Universal Asynchronous Receiver and Transmiter (UART)

COSC-594/690

Stephen Marz

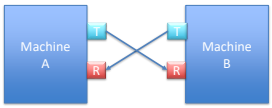
TICKLE COLLEGE OF ENGINEERING

MIN H. KAO DEPARTMENT OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE

1

Hardware

- UART has two wires.
  - A's Receiver to B's Transmitter
  - A's Transmitter to B's Receiver



2

Clocking

- UART is asynchronous
  - Both machine A and B keep a separate clock
  - The clock isn't shared between the two
- The BAUD rate is important
  - As long as the clocks don't drift, the periodicity of the clock should be equivalent between two machines.

3

UART Protocol

- Both machines must agree on a few things
  - BAUD (signaling) rate
    - Essentially a ratio to the clock frequency
  - # of Start bits
  - # of Stop bits
  - Parity: even, odd, or none
- We use 115200n8 (*Baud***Parity***Databits*)
  - 115200 baud rate
  - No parity (n)
  - 8 bits data
  - Implied: 1 start and 1 stop bit.

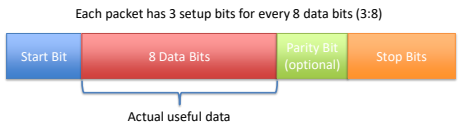
4

UART Packet

- Start bit
  - A line change from high to low signals that a packet is coming.
- Data bits
  - 8 more bits representing the data in this one packet.
- Parity bit
  - Skipped with "no" parity
- Stop bits
  - Line is moved from low to high

5

UART Packet



6

UART Microcontroller

- SiFive and ns16550a
- These will handle transmitting and receiving packets.
  - They both store the data in hardware buffers.
- Fire-and-forget
  - Tell the microcontroller to read or write
    - it essentially does the rest.

7

Programming SiFive

Memory Map

UART Register Offsets		
Offset	**Name**	**Description**
0x000	txdata	Transmit data register
0x004	rxdata	Receive data register
0x008	txctrl	Transmit control register
0x00C	rxctrl	Receive control register
0x010	ie	UART interrupt enable
0x014	ip	UART Interrupt pending
0x018	div	Baud rate divisor

Table 19.1: Register offsets within UART memory map.

8

Programming SiFive

Transmit Register

Transmit Data Register (txdata)				
Register Offset		0x000		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[7:0]	data	RW	X	Transmit data
[30:8]	*Reserved*	RW	X	
31	full	RW	X	Transmit FIFO full

Table 19.2: Transmit Data Register

9

Programming SiFive

Receive Register

Receive Data Register (rxdata)				
Register Offset		0x004		
Bits	Field Name	Attr.	Rst.	Description
[7:0]	data	RO	X	Received data
[30:8]	*Reserved*	RW	X	
31	empty	RO	X	Receive FIFO empty

Table 19.3: Receive Data Register

10

Programming SiFive

Control Registers

Transmit Control Register (txctrl)				
Register Offset		0x008		
Bits	Field Name	Attr.	Rst.	Description
0	txen	RW	0x0	Transmit enable
1	nstop	RW	0x0	Number of stop bits
[15:2]	*Reserved*	RW	X	
[18:16]	txcnt	RW	0x0	Transmit watermark level
[31:19]	*Reserved*	RW	X	

Table 19.4: Transmit Control Register

Receive Control Register (rxctrl)				
Register Offset		0x00C		
Bits	Field Name	Attr.	Rst.	Description
0	rxen	RW	0x0	Receive enable
[15:1]	*Reserved*	RW	X	
[18:16]	rxcnt	RW	0x0	Receive watermark level
[31:19]	*Reserved*	RW	X	

Table 19.5: Receive Control Register

11

Programming SiFive

Rate Register

$$f_{\text{baud}} = \frac{f_{\text{in}}}{\text{div} + 1}$$

Baud Rate Divisor Register (div)				
Register Offset		0x018		
Bits	Field Name	Attr.	Rst.	Description
[15:0]	div	RW	0xFFFF	Baud rate divisor
[31:16]	*Reserved*	RW	X	

Table 19.8: Baud Rate Divisor Register

12

Programming SiFive

- Set the divisor using BAUD formula.
- Timer frequencies
  - HiFive1: 17,422,745
  - E31: 32,500,000
  - Qemu: 65,000,000
    - Qemu will really take any frequency and still work, but this makes it a bit more stable.
- Write to *div* register

$$f_{\text{baud}} = \frac{f_{\text{in}}}{\text{div} + 1}$$

13

Divisor Example

$f_{baud} = 115{,}200$

$f_{in} = 65{,}000{,}000$

$f_{baud} = {}^{f_{in}}/_{div+1}$

$$115{,}200 = \frac{65{,}000{,}000}{div + 1}$$

$$115{,}200 \times (div + 1) = 65{,}000{,}000$$

$$div + 1 = \frac{65{,}000{,}000}{115{,}200}$$

$$div = \frac{65{,}000{,}000}{115{,}200} - 1$$

$$div = 564 - 1 = 563$$

The divisor is 563, which is the value that needs to go into the *div* register.

The microcontroller will divide the clock by 563 periods to match a signaling rate of 115,200 baud.

14

UART

COSC-594/690

Stephen Marz

THE UNIVERSITY OF TENNESSEE KNOXVILLE

15