example_led)

Spúšťam aplikáciu GoWin.
Po výzvze o zadanie licenčného súboru mám dve možnosti:
Buď požiadam emailom o vydanie bezplatnej licencie pre lokálny počítač, alebo použijem online licenciu z internetu.
Pre prvotné testovanie teda použijem internetovú licenciu:



Do Envirtonment Variables pre Win je treba doplniť položku "LM_LICENSE_FILE" (potrebné pre syntézu SynplifyPro):



OK, licencia je v poriadku a znova spúšťam GoWin IDE.

Po spustení vyberám stiahnutý example projekt "example_led" (GoWinIDE Menu -> File -> Open -> led_prj.gprj)
Po načítaní projektu, v ľavej časti pracovnej plochy vyberám okno "Process", v ktorom je volanie jednotlivých funkcií IDE.
poznámka: Tieto volania na rozdiel od volaní cez menu IDE, majú výhodu, že všetky parametre si preberajú z aktuálneho projektu. Volania z menu "Tools" sú všeobecné, a treba parametre nastavovať samostatne.
Ďalej popisované volania akcií realizujem práve cez toto okno "Process".

Po otvorení projektu pre istotu skontrolujem typ FPGA, či je naozaj zvolený čip použitý na doske Tang Nano:GW1N-LV1QN48C6/I5 (GoWinIDE Menu -> Project -> SetDevice) a preverím, či je naozaj vybratý GW1N-LV1QN48C6/I5

Ako sa súčiastka bude v zapojení správať, je popísané v zdrojovom súbore led.v (okno Design->Led_prj->Verilog Files-> src\led.v)

```
module led (
    input sys_clk,
    input sys_rst_n,
    output reg [2:0] led // 110 B, 101 R, 011 G
);

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'd1200_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 <= 3'b110;
```

```
21.
        else if (counter ==             24'd1200_0000)      // 0.5s delay

22.
          led[2:0] <= {led[1:0],led[2]};

23.
       else

24.
          led <= led;

25.
    end

26.


27.
    endmodule
```

K popísaniu správania FPGA súčiastky je možné použiť buď schému, alebo popísanie jazykom HDL: najbežnejšie sú VHDL a Verilog.
V tomto prípade je použitý VERILOG.
Základnou jednotkou návrhu je modul. Každý program vo Verilog-u obsahuje definíciu portov, popis I/O, deklaráciu interných signálov a definíciu funkcií.
V tomto príklade modulu led, máme dva vstupné signály: sys_clk a sys_rst_n a tri výstupné signály LED[0] - LED[2].
Interný signál je 24 bitové počítadlo: counter[23:0]
Následne sú definované dve funkcie, ktoré sa vykonajú pri nábežnej hrane sys_clk signálu (platný vstupný clock impulz),
alebo pri dobežnej hrane sys_rst signálu (akivácia reset signálu).
V prvej funkcii sa inkrementuje počítadlo, prípadne sa nuluje a v druhej sa podľa stavu počítadla prepínajú výstupy LED (zelená, červená, modrá).

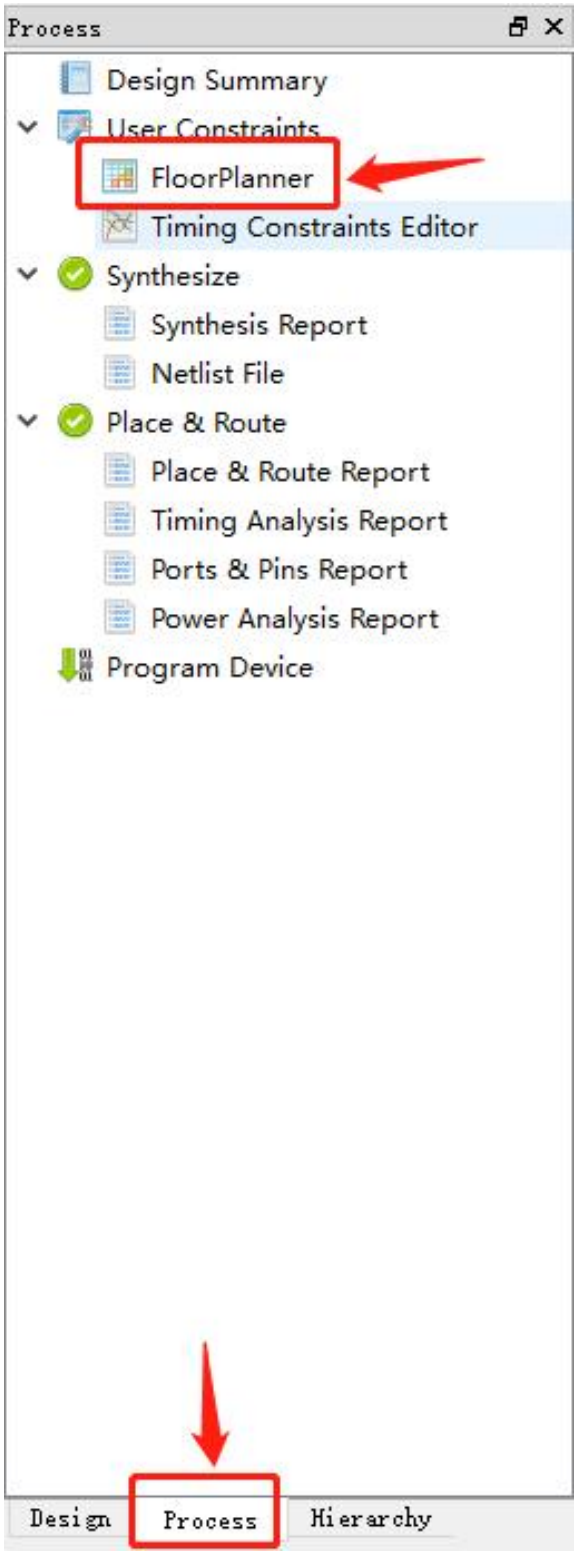
Pre začiatok nebudem nič meniť, len vyskúšam postupnosť krokov potrebných k definovaniu správania súčiastky v zapojení na doske TangNano.

Každé spracovanie prebieha v princípe v troch fázach :
1. Mapovanie - pridelenie signálov fyzickým vývodom zvolenej súčiastky a nasvaenie ich HW vlastností.
2. Syntéza - postup pri ktorom aplikácia hľadá hardwérové alternatívy k popisu funkcie zo zdrojového súboru.
3. Place and Route - vyhľadanie najvhodnejšieho usporiadania a prepojenia jednotlivých komponentov v matici daného FPGA.

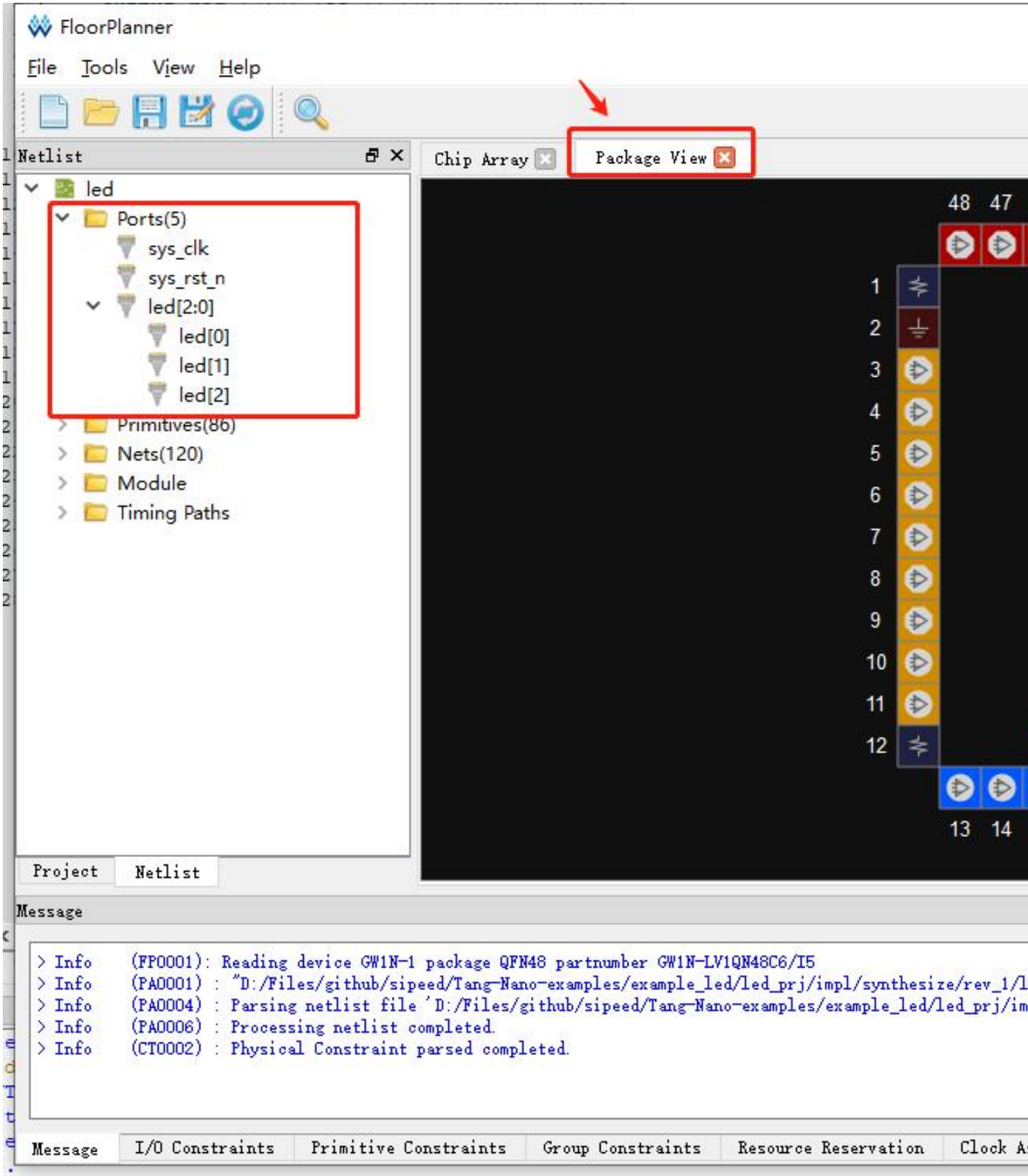
V GoWin IDE tieto kroky pekne vidno práve v process okne:
- User Constraints
- Synthesize
- Place & Route
- Programming device (super, že netreba spúšťať externú aplikáciu)

V User Constraints je zmienený mapper - "FloorPlanner". Okrem toho aj "Timing Constraints Editor", ktorý ale v tomto momente nie je treba.

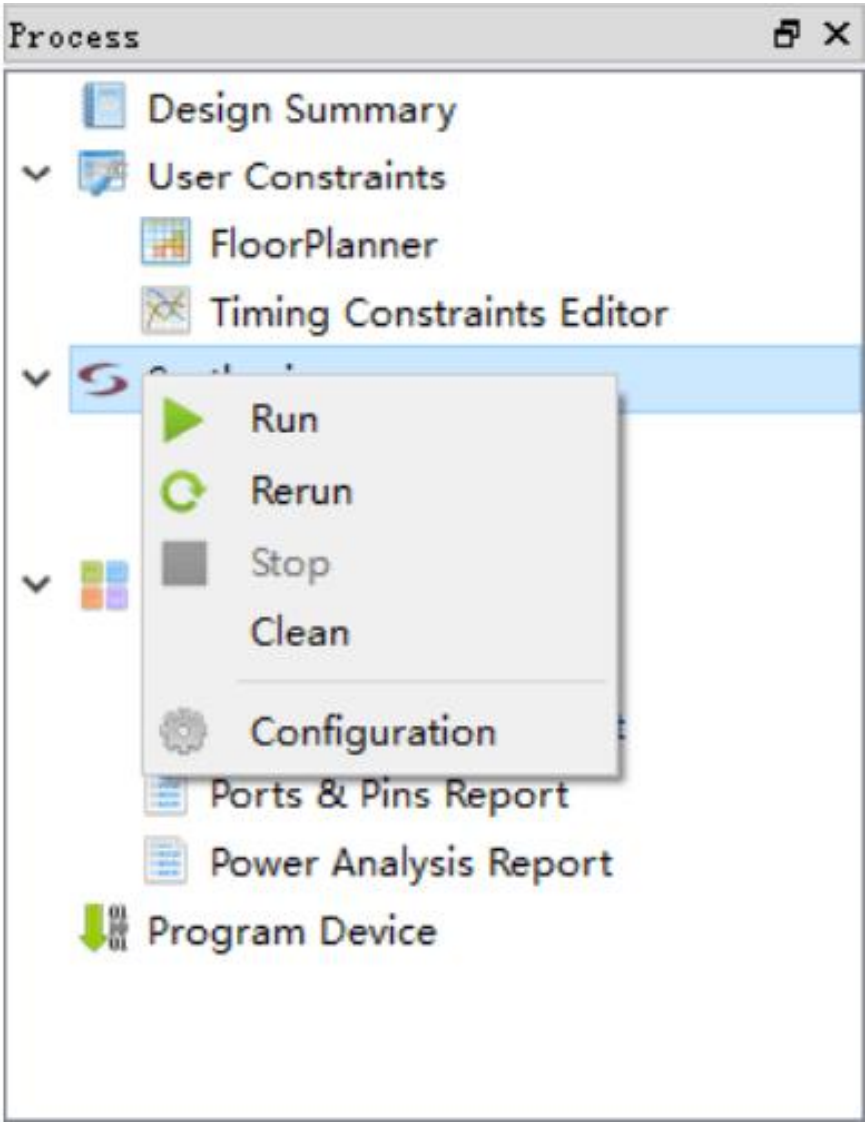
V projekte už sú síce namapované signály, ale môžem sa pozrieť ako.



Vo FloorPlanneri v package view následne uvidím fyzické pridelenie signálov k vývodom súčiastky.



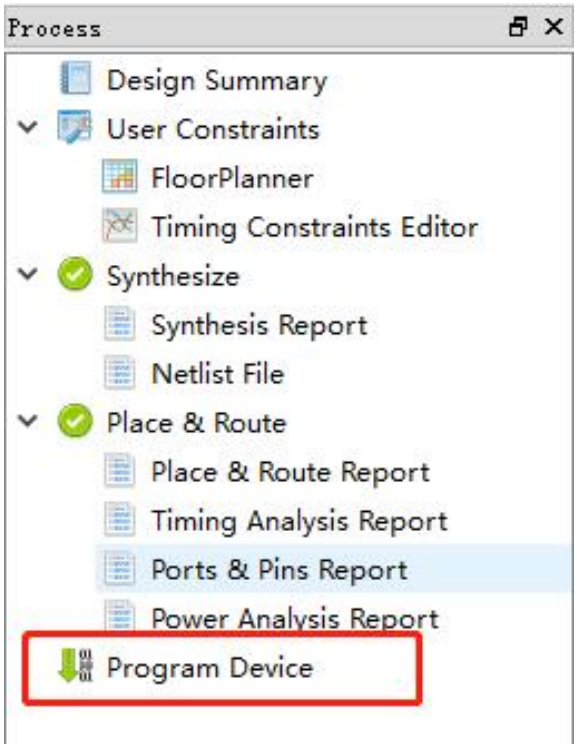
Následne spúšťam syntézu (pre zmenu - pravé myšie tlačítko a voľba a RUN).
Pokiaľ od poslednej syntézy nedošlo k zmene v projekte, výsledkom bude hlásenie "Nothing is executed for the Synthesize process". Ak silou-mocou chcem spustiť syntézu,
zvolím "Clean&Rerun All", kedy sa odstránia predchádzajúce výstupy a syntéza sa spustí znova.



Výsledkom syntézy je súbor typu „netlist“ pre ďalšiu prácu a report súbor pre užívateľa.
V netliste (textová forma) je vidno, ako aplikácia poprepájala jednotlivé LUT aby dosiahla to čo sme popísali a aké externé a interné signály na to použila.

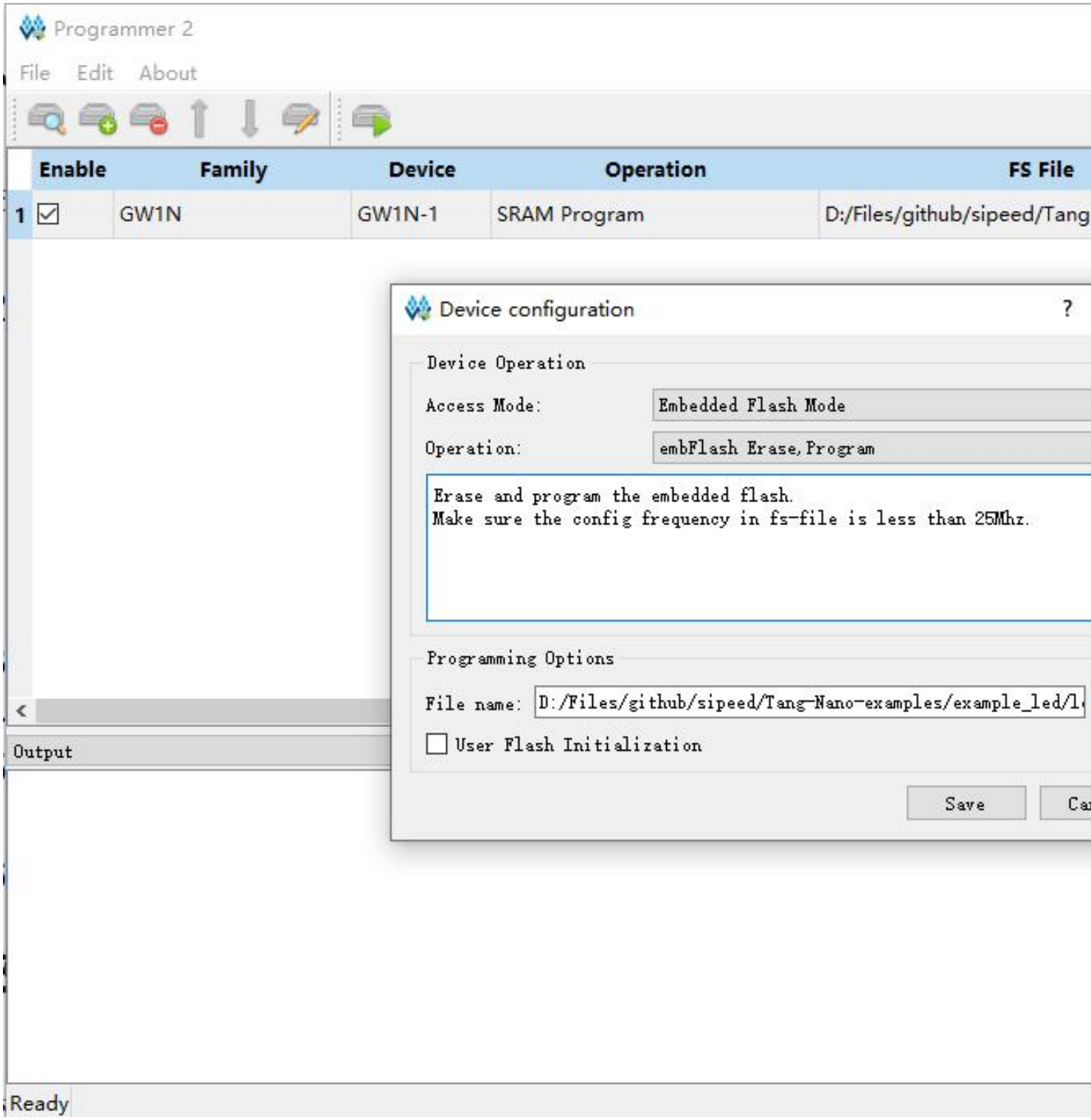
Spustením Place&Route sa tieto prepojenia aplikujú na fyzicky zvolený FPGA čip, kde sa aj dozviem ako je FPGA obsadené.
V tomto prípade bolo využité 4% jeho kapacity - t.j. 57 LUT z celkových 1152 dostupných.
Výstupom je súbor led_prj.fs, ktorý pomocou programátora fyzicky nahrám do súčiastky.
btw, v GoWin IDE stačí spustiť iba Place&Route - aplikácia si všetky potrebné a neuskutočnené kroky zavolá sama.

Posledným krokom je teda už len nahranie programu do cieľovej súčiastky.
Spúšťam programátor (Program Device).



Tu ako prvé vyberám typ kábla (GoWinProgrammer Menu -> Edit -> Cable Settings -> Cable).
V ponuke kliknem na "Query" nech si nájde sám (TangNano kit musí byť samozrejme pripojený do USB).
Na moje prekvapenie, aplikácia našla paralelný port LPT1...
Ikonkou pod menu je lupa - "Scan Device" - klikám teda na ňu, nech si aplikácia zakomunikuje cez kábel s čipom.
Výsledok - No devices found! Logické, na LPT nemám pripojené nič. Vraciam sa teda späť k nastaveniu kábla a vyberám manuálne:
USB Debugger A, port: USB Debugger A/1/Tang a frekvencia 15MHz. Save a opäť scan devices.
Úspech, aplikácia našla dokonca 2 čipy (neviem prečo) ale dala na výber s ktorým plánujem pracovať.
Keďže viem, že na TangNano doske je osadený GW1N-LV1QN48C6/I5, vyberám teda GW1N-1
V hlavnom okne programátora vidím zvolený čip a typ operácie, ktorú plánujem previesť.
Teraz zavriem okno s programátorom, nech si aplikácia nastavenie uloží a spúšťam ju znova (pri projekte to stačí spraviť iba jeden krát).
Teraz v hlavnom okne programátora vidím zvolený čip a typ operácie, ktorú plánujem previesť a aj zvolený FS súbor, kontrolnú sumu atď..

Podľa manuálu k tomuto blikaciemu príkladu, treba zmeniť typ pamäte, do ktorej fs súbor nahrávam, tak idem podľa návodu:
Následne klikám na Program/Configure (GoWinProgrammer Menu -> Edit -> Configure Device), kde zmením Access mode na "Embedded Flash Mode" a operáciu na "embFlash Erase, Program".



Ak by som nechal fs súbor nahrať iba do RAM, správanie FPGA by sa síce zmenilo no len do vypnutia napájania.
Pre trvalý zápis je preto treba zvoliť Flash pamäť.

Následne klikám na Program/Configure (GoWinProgrammer Menu -> Edit -> Program/Configure) a sledujem či sa zmení správanie FPGA čipu, resp celej Tang Nano dosky.
Jasné že nie, nahral som tam predsa to čo tam bolo - ale úspešne!
No týmto postupom som si vyskúšal celý reťazec potrebný pre nastavenie chovania konkrétneho FPGA čipu.

Teraz mám otvorené možnosti na "hranie sa s konkrétnym čipom".
Ako prvé, skúšam zmeniť interval blikania...

Odkazy:

- SiPEED LED Example popis

- GoWin FPGA GW1N-LV1 datasheet

- SiPEED Tang nano datasheet

- SiPEED Tang Nano home page

- SiPEED Tang Nano dokumentácia

- Úvod do VHDL

- Úvod do VHDL a FPGA (prezentácia)

- Verilog Tutorial