Задание:
Разработать синтезируемое описание модуля (RTL), используя Verilog/SystemVerilog и тестовое окружение к нему.

Требования:
Модуль должен иметь интерфейс, описанный в задаче. В том случае, если при выполнении задания какие-либо входные сигналы не были задействованы, объяснить почему.

Плюсы при выполнении задания:
- Испрользование при разработке Assertions
- Синтез полученного RTL (Vivado/Quartus)

Вариант	Модуль	Параметры	Интерфейс			Функциональное описание	Доп. требования к тестовому окружению
1	FIFO	DEPTH - глубина очереди WIDTH - ширина входного интерфейса	clk	input	Синхросигнал	Очередь FIFO (First-in First-out). Входной и выходной интерфейсы используют handshaking.	Обязательно промоделировать ситуацию полного заполнения очереди
			rst_n	input	Ресет		
			s_val	input	Валидность входных данных		
			s_data[WIDTH-1:0]	input	Входные данные		
			s_rdy	output	Готовность очереди к приёму данных		
			m_val	output	Валидность выходных данных		
			m_data[WIDTH-1:0]	output	Выходные данные		
			m_rdy	input	Готовнсть окружения к приёму данных		
2	4-входовой арбитр с круговым приоритетом		clk	input	Синхросигнал	Арбитр выдаёт грант одному из направлений согласно правилам кругового приоритета	
			rst_n	input	Ресет		
			val[3:0]	input	Входные запросы		
			gnt[3:0]	output	Выдача гранта		
3	Crossbar 2x2 с фиксированным приоритетом	WIDTH - ширина входного интерфейса	clk	input	Синхросигнал	Кроссбар распределяет запросы от двух входных интерфейсов (s0 и s1) между двумя выходными (m0 и m1), причём интефрейс s0 имеет приоритет над s1	
			rst_n	input	Ресет		
			s0_val	input	Валидность входных данных на шине 0		
			s0_dst	input	Направление запроса (0: m0; 1: m1)		
			s0_data[WIDTH-1:0]	input	Входные данные на шине 0		
			s0_rdy	output	Готовность модуля к приёму данных на шине 0		
			s1_val	input	Валидность входных данных на шине 1		
			s1_dst	input	Направление запроса (1: m0; 1: m1)		
			s1_data[WIDTH-1:0]	input	Входные данные на шине 1		
			s1_rdy	output	Готовность модуля к приёму данных на шине 1		
			m0_val	output	Валидность выходных данных на шине 0		
			m0_src	output	Источник запроса (0: s0; 1: s1)		
			m0_data[WIDTH-1:0]	output	Входные данные на шине 0		
			m0_rdy	input	Готовность окружения к приёму данных на шине 0		
			m1_val	output	Валидность выходных данных на шине 1		
			m1_src	output	Источник запроса (1: s0; 1: s1)		
			m1_data[WIDTH-1:0]	output	Входные данные на шине 1		
			m1_rdy	input	Готовность окружения к приёму данных на шине 1		
4	Преобразователь в код Грея и обратно	WIDTH - ширина входного интерфейса	Module 0:			В результете должно получиться 2 модуля: bin2gray и gray2bin, выполняющие преобразование из бинарного представления в код Грея и обратно	В тестовом окружении модули замкнуты друг на друга и обмениваются gray-coded данными, в то время как окружение проверяет binary-данные
			bin[WIDTH-1:0]	input	Входное значение (binary)		
			gray[WIDTH-1:0]	output	Выходное значение (gray)		
			Module 1:				
			gray[WIDTH-1:0]	input	Входное значение (gray)		
			bin[WIDTH-1:0]	output	Выходное значение (binary)		
5	Bus Downsizer 32x8		clk	input	Синхросигнал	Устройство принимает 32-битное слово на входном интерфейсе, разбивает его на 4 8-битных и последовательно передаёт на выходной m_data<0> = s_data[7:0] m_data<1> = s_data[15:8] m_data<2> = s_data[23:16] m_data<3> = s_data[23:16]	
			rst_n	input	Ресет		
			s_val	input	Валидность входных данных		
			s_data[31:0]	input	Входные данные		
			s_rdy	output	Готовность очереди к приёму данных		
			m_val	output	Валидность выходных данных		
			m_data[7:0]	output	Выходные данные		
			m_rdy	input	Готовнсть окружения к приёму данных		
6	Bus Upsizer 8x32		clk	input	Синхросигнал	Устройство принимает 4 8-битных слов на входном интерфейсе, склеивает из них одно 32-битное и передаёт на выходной интерфейс m_data = {s_data<3>, s_data<2>, s_data<1>, s_data<0>}	
			rst_n	input	Ресет		
			s_val	input	Валидность входных данных		
			s_data[7:0]	input	Входные данные		
			s_rdy	output	Готовность очереди к приёму данных		
			m_val	output	Валидность выходных данных		
			m_data[31:0]	output	Выходные данные		
			m_rdy	input	Готовнjсть окружения к приёму данных		
7	Protocol Converter: handshaking -> token	WIDTH - ширина входного интерфейса DEPTH - глубина буфера MAX_CRD_LIMIT - максимальное число неиспользованных токенов	clk	input	Синхросигнал	Устройство принимает и буферизирует сообщения на входном интерфейсе, используя handshaking Выдача сообщений возможна только при наличии положительного числа токенов, асинхронно приходящих от внешней среды. После ресета число токенов в устройстве - MAX_CRD_LIMIT	Обязательно промоделировать ситуацию полного заполнения буфера и отсутствия в буфере токенов
			rst_n	input	Ресет		
			s_val	input	Валидность входных данных		
			s_data[WIDTH-1:0]	input	Входные данные		
			s_rdy	output	Готовность очереди к приёму данных		
			m_val	output	Валидность выходных данных		
			m_data[WIDTH-1:0]	output	Выходные данные		
			m_tkn	input	Токен, разрешающий приём одного сообщения		
8	Protocol Converter: token -> handshaking	WIDTH - ширина входного интерфейса DEPTH - глубина буфера MAX_CRD_LIMIT - максимальное число неиспользованных токенов	clk	input	Синхросигнал	Устройство принимает и буферизирует сообщения на входном интерфейсе и выдаёт в окружение токены, причём устройство должно гарантированно принимать сообщение при наличии у окружения свободных токенов. При выдаче данных наружу используется handshaking. После ресета число токенов в окружении - MAX_CRD_LIMIT	Обязательно промоделировать ситуацию полного заполнения буфера
			rst_n	input	Ресет		
			s_val	input	Валидность входных данных		
			s_data[WIDTH-1:0]	input	Входные данные		
			s_tkn	output	Токен, разрешающий приём одного сообщения		
			m_val	output	Валидность выходных данных		
			m_data[WIDTH-1:0]	output	Выходные данные		
			m_rdy	input	Готовнjсть окружения к приёму данных		