Домашнее задание №2.

Язык Verilog/SystemVerilog.
Задания можно выполнять и сдавать в любом порядке.

Для каждого задания необходимо:

- Написать синтезируемый модуль, который решает поставленную задачу.
- Написать модуль-тестбенч (testbench) к этому модулю, используя несинтезируемые конструкции.
- Провести симуляцию в ModelSim (или аналогичном симуляторе). Убедиться, что модуль корректно работает.
- Сделать проект в Quartus'е. Собрать синтезируемый модуль. Добиться успешного завершения сборки, почитать warnings и по возможности устранить их.

Если по уровню сложности, то они располагаются примерно так (по возрастанию):
A6, A5, A7, A2, A3, A1, A4.

Считаем, что все входные сигналы в заданиях A1-A4 синхронны с синхроимпульсом (clk). Никакой дополнительной пересинхронизации делать не надо.

A1. Часы

Необходимо сделать эмуляцию часов: на выход модуля выдается «текущее» время: часы, минуты, секунды, милисекунды. Синхроимпульс, который заходит в модуль имеет период 1ms. Необходимо предусмотреть возможность сброса времени в ноль, а так же предустановку какого-то из значений.

Интерфейс модуля:

Имя модуля: rtc_clock

Имя сигнала	Напр-е	Разрядность	Комментарий
clk_i	input	1	По условию, период клока 1 миллисекунда
srst_i	input	1	Синхронный сброс
cmd_type_i	input	3	Команда для выполнения какого-либо действия (сброс, установка одного из значений, ...)
cmd_valid_i	input	1	Сигнал валидности cmd_type_i

cmd_data_i	input	10	Просто данные, сопровождающие команду. Могут использоваться для установки пресета одного из значений.
hours_o	output	5	--
minutes_o	output	6	--
seconds_o	output	6	--
milliseconds_o	output	10	--

A2. Сериализатор

Сериализатор на вход получает «широкие» данные — длинное слово, и отправляет по одному биту эти данные на выход.

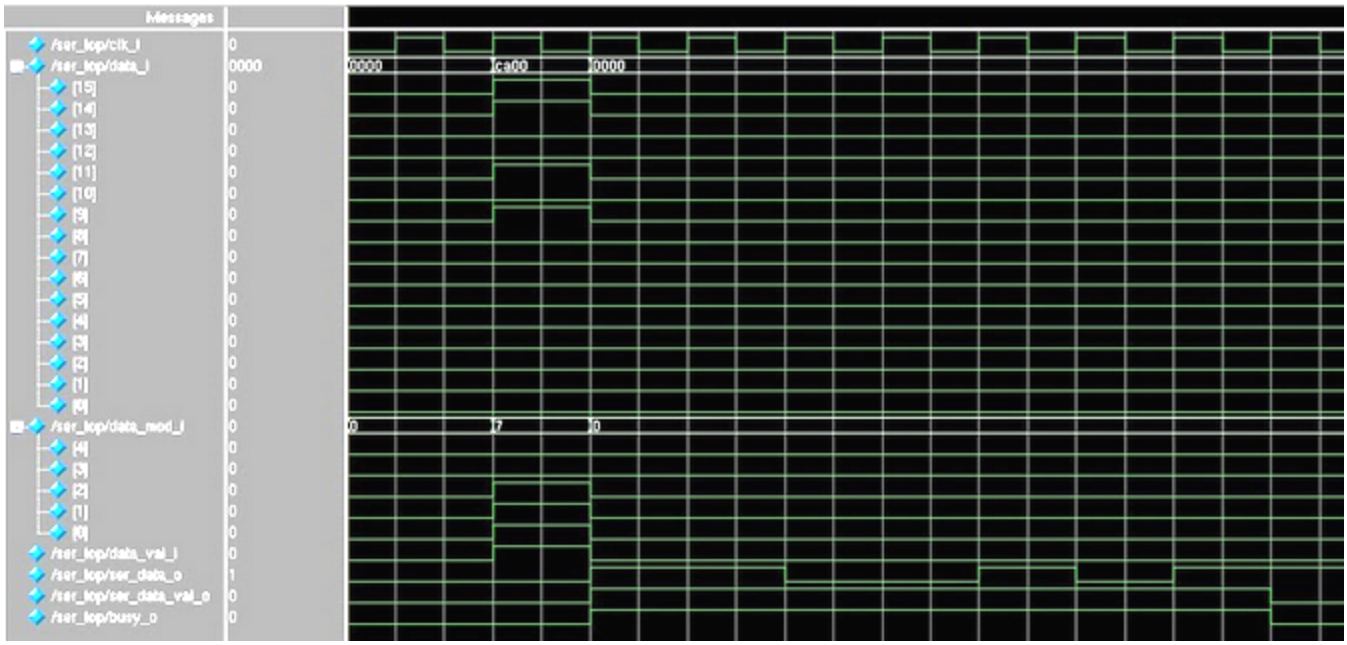
Интерфейс модуля:

Имя модуля: serializer

Имя сигнала	Напр-е	Разрядность	Комментарий
clk_i	input	1	Тактовый сигнал
srst_i	input	1	Синхронный сброс
data_i	input	16	Входные данные
data_mod_i	input	4	Число показывает сколько бит в входных данных валидно. К примеру, если там стоит 5, то валидны старшие 5 бит [15:11], и значит только эти пять бит будут «отправлены» дальше. Минимальное валидное число равно 3. Если mod меньше, то эта посылка игнорируется.
data_val_i	input	1	Сигнал, подтверждающий, что data_i и data_mod_i валидны, держится один такт.

ser_data_o	output	1	«Сериализованные» данные.Будем считать, что первым битом выходит самый старший бит из data_i.
ser_data_val_o	output	1	Подтверждает валидность ser_data_o.
busy_o	output	1	Если 1, то это говорит о том,что модуль сериализации занят, и вышестоящий модуль не пошлет новые data_i данные в тот такт, когда висит busy_o.

Времянки его работы:



A3. Десериализатор

Десериализатор выполняет функцию обратную сериализатору. Модуль “набирает” выходное слово из валидных входных бит начиная с нулевого бита. Как только набралось 16 валидных бит модуль выдает данные на выход.

Интерфейс модуля:

Имя модуля: deserializer

Имя сигнала	Напр-е	Разрядность	Комментарий
clk_i	input	1	Тактовый сигнал
srst_i	input	1	Синхронный сброс
data_i	input	1	Входные данные
data_val_i	input	1	Каждый валидный бит посылки, поступающий на data_i сопровождается активным состоянием этого сигнала (“1”)
deser_data_o	output	16	«Сериализованные» данные.Будем считать, что первым битом выходит самый старший бит из data_i.
deser_data_val_o	output	1	Подтверждает валидность ser_data_o.

A4. Светофор

Требуется написать модуль для управления светофором, который будет зажигать красный, желтый и зеленый фонари во всем известной последовательности: красный, красный и желтый, зеленый, зеленый моргает, желтый, красный.

Модуль может быть включен, выключен. При включении переходит в режим “красный”. Модуль может быть переведен в режим “неуправляемый переход” (это когда просто моргает желтый) и обратно.

Будем считать, что период “моргания” желтого и зеленого одинаковый и определяется статически. Для этого предусмотрен параметр, задающий полупериод моргания в ms.

Будем считать, что время нахождения в состоянии “зеленый мигает” задается статически параметром в кол-ве периодов моргания.

Будем считать, что время нахождения в состоянии “красный+желтый” равно и задается статически параметром в ms.

Время нахождения в остальных состояниях может быть настроено динамически в миллисекундах (как с часами). Для всех настроек используйте шину cmd_i, где будут закодированы все необходимые команды.

При включении устанавливаются значения по умолчанию

Интерфейс модуля

Имя модуля: traffic_lights

Параметр		Комментарий
BLINK_HALF_PERIOD		Длительность полупериода моргания в миллисекундах (время горения при миганиях) для режимов желтый мигает и зелый мигает
GREEN_BLINKS_NUM		Кол-во периодов моргания в состоянии “зеленый мигает”
RED_YELLOW_TIME		Время нах. в состоянии “красный+желтый”
RED_TIME_DEFAULT		Время нахождения в состоянии “красный” по умолчанию
YELLOW_TIME_DEFAULT		Время нахождения в состоянии “желтый” по умолчанию
GREEN_TIME_DEFAULT		Время нахождения в состоянии “зеленый” по умолчанию

Название	Напр-е	Разрядность	Комментарий
clk_i	input	1	Тактовый сигнал
srst_i	input	1	
cmd_type_i	input	3	Код команды для управления устройством (см коды ниже)
cmd_valid_i	input	1	Сигнал валидности команды
cmd_data_i	input	16	Просто данные, сопровождающие команду. Могут использоваться для установки пресета одного из значений.
red_o	output	1	Горит красный
yellow_o	output	1	Горит желтый
green_o	output	1	Горит зеленый

Коды команд:

0	Включить/перевести в стандартный режим
1	Выключить
2	Перевести в режим “неуправляемый переход”
3	Установить время горения зеленого
4	Установить время горения красного
5	Установить время горения желтого

Внимание: необходимо использовать конечный автомат для решения этого задания.

A5. Крайние единицы

Модуль должен во входном N-битном числе найти самую правую и самую левую единицу и оставить только ее.

Пример:
На входе: → На выходе
5'b00110 → 5'b00100; 5'b00010
5'b11111 → 5'b10000; 5'b00001
5'b00000 → 5'b00000; 5'b00000

Интерфейс модуля

Имя модуля: priority_encoder

Параметр		Комментарий
WIDTH		Разрядность данных

Имя сигнала	Напр-е	Разрядность	Комментарий
clk_i	input	1	Тактовый сигнал
srst_i	input	1	Синхронный сброс
data_i	input	WIDTH	Входные данные
data_left_o	output	WIDTH	Выходные данные (5'b00110 → 5'b00100)
data_right_o	output	WIDTH	Выходные данные (5'b00110 → 5'b00010)

A6. Количество единиц

Модуль должен в входном N-битном числе посчитать количество единиц.
latency (через сколько тактов будут могут доступны валидные выходные данные) на вашем усмотрении. Чем меньше, тем лучше.

Интерфейс модуля

Имя модуля: bit_population_counter

Параметр		Комментарий
WIDTH		Разрядность данных

Имя сигнала	Напр-е	Разрядность	Комментарий
clk_i	input	1	Тактовый сигнал
srst_i	input	1	Синхронный сброс
data_i	input	WIDTH	Входные данные
data_val_i	input	1	Сигнал валидности входных данных
data_o	output	$clog2(WIDTH) + 1	Выходные данные.
data_val_o	output	1	Сигнал валидности выходных данных

- $clog2(arg) -- встроенная в верилог функция, вычисляющая логарифм по основанию 2 от arg и округляющая его вверх до целого. Работает на этапе вычисления статических параметров (pre-synthesis).

A7. Антидребезг

Решить проблему дребезга кнопки, который может происходить при нажатии на нее (“глитчи”, возникающие при установлении нового состояния).

Интерфейс модуля:

Имя модуля: debouncer

Параметр		Комментарий
CLK_FREQ_MHZ		Частота clk_i в MHz
GLITCH_TIME_NS		Максимальное ожидаемое время, в течение которого наблюдается дребезг, в наносекундах.

Имя сигнала	Напр-е	Разрядность	Комментарий
clk_i	input	1	Синхроимпульс
key_i	input	1	Сигнал с кнопки. 0 - кнопка нажата, 1 - не нажата.
key_pressed_stb_o	output	1	Строб (импульс) на один такт, сообщающий, что кнопка была нажата

ПС: внутри модуля, очевидно, придется пересчитать GLITCH_TIME_NS в кол-во тактовых интервалов, опираясь на CLK_FREQ_MHZ. Подумайте, какое значение GLITCH_TIME_NS будет минимальным при частоте CLK_FREQ_MHZ = 20?