1. Установка Gowin EDA, Icarus Verilog (iverilog), Verilog Virtual Processor (vvp), GTKWave. Симуляция тестового проекта

1. Установка программного пакета для симуляции

Для проведения временной симуляции Verilog-проектов используется следующий набор программного обеспечения:

- компилятор Icarus Verilog (iverilog), транслирующий Verilog-программы и тестбенчи в netlist-файлы
- среда исполнения Verilog Virtual Processor (vvp), симулирующая работу скомпилированного Icarus Verilog модуля
- визуализатор временных диаграмм GTKWave

Перечисленное программное обеспечение является открытым и может быть также загружено самостоятельно.

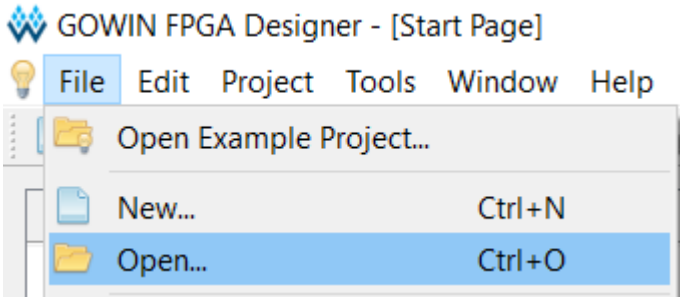
2. Установка Gowin EDA Education

Разработка для ПЛИС производства компании Gowin в рамках данного курса осуществляется в официальной среде Gowin EDA. Версия Education не требует лицензий для использования и незначительно ограничена в выборе устройств и IP-ядер.

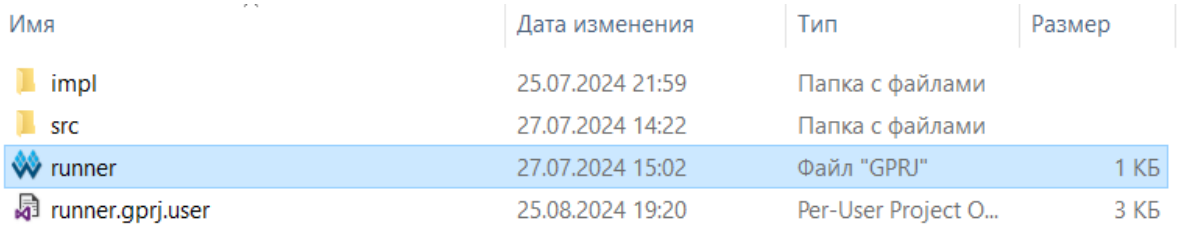
Для использования Gowin EDA требуется установка и настройка программы (установка драйверов, подготовка портов к использованию), может потребоваться перезагрузка системы для завершения настройки COM-портов.

3. Тестовый проект runner

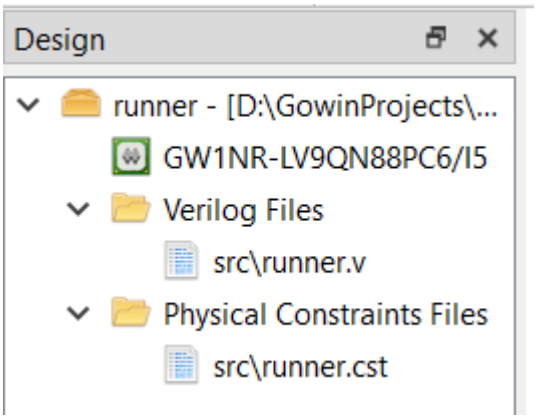
После открытия Gowin EDA вызвать меню открытия файлов.



Найти и открыть в диалоговом окне файл проекта “runner” (расширение “.gprj”).



При открытии проекта демонстрируется окно Design Summary с настройками проекта. В левой части программы перечислены входящие в проект файлы. В constraints-файле “src\runner.cst” перечислены привязки входов/выходов модуля к пинам ПЛИС.



Дальнейшая работа ведётся с файлами “src\runner.v” и “src\runner_tb.v”, последний из которых не включен в иерархию проекта, но доступен непосредственно в папке.

Первый из файлов содержит описание самого цифрового модуля. Этот модуль управляет светодиодами на плате, включая один за другим. На данный момент переключение происходит каждые 4.500.000 тактов, что соответствует ⅛ секунды при тактовой частоте 27 МГц. Это определено условным оператором на строке 21. Для симуляции необходимо уменьшить это число, например, до значения 5.

Второй файл описывает тестбенч, то есть тестовый стенд для симулируемого модуля. Тестбенч сам представляет из себя модуль, который должен включать в себя симулируемый, а также все необходимые входы и выходы для его подключения. Для работы данного тестбенча используются следующие операторы и команды:

- `timescale – определяет временную длительность задержек и точность их округления, если указаны дробные задержки
- initial – все команды в этом блоке выполняются в начале симуляции
- forever – команды после этого оператора выполняются на протяжении всей симуляции
- # – оператор задержки
- $finish – команда завершения симуляции
- $dumpfile – команда создания выходного файла симуляции
- $dumpvars – команда выбора данных для выгрузки в выходной файл, выбирается глубина (по иерархии) и модуль, относительно которого считается глубина

В подготовленном тестбенче тактирующий сигнал clk изначально устанавливается в значение 0, а сигнал сброса reset – в 1. Через 20 нс сигнал сброса устанавливается в значение 0, ещё через 20 нс – обратно в 1. Тактирующий сигнал переключается между 0 и 1 каждые 5 нс. Симуляция длится 1000 нс. Переменные тестбенча выгружаются в файл “./runner_out.vcd”, путь к файлу указывается относительно среды выполнения, в которой будет запущена симуляция. Важно явно указать все значения переменных модуля, иначе они примут неизвестное значение и симуляция будет некорректной.

4. Компиляция модуля runner и его тестбенча с помощью Icarus Verilog

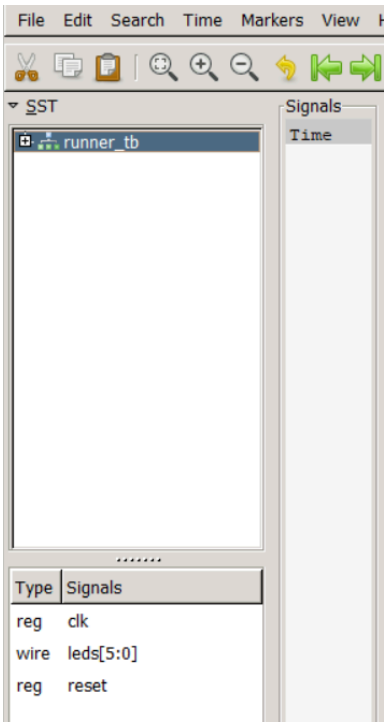
Запустить терминал в папке проекта (например, Windows Powershell – нажать правой кнопкой мыши в папке, зажав Shift, и открыть окно Powershell) и выполнить следующую команду:
iverilog -o ./compiled ./src/runner.v ./src/runner_tb.v
При выполнении команды Icarus Verilog включает файлы runner.v и runner_tb.v и генерирует выходной файл compiled. Если же в исходных файлах содержится ошибка, то компиляция провалится и Icarus Verilog выведет в терминал название файла и строку, содержащую ошибку.

5. Симуляция модуля с помощью VVP

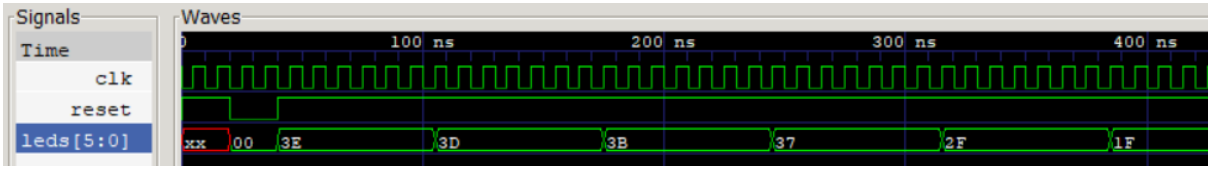
Далее необходимо выполнить симуляцию следующей командой:
vvp ./compiled
Verilog Virtual Processor проведёт симуляцию в соответствии с инструкциями в тестбенче. Благодаря указанным в нём командам по созданию выходного файла и записи в него, в папке с проектом появится файл runner_out.vcd – value change dump, описывающий изменения в значениях переменных на протяжении симуляции.

6. Просмотр полученной временной диаграммы в GTKWave

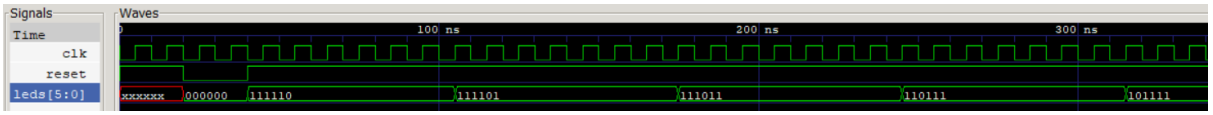
В папке gtkwave64, подпапке bin открыть gtkwave. В меню File открыть новую вкладку по команде Open New Tab и выбрать в диалоговом окне полученный ранее vcd-файл.
В окне SST нажать на модуль runner_tb, чтобы вывести его переменные в окне ниже.



Эти сигналы перетаскиваются на основное окно справа для визуализации. Чтобы уменьшить масштаб и рассмотреть всю длительность симуляции, можно использовать кнопки меню сверху или колёсико мыши с зажатой клавишей Ctrl.



По умолчанию для сигнала leds выбрано представление в виде шестнадцатеричных чисел. Для двоичного представления можно нажать правой кнопкой мыши на сигнал в меню Signals и выбрать в меню Data Format формат Binary.



Теперь заметно, что по битам массива leds бегает ноль, который на реальной плате включает один из диодов.