CSULB - CALIFORNIA STATE UNIVERSITY LONG BEACH
CECS – COMPUTER ENGINEERING AND COMPUTERSCIENCE DEPARTMENT
CECS 460 – SYSTEM ON CHIP DESIGN

SYSTEM ON CHIP SPECIFICATION
UART + PICOBLAZE KCPSM6 + CELLRAM

Filipe Calasans Portugal de Oliveira

May 2013

1. Introduction

The LBUART module provides asynchronous communication referred as UART RS232 Transmitter with easy interfacing with microcontrollers. It is able to communicate with rates up to 115200 BAUD, parity support and 8/7-bit word length. This module does not provide FIFO support, but can be easily interfaced with one to prevent data loss, and ensure that the processors will not spend much time servicing the UART rather than compute more critical tasks. This module was interfaced with the Xilinx Picoblaze KCPSM6 8-bit microprocessor on a SoC.

1.2 Features

- 7/8 bits word length.
- None, Odd and Even Parity support.
- Baud rate: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200.
- No FIFO included.
- 1 Stop bit.
- All features present on the Picoblaze KCPSM6 8-bit microprocessor

2. Applicable documents

2.1.1 Customer specifications

Slides “TX defined” and “UART” provided by Mr. Tramel in class.

3. Requirements

3.1.1 Performance Requirements

The Table 1 shows the bit time requirement for each BAUD rate value. The UART module must be able to generate and sample bits with duration sufficient close of those values. The module must accomplish the read/write Picoblaze IO port requirements, for it can be interfaced with the processor.

BAUD Rate	Bit Time	Engineering Notation
300	3.3333E-03	3.3333 ms
1200	8.3333E-04	833.33 us
2400	4.1667E-04	416.66 us
4800	2.0833E-04	208.33 us
9600	1.0417E-04	104.16 us
19200	5.2083E-05	52.083 us
38400	2.6042E-05	26.041 us
57600	1.7361E-05	17.361 us
115200	8.6806E-06	8.6806 us
230400	4.3403E-06	4.3403 us
460800	2.1701E-06	2.1701 us
921600	1.0851E-06	1.0851 us

Table 1: Bit time for each baud rate value.

4. Implementation

4.1 Design Description UART

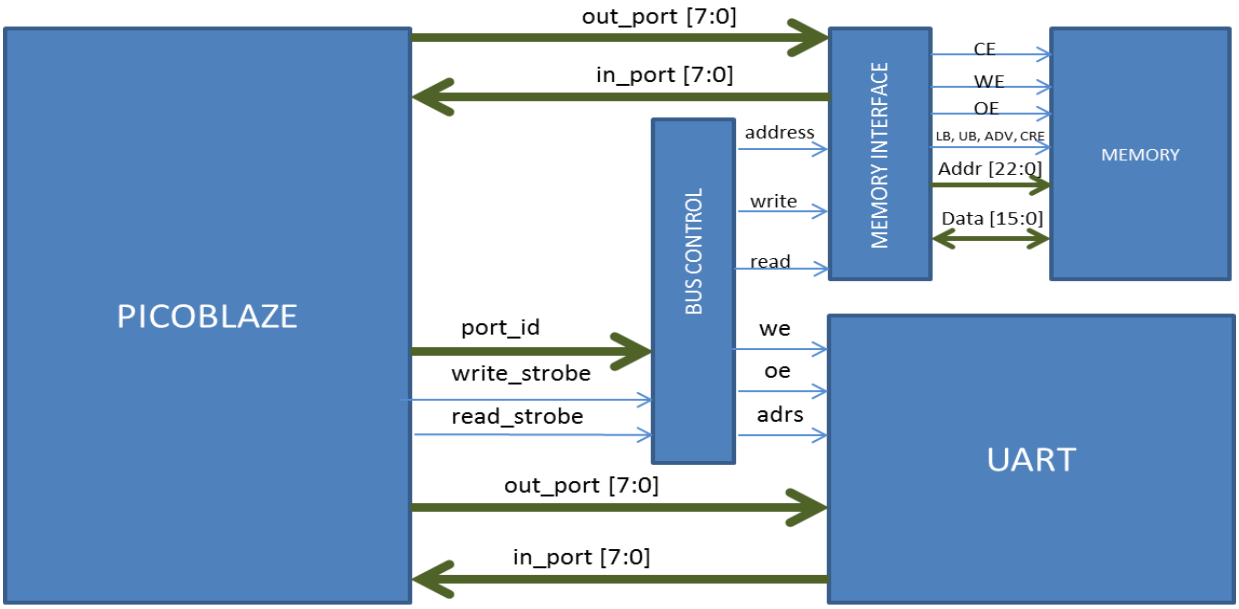
The UART module implemented is compound by two independent modules: TX and RX. They run over the same configuration (bit8, parity_en, odd_n_even and BAUD) which influence the number of bits read/written as well the transmition velocity. The module is capable to receive and transmit simultaneously. Data are available to users through TX register, Status register and RX buffer register.

To transmit data, first the data is loaded into the TX shift register then according to configuration bits bit8, parity_en and odd-n_even, additional bits are included to data bits, then the data is transmitted sequentially through the tx line respecting the bit time shown in Table 1 on section 3.1.1. The sequence is transmitted according to the following sequence: start bit (0), followed by data bits from LSB to MSB, parity bit when parity_en is asserted and stop bit (1). The data can be written into the Shift register until the TXrdy flag (STATUS[1]) is equal to one.

To receive data the UART Rx engine verifies the RX line watching if any transition from one to zero occurs, this indicates the transmission beginning. Then the data is captured by the RX shift register that is responsible to shift data in, following the data bit time generate by the RX FSM. The Status register can be accessed by users to know the current status of the RX module.

4.2 Block Diagram

SOC



The SoC has the major components: Picoblaze, Memory, Bus Control, Memory Interface, UART.

Picoblaze: A Xilinx KCPSM6 8-bits microprocessor.

Memory: Async ClellularRAM 1.5 MT45W8MW16BGX. 8MBx16.

Bus Control: Module responsible to arbitrate which device must have the control of the data bus. The bus control module knows which peripheral must be selected by the signals Port ID, Write and read strobe.

Memory Interface: This module allows the 8-bit microprocessor communicates with the memory which has address line of 23 bits and data line of 16 bits, this is possible because this module has some registers able to store the addresses and data fragments.

This module is responsible also to generate the memory control signals that accomplish the memory time requirements.

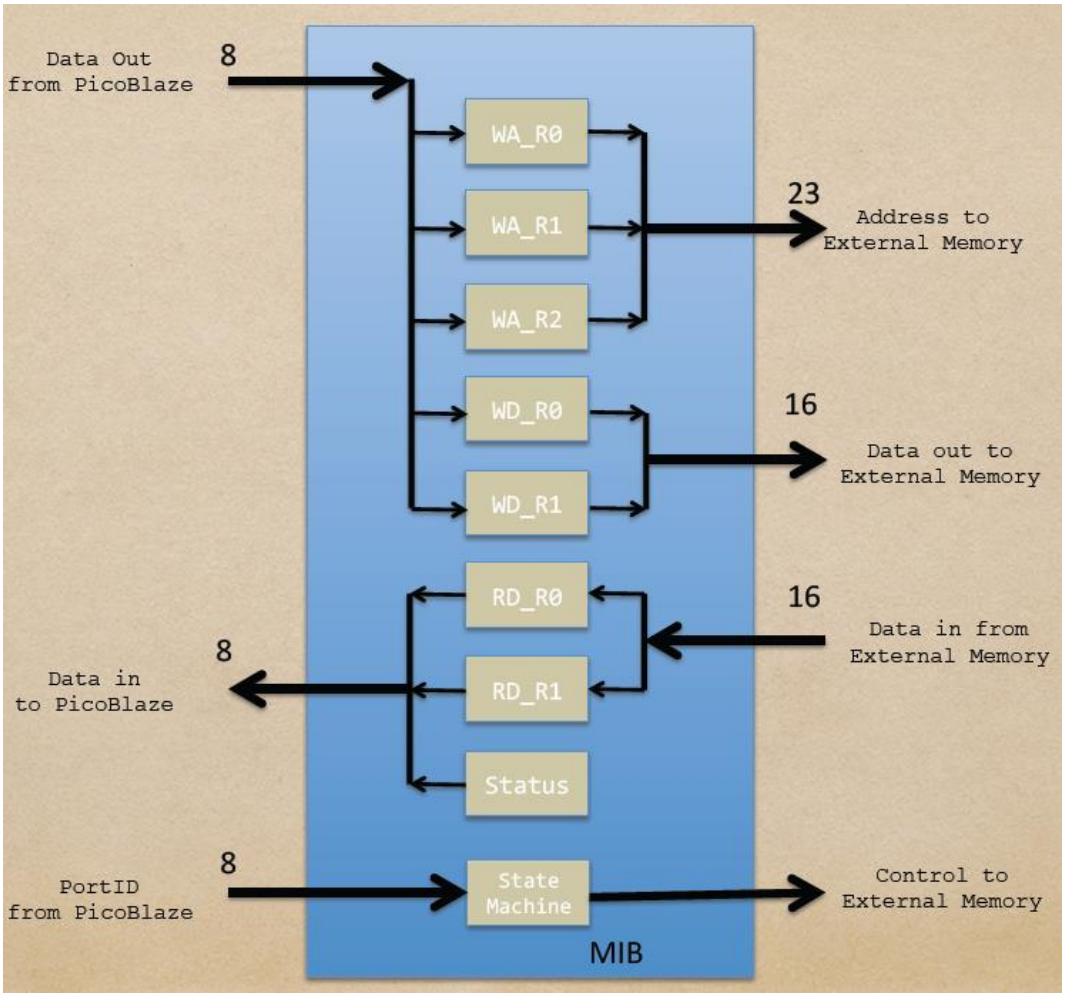
UART: see in section 4.1.

The following table shows the IO map.

Table 2: Memory IO map.

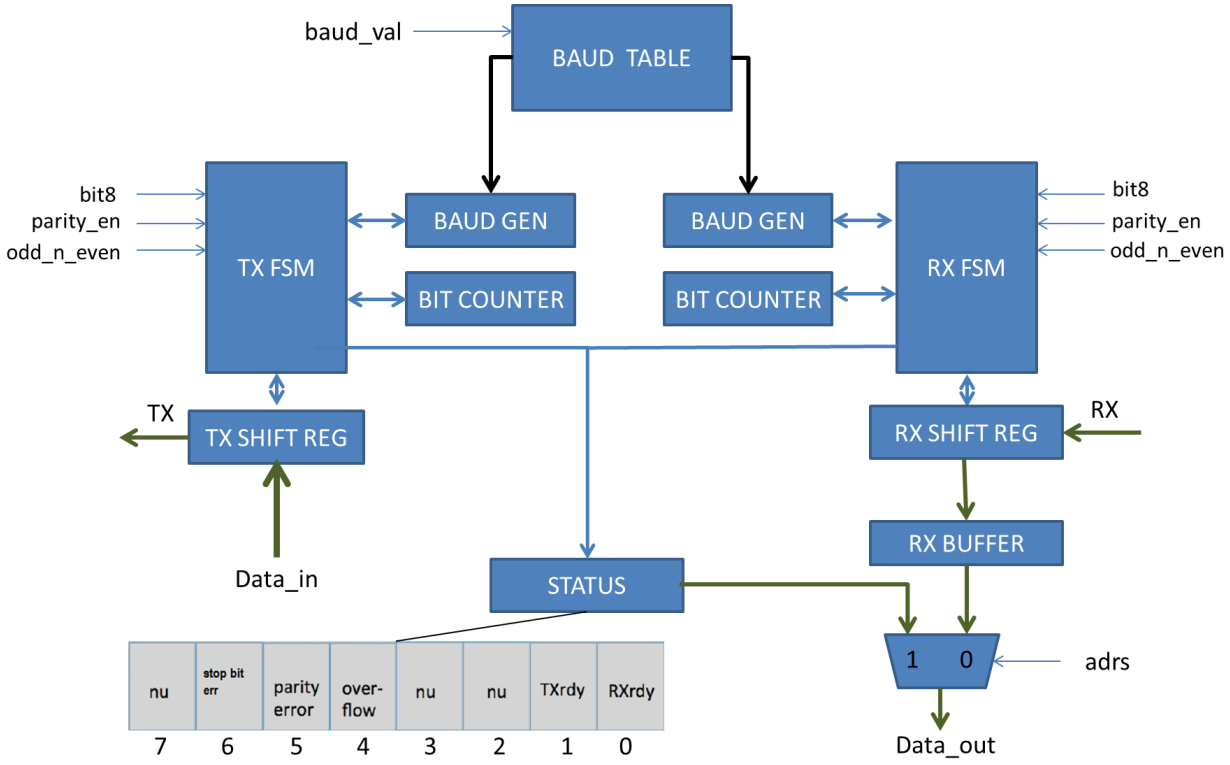
IO Map		
Functionality	**Address**	**Operation**
Port write memory address 0	0x01	OUTPUT
Port write memory address 1	0x02	OUTPUT
Port write memory address 2	0x03	OUTPUT
Port write memory data 0	0x04	OUTPUT
Port write memory data 1	0x05	OUTPUT
Port read memory data 0	0x06	INPUT
Port read memory data 1	0x07	INPUT
Port do memory write	0x08	OUTPUT
Port do memory read	0x09	OUTPUT
Port uart data	0x0A	INPUT, OUTPUT
Port read status UART	0x0B	INPUT
Port read status MEMORY	0x0C	INPUT

MEMORY INTERFACE



The Memory interface block allows the user write 16-bit data in a memory addressed for 23bits. For, the interface has 3 registers which is responsible for store the addresses, 2 registers responsible to store data read from memory, 2 register that store data to be write in the memory and one register responsible for storing the memory status. These register can be accessed using the instructions INPUT and OUTPUT of the Picoblaze instruction set. The memory map is shown in the Table 2.

UART



General Description:

The modules FSM are responsible for generates the control signals to the modules: Shift registers, Baud GEN and Bit Counter.

BAUD GEN: Responsible to generate a 1 clock tick pulse when the module reaches the bit time. Bit time depends of the value of baud_val signal. This value can be changed when the module is reset or the load is requested.

Bit Counter: It is responsible to count the number of bits that have been transmitted/received.

TX Shift Register: Responsible to store the data that will be transmitted and shift it every time the FSM generates the high level signal shift. This pulse is generated every time the **BAUD GEN** ends to count the bit time.

RX Shift Register: it is responsible to shift data RX into the register every time the FSM generates the high level signal shift.

Rx Buffer: It is responsible to store the data received. The user has access to this register when *adrs* signal is zero.

Status: This module stores the status flags. It can be accessed by users setting the adrs signal to one.

4.3 I/O Interface Description UART

Name	Type	Mode	Description
CLK	Input	Sync/Async	Main system clock 100 MHz
Reset	Input	Sync/Async	Active low assynchronus
Data_in[7:0]	Input	Sync	Transmit write data bus
Data_out[7:0]	Output	Sync/Async	Receive data bus, Rx buffer and Status register
WE	Input	Sync/Async	Active high write enable. This signal indicates that the data presented on data_in[7:0] bus should be registered by transmit buffer. This signal can be active while the transaction is occurring but must be disserted before the transmit has finished the transmission and should only be active when TXrdy signal, status[1], is active.
OE	Input	Sync/Async	Active high read enable. This signal is used to indicate that data on data_bus[7:0] has been read and will reset the flags: error stop bit, overflow, parity and Data Rx ready (status[6:4] and status[0]).
CSn	Input	Sync/Async	Active low chip select. The Csn signal qualifies both WE and OE signals.
Bit8	Input	Sync/Async	Control bit for data bit width for both receive and transmit functions. When bit8 is a logical ‘1’, then the data width is eight bits, otherwise, the data width is seven bits and data defined by data_in[7] is ignored and data_out[7] is a don`t care.
Parity_en	Input	Sync/Async	Control bit to enable parity for both receive and transmit functions. Parity is enable when the bit is set to logical '1'.
Odd_n_even	Input	Sync/Async	Control bit to define odd or even parity for both receive and transmit functions. When the parity_en control bit is set , a ‘1’ on this bit indicates odd parity and ‘0’ indicates even parity
Baud_val	Input	Async	4 bit control bus used to define the baud value.

Adrs	Input	Sync	This bit control which data is put on the bus data_out. When the bit is set to logical value ‘1’ the status register is put on the bus data_ou[7:0] otherwise the rx data available is put on it.
Rx	Input	Sync/Async	Serial receive data
Tx	Output	Sync/Async	Serial Transmit data.

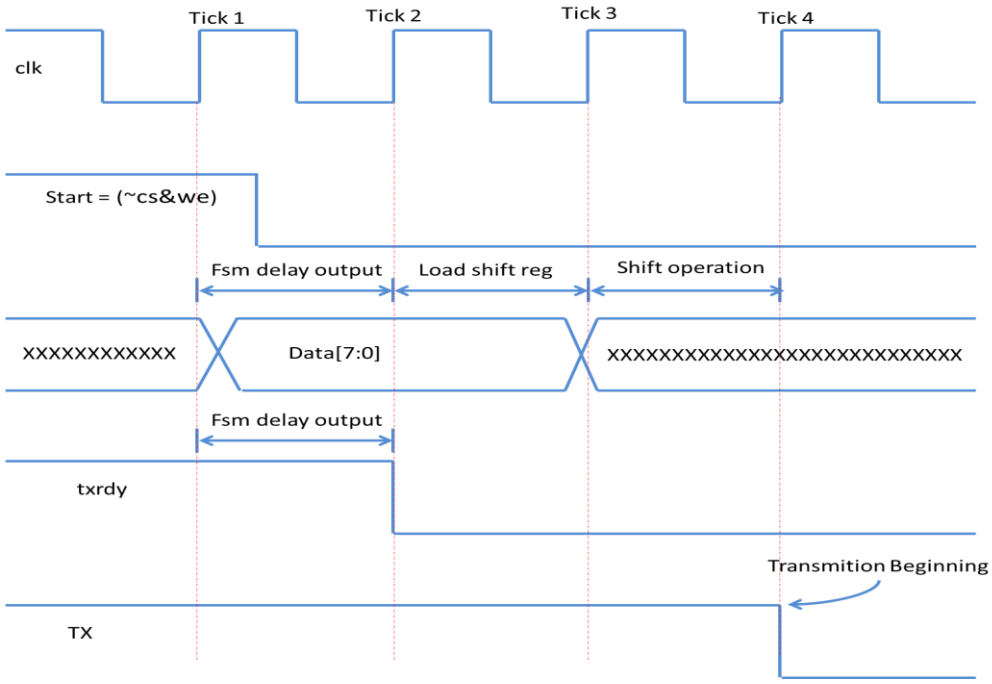
System On Chip: UART + Picoblaze + Memory

Name	Type	Mode	Description
CLK	Input	Sync/Async	Main system clock 100 MHz
Reset	Input	Sync/Async	Active low assynchronus
Interrupt	Input	Sync/Async	Active High interrupt control. Providing interrupts have been enabled within the program then when this input is driven High KCPSM6 will perform an interrupt in which the address is forced to an interrupt vector (default 3FF but can be defined by the user) and the current states of the flags and register bank selection are preserved. Please see section on interrupts for more details.
Interrupt_ack	Output	Sync/Async	This output will pulse High for one clock cycle as KCPSM6 starts to service an interrupt by calling the interrupt vector. ‘interrupt_ack’ is generally used by the peripheral logic to cancel the interrupt signal to guarantee that no interrupts are missed and to ensure that each interrupt is only serviced once.
Bit8	Input	Sync/Async	Control bit for data bit width for both receive and transmit functions. When bit8 is a logical ‘1’, then the data width is eight bits, otherwise, the data width is seven bits and data defined by data_in[7] is ignored and data_out[7] is a don`t care.
Parity_en	Input	Sync/Async	Control bit to enable parity for both receive and transmit functions. Parity is enable when the bit is set to logical '1'.
Odd_n_even	Input	Sync/Async	Control bit to define odd or even parity for both receive and transmit functions. When the parity_en control bit is set , a ‘1’ on this

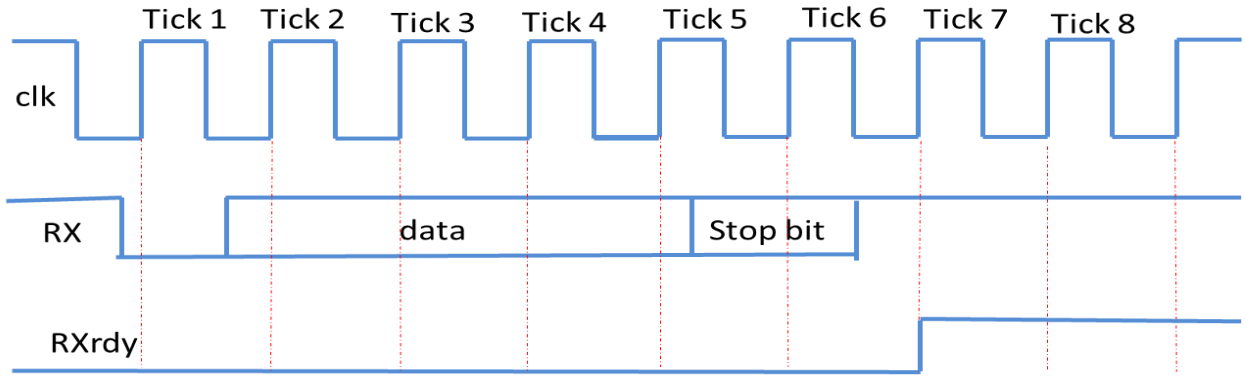
			bit indicates odd parity and ‘0’ indicates even parity
Baud_val	Input	Async	4 bit control bus used to define the baud value.
Rx	Input	Sync/Async	Serial receive data
Tx	Output	Sync/Async	Serial Transmit data.

4.3 Data Flow

Transmit:



Receive:



4.7.1 Clocks

The module works properly with clock of 100MHz.

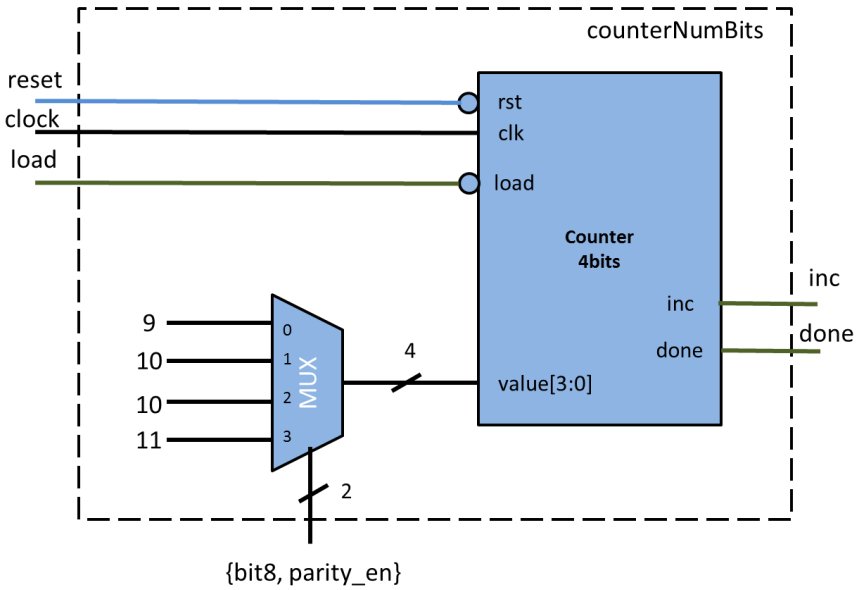
5. Internally Developed Blocks

5.1 lb_counterNumBits

5.1.2 Functional Requirements

The module increments its value every positive edge of the clock that the increment is set. The reset is asynchronous and set up the current value to zero, and the counting value to one of the values: nine, ten or eleven depending of the value of select bits {bit8, parity_en}. If load is deserted at the positive edge of the clock the counting value is updated according to select bits and the current counting value is updated to zero.

5.1.3 Block Diagram



5.1.4 I/O Definition

Input

reset: asynchronous reset**.**
clock: clock provided by the clock source of the domain
Inc: when the signal is one at posedge of the clock, the modules increments my one the current value.
load: when the signal is zero at posedge of the clock the module load one of the four values predefined in the MUX, and set up the current value to zero.
{bits,parity_en}: selects one of the four values predefined in the mux.

Output

done: when the modules reached the counting, the signal is asserted. It is set up to zero if a new increment occurs.

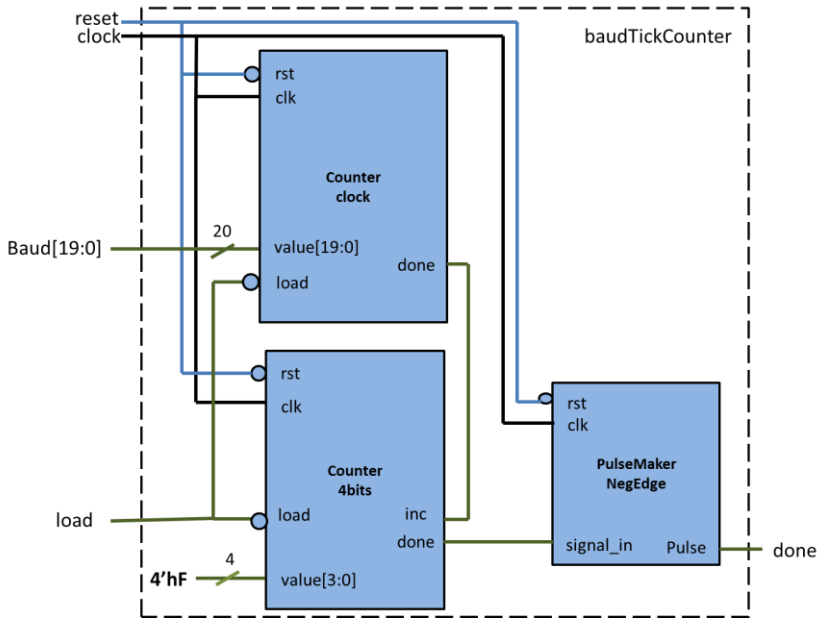
5.1.5 Verification Plan

5.2 lb_baundTickCounter

5.2.1 Functional Requirements

This module counts (16*baud[19:0]) clock ticks and then trigger an 1 clock pulse. The reset is asynchronous and set the current value to zero and load the number of clock ticks it must count from baud[19:0]. The signal load when is clear at the posedge of the clock makes the counter to set the counting to zero and load a the new value of baud[19:0].

5.2.2 Block Diagram



5.2.4 I/O Definition

Input

reset: asynchronous reset**.**
clock: clock provided by the clock source of the domain
baud[19:0]: number of clock that must be counted.

load: when the signal is zero at posedge of the clock the module loads the baud[19:0] value, and set up the current value to zero.

Output

done: the module triggers an 1 clock pulse when the counting is done.

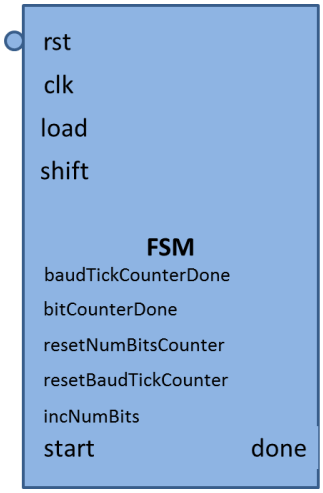
5.2.5 Verification Plan

5.3 lb_fsm_Tx

5.3.1 Functional Requirements

This module is responsible to generate the control signal to the UART Tx module. It is responsible to generate the signals of load, shift, increment of number of bits, reset of baud generator and number of bits counter. The FSM begins the transmition process when start goes to one and just stops when the transmition is done. The set up process is done during the first four clock ticks (section 4.3), so any change in the configuration parameter such as parity, bit8, and baud rate, does not affect the current transmition process.

5.3.2 Block Diagram



5.3.4 I/O Definition

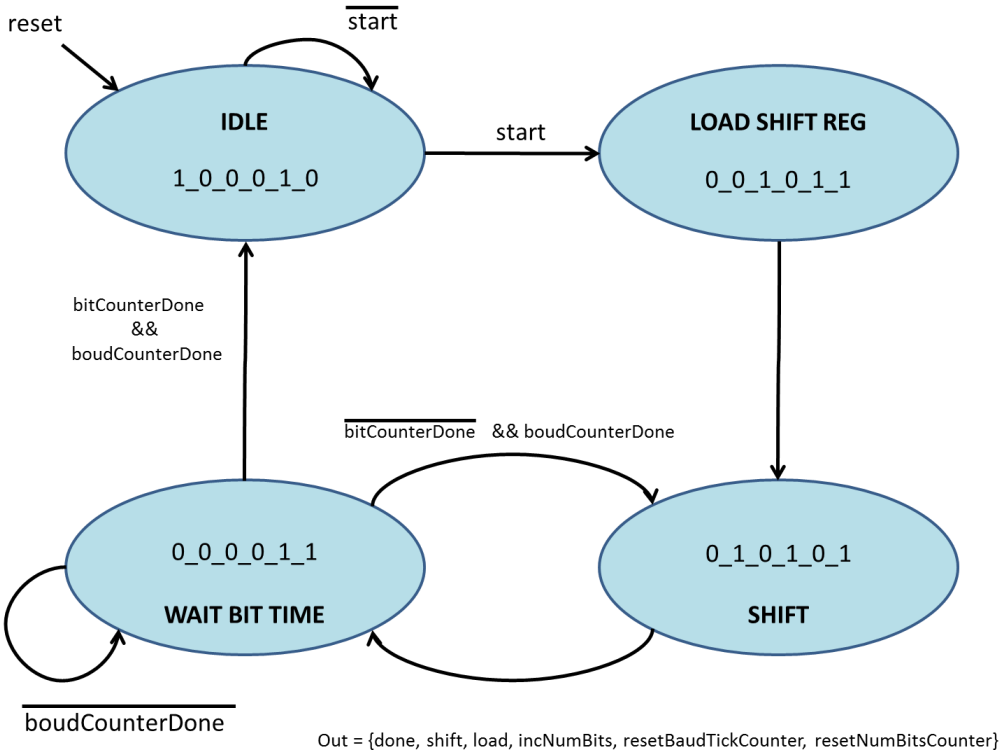
Input

reset: asynchronous reset**,** it sets up the fsm to state IDLE.
clock: clock provided by the clock source of the domain
baudTickCounterDone: signal that indicates if the baud generator is done.
bitCounterDone: Indicates to the FSM if the transmition has finished.
Start: when the signal is one indicates that there is data to be transmitted.

Output

done: this module remains one while there is no data being transmitted.
incNumBits: indicates that the transmition of a bit is done.
resetBaudTickCounter: restart the baud generator to a transmition of a new bit. It is Negedge sensitive.
resetNumBitsCounter: when a new transmition starts this signal is set to zero.
Load: Indicates when the shift register must be loaded.
Shift: Indicates when the data in the shift register must be shifted.

5.3.6 State Machine



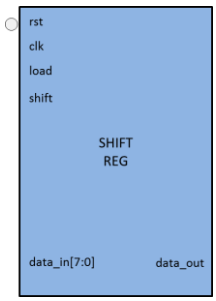
5.3.7 Verification Plan

The test bench was developed to verify if the state machine was having the appropriated state transitions. So, some inputs were generated and we observed if the outputs were following the state diagram.

5.4 lb_11bitShiftRegister

5.4.1 Functional Requirements

5.4.2 Block Diagram



5.4.4 I/O Definition

Input

reset: asynchronous reset**.** It sets up data_out with value 1, and sets up the register with value zero.
clock: clock provided by the clock source of the domain
data_in[10:0]: data that is loaded into the register.
load: when the signal is one at posedge of the clock the module loads the data_in[10:0] value, and sets up the current value to zero. This operation have higher priority than shift operation.
Shift: when this signal is high at the posedge of the clock the module perform a shift right operation and fills the MSB with one.

Output

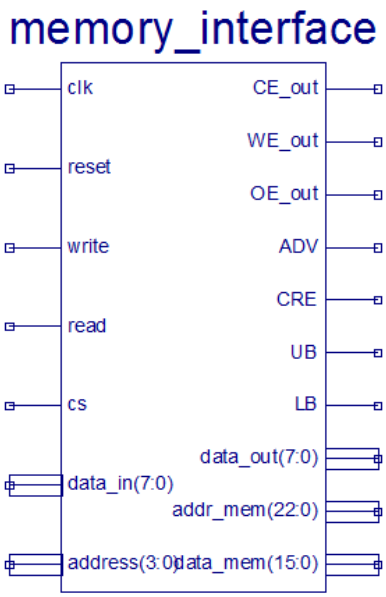
data_out: it is the bit shifted.

5.2.5 Verification Plan

We loaded and performed shift operation until all bit be shifted throughout the data_out

5.5 mem_interface

5.5.2 Block Diagram



5.5.4 I/O Definition

Input

reset: asynchronous reset**,** it sets up the fsm to reset state.
clock: clock provided by the clock source of the domain
write: indicates that data must be written into registers or memory.
Read: indicates that data must be read from registers
Cs: Chip select.
Data_in[7:0] : data provided to be written into internal registers.

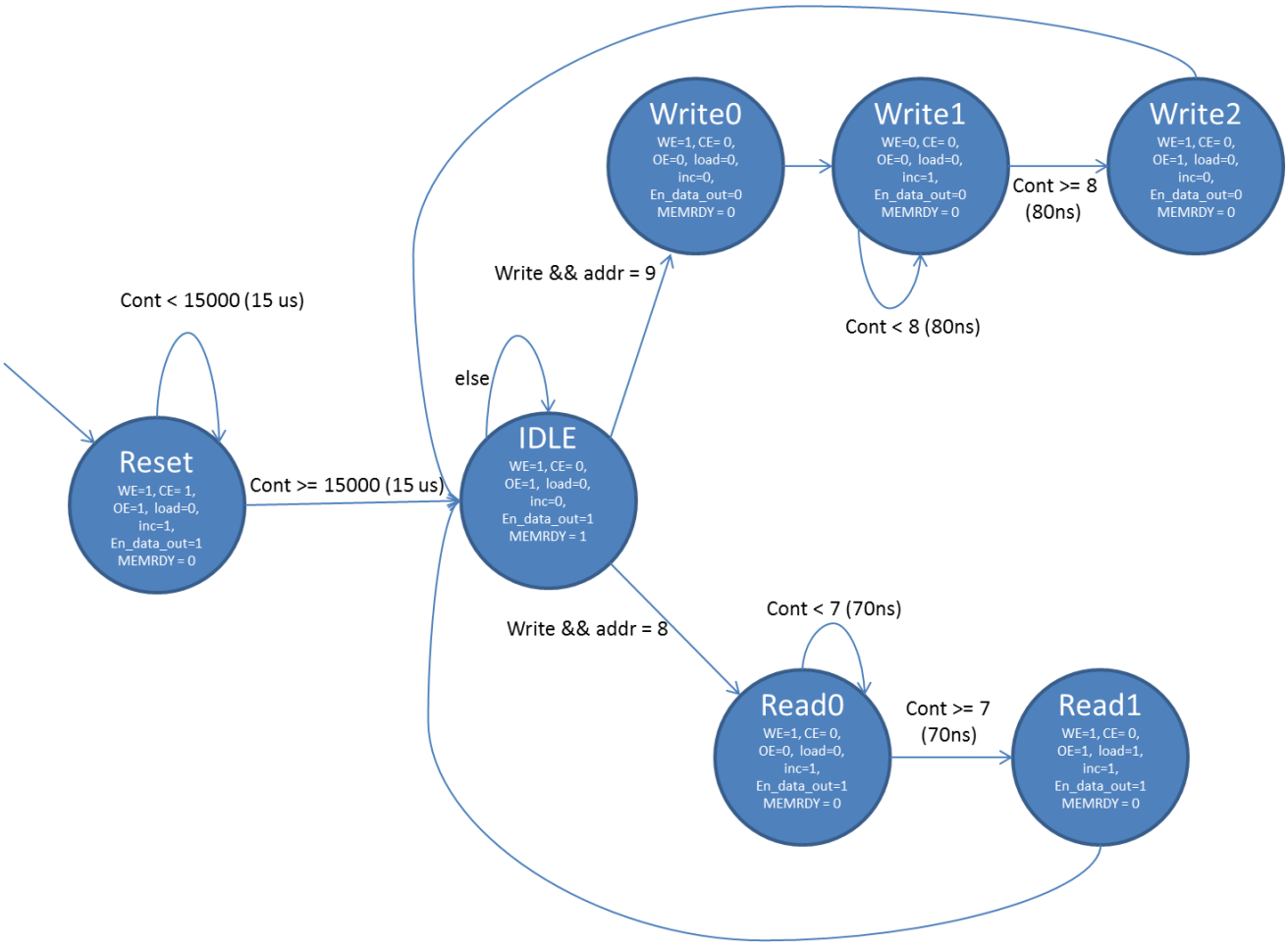
Output

Data_out[7:0] : data which are gonna be read by picoblaz.
CE_out, WE_out, OE_out, ADV, CRE, UB, LB: Memory control signals.
Addr_mem [22:0]: Address which must be read in the memory. The full address is given by {addr2,addr1, addr0}.

Inout

Data_mem[15:0]: memory data bus.

State Machine



Verification Plan:

A bottom up approach was used to test this module. We verified if the state machine was valid than we simulated the interface using the cell ram Verilog module provided by the vendors. Then we created a test block with Processor, memory interface and cell ram block to ensure that reading and writing were working properly, for we developed a simple firmware that was responsible to read and write sequential positions of memory. Then we took off the cellram module and we interfaced the SoC with the physical cell RAM.