Gowin FPGA Design

User Guide

SUG113-1.2E, 06/07/2022

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
07/26/2018	1.1E	Initial version published.
06/07/2022	1.2E	• Supported Products removed. • The description of Resource Sharing removed. • The description of Full Case and Parallel Case specification removed. • The description of Section "3.1.1 Design Constraint Flow" updated. • The description of Section "3.2 Pinout" updated. • The descpition of Chapter "4 Timing Closure" updated.

Contents

List of Figures

List of Table

1 About This Guide

1.1 Purpose

This manual provides the descriptions of coding guidelines, design planning, and timing closure, and these factors directly determine the success of FPGA design.

Coding style directly affects the implementation of the FPGA design and ultimately the performance of the design. Although the synthesis tool integrates a series of optimization algorithms, it is still necessary for you to follow a certain coding style to guide the synthesis tool to achieve optimal results on a specific FPGA architecture.

Design planning helps you target the design to the chosen FPGA device and balance the associated area and speed requirements in a manner that takes full advantage of the functions and features supported by Gowin devices.

Timing closure can ensure that a user design meets a specific timing requirement. This chapter describes timing requirements, timing constraints, and timing optimization.

1.2 Related Documents

The latest user guides are available on the GOWINSEMI Website. You can find the related documents at www.gowinsemi.com:

- SUG100, Gowin Software User Guide
- SUG940, Gowin Design Timing Constraints Guide
- SUG935, Gowin Design Physical Constraints Guide

1.3 Terminology and Abbreviations

The abbreviations and terminology used in this manual are outlined in the table below.

Table 1-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language

Terminology and Abbreviations	Meaning
FSM	Finite State Machine
DSP	Digital Signal Processing
BSRAM	Block Static Random Access Memory
SSRAM	Shadow Static Random Access Memory
RTL	Register Transfer Level
CST	Physical constraint
SDC	Synopsys Design Constraint
CFU	Configurable Function Unit
CLS	Configurable Logic Section

1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Guideline on HDL Coding

2.1 General Requirements of HDL Coding

2.1.1 Coding for Hierarchical Synthesis

Complex system designs require a hierarchical approach as opposed to the use of single modules. A hierarchical coded design can be synthesized as a whole or have each module synthesized separately. When the design is synthesized as a whole, it can be synthesized as either a flat module or multiple hierarchical modules.

Each methodology has its associated advantages and disadvantages. Hierarchical coding is preferable for complex system designs because designs in smaller blocks are easier to keep track of and reduce the design period by allowing the reuse of design modules. Here are some tips for building hierarchical structures:

1. The top level should only be used to call the submodule. It is better to implement control logic in submodules.
2. Any I/O instantiations should be at the top level of the hierarchy.
3. All input, output, and bidirectional pins instantiations should be at the top level of the hierarchy;
4. The tristate statement for bidirectional pins should be written at the top module.
5. Ensure that the output signals of all modules use registers. The advantages are as follows:
   - Combinational logic and registers are in one module, which helps to overcome the lack of no cross-module synthesis.
   - The related logics placed in one module can realize resource sharing and key paths optimization.
   - Divide irrelevant logics into different modules, then you can use different optimization strategies, such as speed first or area first.

2.1.2 Pipeline Design Requirements

Pipeline design can improve design performance by restructuring a

long data path into several levels of logic and breaking it up over multiple clock cycles. Pipeline structure is an effective way to improve data path speed; however, special care must be taken due to the additional clock cycle latency of data path.

2.1.3 Comparing If-Then-Else and Case Statements

The if-then-else statement generates priority-encoded logic, whereas the case statement implements balanced logic, and it is recommended that if statement be nested less than 5 levels.

If-then-else statements can contain a set of different expressions, but a case statement can contain only one expression. If-then-else statements and case statements are equivalent if the conditions are mutually exclusive.

2.1.4 Avoiding Unintentional Latches

FPGA users should avoid using latches. The synthesis tools can build them out with feedback loops, which will increase the design area and result in performance degradation and problems with static timing analysis by introducing combinatorial feedback loops.

The Synthesis tool infers latches from incomplete conditional expressions; for example, an if-then-else statement without an else clause or a case statement without a default clause. To avoid unintentional latches, specify all conditions explicitly.

If the client uses the case statement and does not care about the output value of the default condition, /* synthesis full_case*/ constraint can be added to avoid the generation of the latch.

2.1.5 Global Reset and Local Reset

A global set/reset (GSR) network is built into Gowin devices. There is a direct connection to the core logic. It can be used as an asynchronous/synchronous set or asynchronous/synchronous reset. The registers in CFU and I/O can be individually configured to use the GSR. The global reset resource provides a convenient mechanism by which design components can be reset without using any general routing resources. Local reset usually has smaller fan-out. It is recommended to use common routing as a reset signal.

2.1.6 Clock Enable

Gating clocks is not encouraged in FPGA designs because it can cause timing issues, such as unexpected clock skews. The CLS structure in Gowin devices contains dedicated clock enable signals. You can use clock enable as a better alternative to achieve the same functions without worrying about timing issues. You should take the following into considerations when using the clock enable function of Gowin devices.

1. Clock enable is only supported by flip-flops, not latches.
2. Each CLS in the same CFU shares one clock enable signal.
3. All flip-flops have a positive clock enable input.

2.1.7 Multiplexer

The flexible configurations of LUTs within CLS can realize 2-to-1, 3-to-1, 4-to-1, or 5-to-1 multiplexers, etc. You can create larger multiplexers by programming multiple four-input LUTs.

2.1.8 Bidirectional Buffers

Using bidirectional buffers allows for fewer device pins, controlling output enable, and reducing power consumption. You can disable automatic I/O insertion in your synthesis tool and then manually instantiate the I/O pads for specific pins as needed.

2.1.9 Cross Clock Domains

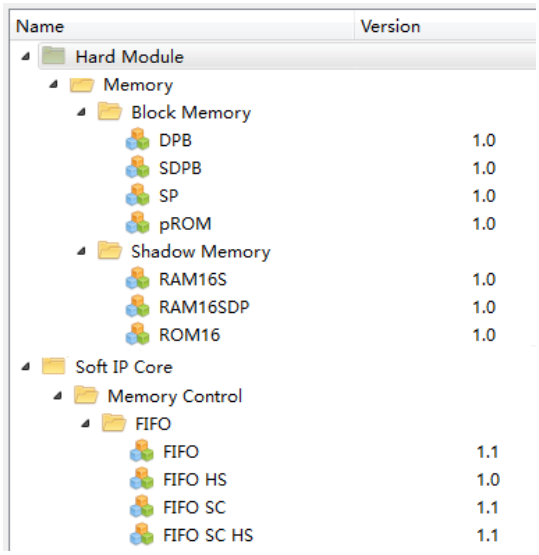
When passing data from one clock domain to another, special care must be taken to ensure that metastability issues do not arise as a result of set-up and hold timing violations. For single-bit signals, it is suggested to use a two/three-stage register structure to eliminate metastability. For multi-bit signals, it is suggested to use asynchronous FIFOs.

2.1.10 Memory Coding

Although RAM behavior description is portable and the coding is straightforward, different coding may generate different synthesis results. For Gowin devices, it's recommended to use the IP Core Generator in Gowin Software to generate block memory, shadow memory, and FIFO.

Gowin devices support multiple memory structures, including dual-port RAM, single-port RAM, semi-dual RAM, read-only ROM, and synchronous/asynchronous FIFO, as shown in Figure 2-1.

Figure 2-1 IP Core Generator Memory

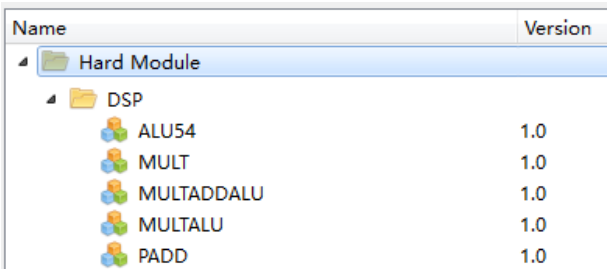


2.1.11 DSP Coding

Although DSP behavior description is portable and the coding is straightforward, different coding may generate different synthesis results. For Gowin devices, it's recommended to use the IP Core Generator in Gowin Software to generate DSP.

Gowin devices support multiple DSP structures, including ALU54, MULT, MULTALU, MULTADDALU, and PADD, as shown in Figure 2-2.

Figure 2-2 IP Core Generator DSP



2.2 Finite State Machine Requirements

A finite state machine advances from the current state to the next state at the clock edge. This section discusses the methods and strategies for state machine coding.

2.2.1 General Description

There are three ways to implement a finite state machine; the first is to process state-jump and state output simultaneously in one process; the second is to process state-jump in one process, state-jump law and state output in another process; the third is to process state-jump, state-jump law, and state output in three independent processes respectively. It is recommended to use the third. It is easier to be read and modified, and it will not cause extra delay for outputting the state directly without registering.

2.2.2 State Coding Methods for State Machines

There are several ways to code a state machine, including binary, one-hot encoding, and gray code. State machines with binary code and gray code have minimal numbers of flip-flops and wide combinatorial logics, whereas one-hot encoding is exactly the opposite.

The greatest advantage of one-hot encoding is that only one bit is required for state comparing. As a result, it decreases the decoding logic Although one-hot encoding uses more bits, i.e. more flip-flops for same states, its coding circuit saves an equivalent area to offset the area consumed by flip-flops.

For small state machines, less than five states, binary code and gray code are typically the defaults. For larger state machines more than five states, one-hot is the default.

2.2.3 Initialization and Default State Values for Safe State Machines

A safe state machine must be initialized to a valid state after power-up. You can initialize it during power-up or by including a reset operation to bring it to a known state. In the same manner, a state machine should have a default state to ensure that the state machine does not go into an invalid state. This could happen if all the possible combinations have been clearly defined in the design source code.

2.3 Gowin FPGA Hardware Features

2.3.1 I/O Logic

I/O logic supports Deserializer, Serializer, delay control, and byte alignment, mainly used for high-speed data transmission. I/O logic supports basic mode, SRD mode, and DDR mode, etc. You can use Gowin devices IO logic according to your design requirements.

2.3.2 DSP

LittleBee® and Arora families DSP support 9-bit/18-bit signed/unsigned multiplication, MAC, accumulation; and support 54-bit ALU, barrel shifter, pipeline and bypass registers.

2.3.3 BSRAM

LittleBee® and Arora families BSRAM can be configured up to 18Kbits, and both data bit width and depth can be configured. Each BSRAM has two independent ports; and they have independent clocks, addresses, data, and control signals, but they share the same storage memory. Each BSRAM has four operation modes: Single Port, Dual Port, Semi Dual Port, and Read-only, and it also supports pipeline and bypass registers.

2.3.4 SSRAM

LittleBee® and Arora families SSRAM can be configured as a single port random memory with depth of 16 and width of 1/2/4 bits, semi dual port random memory with a depth of 16 and width of 1/2/4 bits, and 16 bits x 1 read-only random access memory.

2.4 Lower Power Coding

Optimize an area to reduce logic usage and routing lengths, and then to reduce power. It is recommended to use the IP Core Generator in Gowin Software to call the basic cells in Gowin devices for the most power-efficient (least area and resources) implementation. Eliminate known glitches for power reasons, reduce I/O toggle rate, and enter into the sleep state to reduce system power.

2.5 Coding to Avoid Simulation/Synthesis Mismatches

Certain coding styles can lead to synthesis result that differs from simulations. This is caused by error information that is ignored by a simulator but can be detected by a synthesis tool; as such, Gowin coding style is recommended.

2.5.1 Sensitivity Lists

Sensitivity lists must contain all input and output signals to avoid mismatches between simulation and synthesized logic.

2.5.2 Blocking/Nonblocking Assignments

Use blocking assignments to generate combinational logic; use nonblocking assignments to generate sequential logic.

2.5.3 Signal Fan-out

Signal fan-out control is designed to maintain reasonable post-synthesis fan-outs. The synthesis tool reduces signal fan-out by duplicating circuits. Use syn_maxfan for specific signals to acquire better timing closure.

Gowin device architectures are designed to handle high fan-out clock signal with dedicated clock and handle high fan-out reset signal with dedicated global reset network. However, synthesis tools tend to replicate logic. This type of logic replication occupies more resources in the devices and makes performance checking more difficult. Use syn_maxfan flexibly based on actual conditions to avoid the generation of many logic replication.

3 Design Planning

FPGA design mainly includes the following two phases:

1. Before designing, it is necessary to define the function and architecture; choose a suitable FPGA device and then write RTL code.
2. Target your design to the chosen FPGA device and fully utilize the chosen device.

Each of the two phases described above affects the other. This chapter focuses on the second phase and explains how you can fully utilize the functions and features provided by Gowin devices.

3.1 Gowin Software Design Planning Flow

Design planning is not mandatory for all designs, but it will be beneficial to most designs, especially in the case of large designs that involve high resource utilization and/or a tight timing requirements. For these designs, design planning can help reduce potential placement and routing problems or timing issues, and it can increase the possibilities for design reuse and migration.

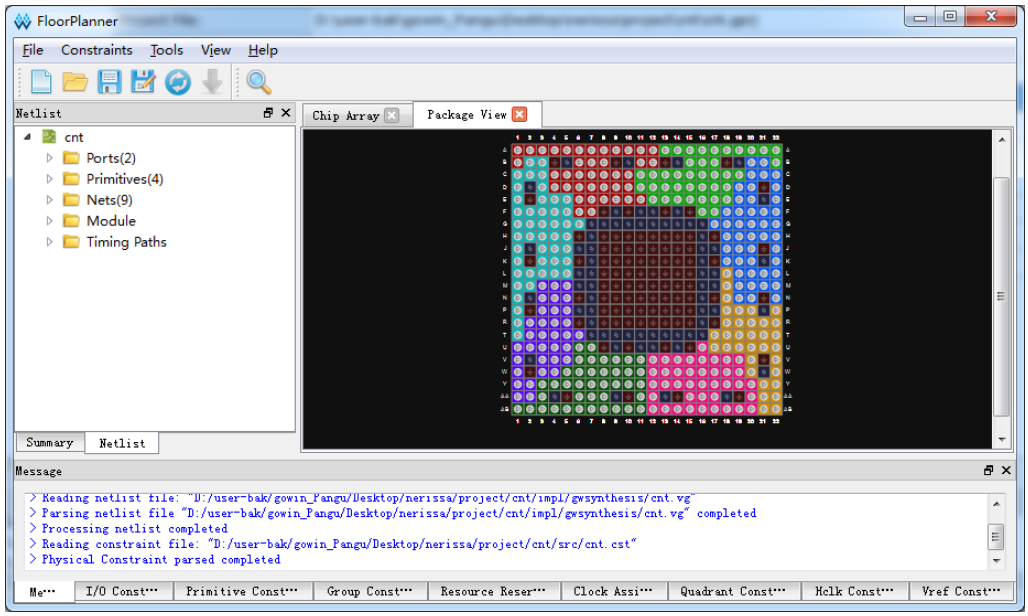
In Gowin Software, design planning starts when the synthesis has been completed successfully and before placement and routing. CST files contain all the design planning constraints required to guide backend placement and routing. If design planning is modified, i.e. CST files are modified, the design returns to the stage after synthesis before placement and routing. For CST details, you can see *SUG935, Gowin Design Physical Constraints User Guide*.

3.1.1 Design Constraint Flow

Gowin Software allows you to set constraints for post-synthesis ports, netlist, registers, and instances. The CST file is the main input for defining design planning.

Backend placement and routing software acquire users constraints by reading the CST files. If there is a syntax error or an illegal constraints error, the backend placement and routing software will exit directly. CST files can be text-edited or generated using FloorPlanner, as shown in Figure 3-1. For the details, you can see *SUG935, Gowin Design Physical Constraints User Guide*.

Figure 3-1 FloorPlanner Interface



3.1.2 Design Planning Tool

Gowin Software provides FloorPlanner for design planning, and its functions are as follows:

- View all the design elements that you can manage.
- View the hardware resources that are available for the chosen device.
- Assign specific design elements to specific FPGA resources.
- Support drag and drop to define constraints.
- Supports constraints legality check.
- Supports manual adjustment to optimize timing path.

3.2 Pinout

Pinout planning is the process of defining your FPGA I/O protocols and locations on the chosen device. It is based on your actual design and the chosen device.The pinout planning process involves the following steps:

1. Assign your design ports to specific I/O locations.
2. Define the I/O protocol and other I/O characteristics.
3. Check your assignments for legal usage.
4. Exchange your pin assignments with other parties according to PCB and circuit diagrams, if necessary.

3.2.1 Pinout Rules

- Distribute dedicated pins first, such as clock input, phase-locked loop input, DDR, etc.
- Distribute the common pins instead of specific pins to the specific BANK; this helps the backend routing tool to achieve an optimum result.

- Check if duplicated pins conflict with the device programming modes.

3.2.2 Pin Migration

For the device with the same package, you might want to migrate the design to a device with a larger capacity for further function extension or to a device with a smaller capacity to reduce cost. The pinout should stay the same or changes should be minimal to avoid PCB redesign. Gowin Software provides a pin migration feature. You can view incompatible pins in FloorPlanner. Use LOC_RESERVE to disable these pins.

3.3 Clock Assignment

3.3.1 Clock Assignment Rules

Clock frequency and clock fan-out are the main concerns when assigning clock resources assignment. The total number of available clock resources in Gowin FPGA devices is also a deciding factor.

In general, dedicated resources give better timing results because of the minimized relative time delay. The routing resources saved also ease routing congestions in highly congested designs. In rare cases, you might use the general routing resources as a clock. Because general routing will increase the time delay, it should only be used in the designs that are characterized by low speed and low fanout. Here are the general rules for clock resource assignment:

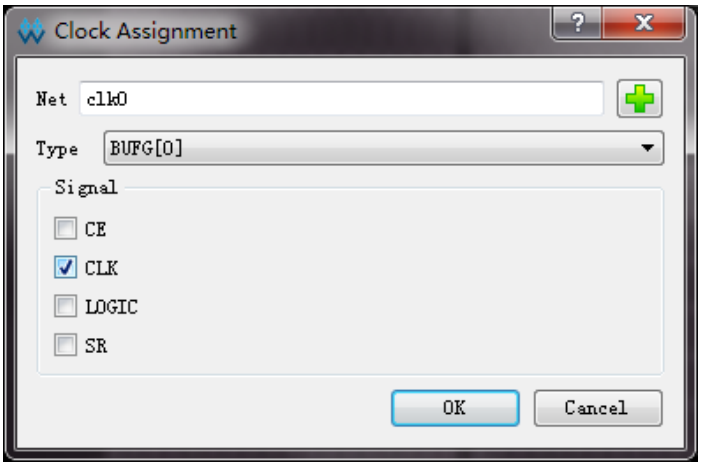
- Determine the clock numbers and the fanout for each clock in your design.
- Determine the clock resources provided by the target device.
- Determine the speed requirement for each clock.
- Assign the high-speed and high-fanout clock as the global clock.
- If the number of global clocks is less than the clock number in your design, use quadrant clocks.
- Use high-speed clock resources for high-speed interfaces with high fanout.

3.3.2 Clock Resources Assignment Constraint

NET_LOC "xxx" BUFG0 = CLK | CE | SR | LOGIC

Assign the clock signal to the dedicated global clock network; BUFG0~BUFG7 is the eight global clock supported by Gowin devices. CLK | CE | SR | LOGIC indicates the constraint object. CLK is clock; CE is clock enable; SR is synchronous reset, and LOGIC is a logic device, as shown in Figure 3-2.

Figure 3-2 Global Clock Constraints Interface



3.4 Logic Resource Constraint

3.4.1 Definition

The logic constraint is the logical partitioning of design elements, which results in physical placement or implementation changes of the design. Logic constraint is accomplished by specifying FPGA location preferences.

For Gowin devices, logic constraint can improve the design performance, especially for the designs with good structures. Logic constraint provides a combination of automation and user control for design reuse and modular, hierarchical, and incremental design flows.

3.4.2 Constraint Syntax

INS_LOC "cnt[5]" R2C2

Specify the specific object to the specific CFU location.

GROUP hh = { "cnt_Z[1]" "cnt_Z[2]" "cnt_Z[3]" "cnt_Z[4]" "cnt_Z[5]" "cnt_Z[6]" "cnt_Z[7]" }

GRP_LOC hh R[3:6]C[4:6]

Group specific objects and specify them to the precise regional location.

3.4.3 Constraint Strategies

- Define regions based on design hierarchy.
- Define regions based on critical paths.
- Define regions based on input/output signals with high fanout.
- Define logic modules and optimize modules individually using different enhancement strategies.

3.4.4 Special Considerations

- Block RAM can be placed alone. Do not group block RAMs.
- Larger logic groups need starting position and relative size.

- Do not group carry chains and bus.
- Do not group supplemental logic.

4 Timing Closure

Every design has to run at a certain speed based on the design requirement. There are generally three types of speed requirement in an FPGA design: timing, throughput, and latency. Throughput and latency are mutually exclusive. High throughput usually means more pipelining, which increases latency, while low latency usually requires longer combinatorial paths, which removes pipelines and can reduce the throughput. Therefore, there is a requirement to balance throughput and latency as a priority.

Timing closure is usually the key factor that impacts the ability of a design to run at a required speed. It can be very challenging to close timing using various techniques. This chapter focuses on timing closure, and explains how to close timing in your design.

4.1 Timing Closure Strategies in Synthesis

When using GowinSynthesis for synthesis, you need to follow some general rules to get better timing closure.

1. Obey Gowin RTL coding rules.
2. Use proper constraints to guide synthesis.
3. Use I/O pin registers to improve pin timing. Input pin register can be used to improve input setup time; output pin register can be used to improve clock to output time.
4. Use I/O delay unit to improve input hold time. Input pin register use may result in hold time issues because of short data channel delay. IODELAY can therefore be added to the data path to compensate for the input hold time; for the details of primitives, you can see section 4.4 in *UG289, Gowin Programmable IO User Guide*.
5. Adjust the relationship between IO input register data and clock using HCLK to meet both the setup time and hold time.
6. Use dedicated GSR resources, i.e. instantiate Gowin primitive GSR in RTL design. If your design contains high fanout reset and set signals, it is advisable that you use GSR resources to reduce routing congestion and enhance routing efficiency.
7. Reduce fanout to improve the main frequency. It is suggested to use syn_maxfan selectively to reduce fanouts at the expense of register

duplication.

8. Use one-hot state encoding. For high-speed design, it's advisable to use one-hot state eccoding. Also it will increase resource utilization rate and power consumption.
9. Resource reuse will increase logic levels and generate a huge delay path. The Synthesis tool usually reuses the non-key paths, but you also need to check the key paths to ensure that no timing issues are caused.
10. Check the synthesis report; read the post-synthesis timing information simply; analyze high fan-out path, key path, and large delay path.
11. Read the synthesis timing report carefully and thoroughly. As the synthesis timing report does not contain placement and routing information, the results outlined in the report are typically better than the actual results. The actual result is usually 1/3 to 1/2 less than the report result.

4.2 Timing Closure Strategies in Place & Route

When using Gowin Software, you need to follow some general rules to get better timing closure.

1. Add appropriate physical and timing constraints to guide place & route.
2. Pay attention to the warnings and errors displayed in the tool.
3. Select appropriate place & route options.
4. Avoid insufficient constraints, such as no clock constraints, asynchronous clock analysis, etc. Reasonable timing constraints can be added.
5. Avoid excessive constraints, such as higher clock frequency than the actual required constraints, multi-cycle paths for IO, etc. multicycle or maxdelay can be used to slacken the timing.
6. Use GCLK for clock enable whenever possible. Clock enable is usually a group of high fan-out signals for driving a large number of registers. If regular routing resources are used, which may results in a huge delay. It's advisable that you use GCLK resources.

4.2.1 Timing Constraint

The Timing Constraints Editor in Gowin Software supports common timing constraints. You can see appendix A in SUG940, *Gowin Timing Constraints User Guide* for timing constraints syntax. If no clock constraint is added, Gowin Software will analyze the design timing according to the default clock frequency.

1. The default clock for LittleBee® family is 50MHz.
2. The default clock for Arora family is 100MHz.
3. Setup timing is analyzed at high temperature and low pressure.
4. Hold timing is analyzed at low temperature and high pressure.

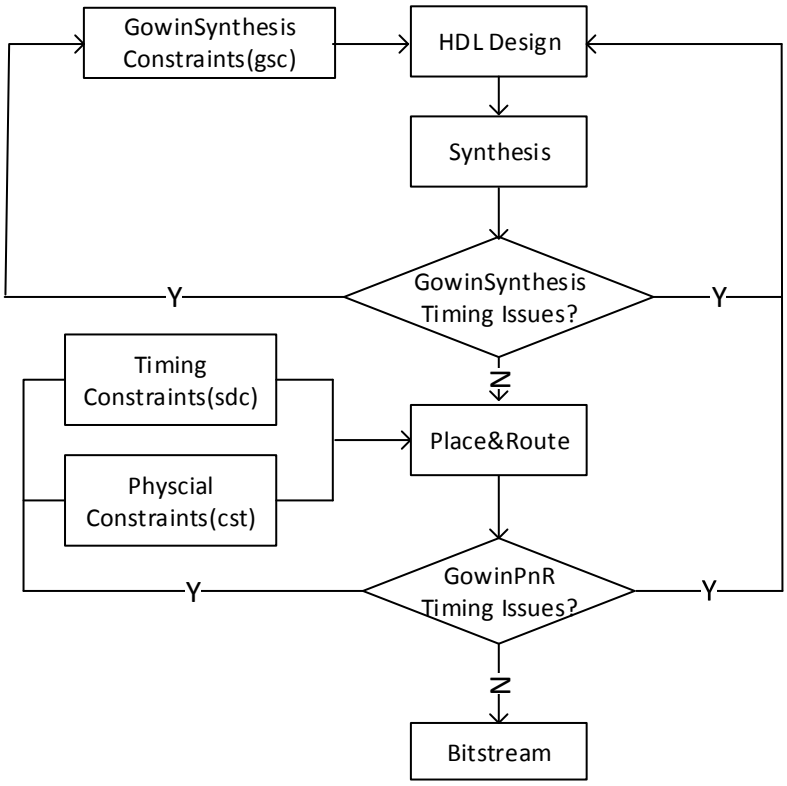
4.2.2 Option Settings

1. Run Timing Driven: Enable by default for timing driven routing; if timing requirements are not high, you can disable this option in order to save running time.
2. Place option: Place algorithm with values of 0 and 1, and the default value is 0.
   - If it is 0, the default place algorithm is used.
   - If it is 1, some time efficiency is sacrificed to try to find a better place based on algorithm 0.
3. Route Option: Route algorithm with values of 0, 1, and 2; and the default value is 0.
   - When it is 0, the default route algorithm is used, and the route is adjusted according to congestion.
   - When it is 1, the route is adjusted according to the timing.
   - When it is 2, the route speed will be relatively fast.
4. Place input register to IOB: Place registers driven by input Buffer to IOB to improve IO logic timing, and the default value is True.
5. Place output register to IOB: Place registers driven by output/tristate Buffer to IOB to improve IO logic timing, and the default value is True.
6. Place inout register to IOB: Place registers driven by in/out Buffer to IOB to improve IO logic timing, and the default value is True.

4.3 Solve Timing Issues

During the process of Gowin Software compilation, the timing analysis reports are output at the steps of synthesis and place & route respectively, and the timing requirements can be confirmed initially by the report. The flow of checking and solving timing issues is as shown in Figure 4-1.

Figure 4-1 Flow of Solving Timing Issues



4.3.1 Synthesis Timing Report Analysis

Before synthesis, it is a priority to check that the RTL conforms to Gowin coding guideline; for example, whether the DSP/BSRAM outputs go through registers; how the state machine is coded, etc.

After synthesis, read the timing analysis results in the synthesis report.

1. Check whether the maximum operating frequency meets the design frequency requirements by the "Max Frequency Summary", as shown in Figure 4-2.

Figure 4-2 Max. Frequency Summary

Max Frequency Summary:

No.	Clock Name	Constraint	Actual Fmax	Logic Level	Entity
1	clk	50.0(MHz)	136.9(MHz)	6	TOP
2	dclk	50.0(MHz)	144.8(MHz)	5	TOP
3	clk_01ms	50.0(MHz)	73.3(MHz)	9	TOP
4	clk_100ms	50.0(MHz)	85.4(MHz)	9	TOP
5	clk_game	50.0(MHz)	81.2(MHz)	9	TOP
6	clk_5ms	50.0(MHz)	53.0(MHz)	13	TOP
7	clk_1ms	50.0(MHz)	189.2(MHz)	4	TOP

2. Confirm the estimated timing of the worst path by "Detail Timing Paths Information".
   - If the number of logic levels on the worst path is large, registers need to be inserted to reduce the number of logic levels. For C6 speed chips of LittleBee® family, if run up to 100MHz, the number of logic levels needs to be controlled to less than 4; for C8 speed chips of Arora family, if run up to 100MHz, the number of logic levels needs to be controlled to less than 8.

- If the main delay on the worst path is caused by BSRAM, the BSRAM output can be delayed by one clock cycle.
- If the main delay on the worst path is caused by DSP, the DSP output can be delayed by one clock cycle.
- If the delay on the worst path is primarily caused by SSRAM, SSRAM can be converted to a shift register, using the synthesis attribute syn_srlstyle.

4.3.2 Place & Route Report Analysis

By analyzing the resource usage in place and route report, ensure that design resources are properly utilized. If there is excessive utilization, the RTL design needs to be updated, or synthesis attribute constraints need to be added.

Resource Usage Summary

Before analyzing the timing report, the place & route report needs to be reviewed to confirm that the resource utilization is not excessive. In general, when the CLS utilization reaches 85% or more, or BSRAM/DSP resource utilization exceeds 80%, which can be called an excessive design; and it can lead to timing closure issues, as shown in Figure 4-3.

Figure 4-3 Resource Usage Summary

Resource Usage Summary:

Resource	Usage	Utilization
Logic	7057/8640	81%
--LUT,ALU,ROM16	7009(1415 LUT, 5594 ALU, 0 ROM16)	-
--SSRAM(RAM16)	8	-
Register	5828/6867	84%
--Logic Register as Latch	0/6480	0%
--Logic Register as FF	5827/6480	89%
--I/O Register as Latch	0/387	0%
--I/O Register as FF	1/387	1%
CLS	3781/4320	87%
I/O Port	38	-
I/O Buf	38	-
--Input Buf	2	-
--Output Buf	36	-
--Inout Buf	0	-
IOLOGIC	0	0%
BSRAM	24 SDP	92%
DSP	18 MULT18X18	90%
PLL	1/2	50%
DCS	0/8	0%
DQCE	0/24	0%

Clock Usage Summary

In theory, the number of clocks in the design should not exceed that of clocks of the target device. Otherwise, some general routing resources are used as clocks, which may cause setup or hold violations. You can view the clock resource usage in the placement and routing report, as shown in Figure 4-4.

Figure 4-4 Global Clock Usage Summary

Global Clock Usage Summary:

Global Clock	Usage
PRIMARY	1/8(12%)
SECONDARY	1/8(12%)
GCLK_PIN	2/7(28%)
PLL	1/2(50%)
CLKDIV	0/8(0%)
DLLDLY	0/8(0%)

4.3.3 Place & Route Timing Report Analysis

Reduce Register Fan-out

Read the worst path of the "Setup Analysis Report" in the timing report, and check the number of fan-outs of the starting register of the timing path. If the fan-out is too large, you can duplicate registers to reduce the fan-out of registers, as shown in Figure 4-5.

Figure 4-5 Worst Timing Path

Data Arrival Path:

AT	DELAY	TYPE	RF	FANOUT	LOC	NODE
0.000	0.000					active clock edge time
0.000	0.000					clk_usbif
0.000	0.000	tCL	RR	1	IOB48[A]	usb_ifclk_ibuf/I
0.683	0.683	tINS	RR	7392	IOB48[A]	usb_ifclk_ibuf/O
0.926	0.243	tNET	RR	1	R28C62[0][A]	usb_rf32/mov_sig_src_o_s0/CLK
1.158	0.232	tC2Q	RF	21	R28C62[0][A]	usb_rf32/mov_sig_src_o_s0/Q
2.766	1.609	tNET	FF	1	R32C33[0][B]	n5914_s2/I2
3.219	0.453	tINS	FF	3	R32C33[0][B]	n5914_s2/F

In the above figure, the number of fan-outs of the starting register of the worst path is 21, and the net delay of 1.609ns occupies 42% of the path delay, which seriously affects the timing closure; in the RTL, add attributes to the net definition of this register, such as:

reg mov_sig_src_o_s0 /* synthesis syn_maxfan = 10 */

Therefore, the RTL should be designed according to the actual fan-out of the register, but it is not that the smaller the fan-out, the better, which may also lead to high fan-out of register source; and it can affect the timing closure.

Optimize BUFS Resource Usage

Generally speaking, the clock enable CE is a high fan-out net, and the BUFS routing resources are preferentially used in the placement and routing. If the delay on the worst timing path is mainly caused by CE using BUFS routing resources, you can add physical constraints to avoid using BUFS, such as:

CLOCK_LOC "ce_0" LOCAL_CLOCK


GOWIN
PROGRAMMING FOR THE FUTURE