Спецификация на ядро UART

RB

25 августа 2023 г.

Список иллюстраций

Список таблиц

1 Введение

Данная спецификация описывает ядро UART, управление которым осуществляется при помощи интерфейса Avalon-MM или, в перспективе, AXI Lite (на данный момент находится в стадии разработки).

1.1 Интеграция

Изначально ядро предполагает интеграцию в QSYS САПР Quartus, но никто не запрещает добавлять его в любой другой дизайн. Пример интеграции показан на рисунке 1

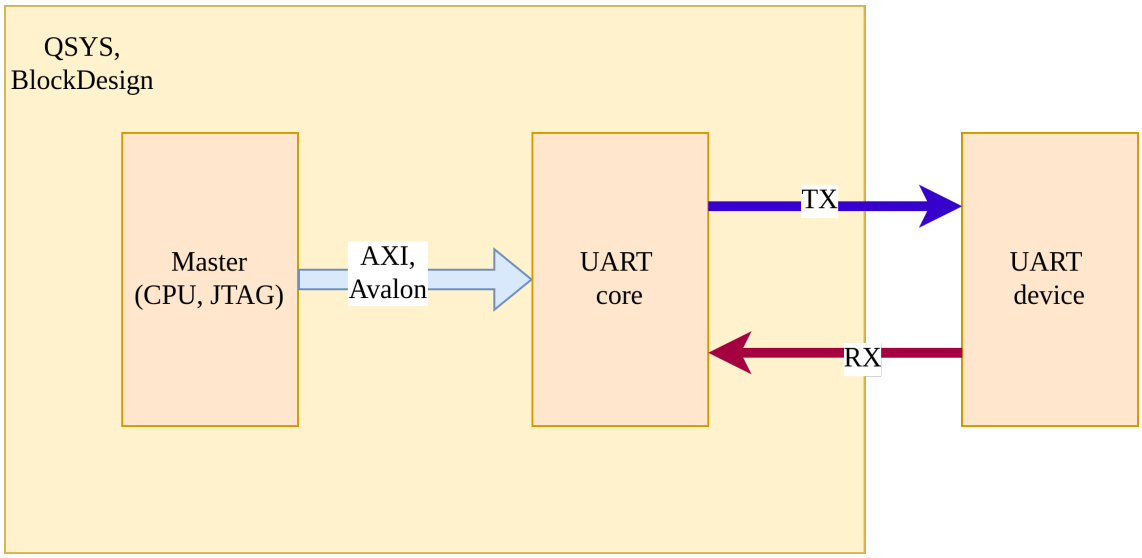

Рис. 1: Пример интеграции ядра в SoC

1.2 Описание сигналов

В таблице 1 перечислены сигналы модуля для связи с внешним миром.

1.3 Возможности ядра

Данное ядро обладает следующими возможностями:

- поддержка интерфейсов Avalon-MM и AXI Lite;
- программная установка любого BaudRate, в том числе не из таблицы стандартных скоростей;
- программное включение бита четности;
- поддержка odd и even битов четности;
- возможность установки до четырех стоп-битов
- настраиваемый размер приемного и передающего буферов.

Таблица 1: Описание сигналов

Имя	Направление	Ширина	Описание
Глобальные сигналы			
clk	input	1	тактовый сигнал для работы модуля
reset_n	input	1	асинхронный сброс по заднему фронту
Интерфейс Avalon-MM Slave			
avmms_write_i	input	1	write
avmms_address_i	input	3	address
avmms_writedata_i	input	32	writedata
avmms_byteenable_i	input	4	byteenable
avmms_read_i	input	1	read
avmms_waitrequest_o	output	1	waitrequest
avmms_readdata_o	output	32	readdata
Интерфейс UART			
uart_rx	input	1	линия RX
uart_tx	output	1	линия TX

2 Использование

2.0.1 Структура директорий

Проект содержит имеет следующую структуру директории:

```
.
├── RTL
├── SIM
├── OTHER
└── DOC
    └── FIGURE
```

`RTL` - исходники ядра на языках Verilog и SystemVerilog,

`SIM` - файлы для симуляции. **uart_mm_top_tb.sv** - тестбенч для симуляции всего ядра. Для него предназначен **uart_mm_top_wave.do** - скрипт для инициализации окна **wave** в **ModelSim/QuestaSim**). **uart_tb.sv** - тестбенч для симуляции только приемопередатчика (проверка битов четности, стоп-битов и т.д.). Для него предназначен **wave.do** - скрипт для инициализации окна **wave** в **ModelSim/QuestaSim**)

`OTHER` - для прочих файлов, содержит таблицу стандартных скоростей,

`DOC` - документация на ядро. Исходники в формате TEX,

`DOC/FIGURE` - хранилище изображений.

3 Архитектура

На рисунке 2 представлена архитектура IP-ядра и основные пути следования данных.

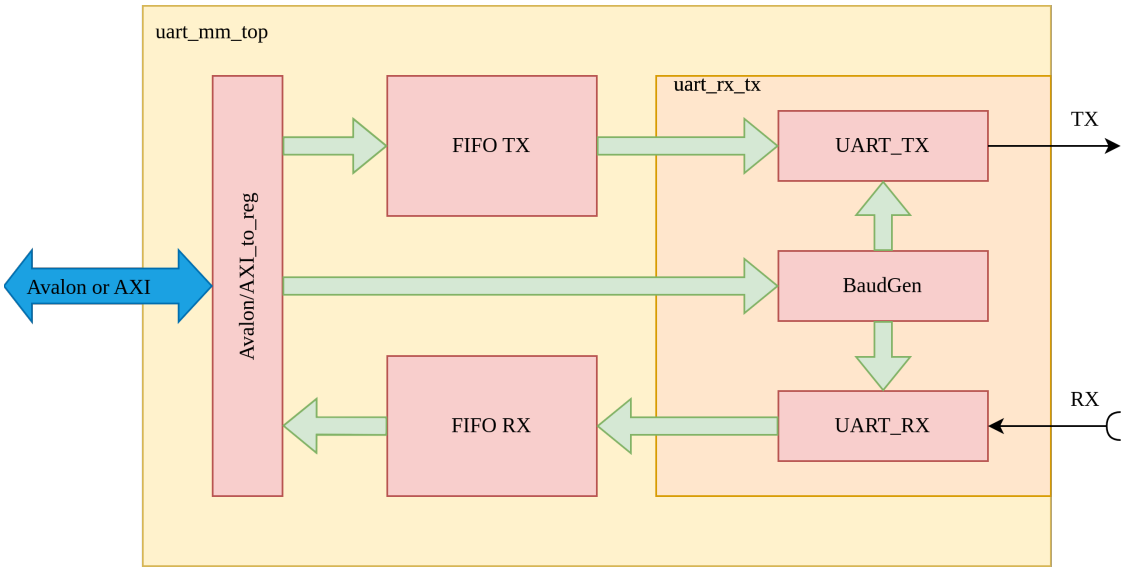

Рис. 2: Архитектура IP-ядра

Рассмотрим процесс передачи. Через интерфейс **Avalon** или **AXI** пользователь осуществляет запись байта в регистр **TX_FIFO**(см. п. 4), далее байт для записи попадает в очередь на передачу - **FIFO_TX**. Данные из **FIFO_TX** попадают на модуль **UART_TX**, где происходит их сериализация и выдача на линию **TX**. Модуль **BaudGen** осуществляет генерацию сигналов для выдачи данных (и приема) на частоте, которую задает пользователь (например 9600 бод). Процесс приема аналогичен процессу передачи. Модуль **UART_RX** фиксирует стартовый бит посылки и начинает десериализацию данных в зависимости от параметров, выставленных пользователем (наличие бита четности и количество стоповых бит). Десериализованный байт попадает в **FIFO_RX**, заодно выполняется проверка бита четности (если последний включен в регистре **CONTROL**) и результат так же записывается в **FIFO_RX**.

4 Карта регистров

Модуль имеет 6 регистров управления, представленные в таблице 2. В таблице приняты следующие обозначения: RO - Read Only (только для чтения), WO - Write Only (только для записи), WR - Write or Read (чтение и запись). Default - значение, принимаемое регистром после сброса.

Имя	Адрес, DW	Ширина	Доступ	Описание
CONTROL	3'b000	32	RW	Управление параметрами передачи
BAUD	3'b001	32	RW	Установка BaudRate
FILL_TX	3'b010	32	RO	Число байт, ожидающих передачи из TX FIFO
FILL_RX	3'b011	32	RO	Число байт, ожидающих чтения из RX FIFO
TX_FIFO	3'b100	32	WO	Запись байт в буфер передачи
RX_FIFO	3'b101	32	RO	Чтение байт из буфера приема

Таблица 2: Регистры управления ядра UART

4.1 CONTROL

Регистр **CONTROL** служит для управления процессами приема, передачи и контроля за флагами приемного и передающего FIFO.

Bit	Access	Default	Decription
[0]	RO	1'b1	FIFO TX Empty
[1]	RO	1'b0	FIFO TX Full
[2]	RO	1'b1	FIFO RX Empty
[3]	RO	1'b0	FIFO RX Empty
[7:4]	RO	4'b0	Reserved
[8]	RW	1'b0	Enable parity bit
[9]	RW	1'b0	Type parity bit
[11:10]	RW	2'b0	Count stop bits
[12]	RW	1'b1	Enable transmitter
[13]	RW	1'b1	Enable receiver

Таблица 3: Биты регистра CONTROL

FIFO TX Empty - данный бит выставляется в единицу если FIFO TX не содержит никаких данных для передачи.

FIFO TX Full - данный бит выставляется в единицу если FIFO TX заполнено. Данные, записываемые в FIFO TX при установленном флаге, будут потеряны.

FIFO RX Empty - данный бит выставляется в единицу если FIFO RX не содержит данных для чтения.

FIFO RX Full - данный бит выставляется в единицу если FIFO RX заполнено. Все данные, которые в дальнейшем будут приняты от UART не запишутся и будут утеряны.

Enable parity bit - включение бита четности.

Type parity bit - тип бита четности, 1'b0 - EVEN, 1'b1 - ODD.

Count stop bits - количество стоповых бит: 2'b00 - 1, 2'b01 - 2, 2'b10 - 3, 2'b11 - 4;

Enable transmitter - данный бит разрешает работу ядра на передачу по UART. В случае отключения передатчика поступающие данные будут накапливаться в FIFO TX.

Enable receiver - данный бит разрешает работу ядра на прием по UART. В случае отключения приемника, приниемые данные по UART не будут записываться в FIFO RX, а будут отбрасываться.

4.2 BAUD

Данный 32-битный регистр служит для управления скоростью, на которой работает приемник и передатчик UART. Значение регистра вычисляется согласно следующему выражению:

$$BAUD = round\left(\frac{SystemClock}{8 * BaudRate}\right) - 1,$$

где $SystemClock$ - тактовая частота работы модуля в Гц, $BaudRate$ - скорость UART в бодах (например, стандартная табличная), $round()$ - операция округления.

4.3 FILL_TX

Данный 32-битный регистр указывает количество байт, которые находятся в FIFO TX и ожидают передачи по UART. При попадании байта в FIFO_TX происходит увеличение счетчика FILL_TX на единицу. Если модуль UART_TX забирает байт из FIFO_TX для передачи, то FILL_TX уменьшается на единицу и т.д.

4.4 FILL_RX

Данный 32-битный регистр указывает количество байт, принятых по UART и ожидающих чтения из FIFO RX. При попаданиие байта из модуля UART_RX в FIFO RX происходит увеличение счетчика FILL_RX на единицу. Если пользователь читает байт из FIFO RX путем доступа к регистру RX_FIFO, то счетчик FILL_RX уменьшается на единицу и т.д.

4.5 TX_FIFO

Данный регистр служит для записи передаваемых байт в FIFO TX. Для того, чтобы поместить байт $byte[7:0]$ в FIFO TX для последующей передачи, необходимо записать его в данный регистр. Ядро принимает для передачи только младшие 8 бит - $TX\_FIFO[7:0]$, игнорируя остальные. Так что не важно, мастер какой ширины управляет модулем (32-,16-,8-битный).

4.6 RX_FIFO

Данный регистр служит для чтения байт из FIFO RX. Поддерживается только одновременное чтение одного байта из FIFO RX (burst не поддерживается). При чтении $RX\_FIFO[7:0]$ возвращает прочитанный из FIFO RX байт, $RX\_FIFO[8]$ возвращает статус проверки бита четности - $1'b1$ в случае ошибки. Данный бит всегда принимает нулевое значение в случае если бит четности отключен в регистре **CONTROL**.