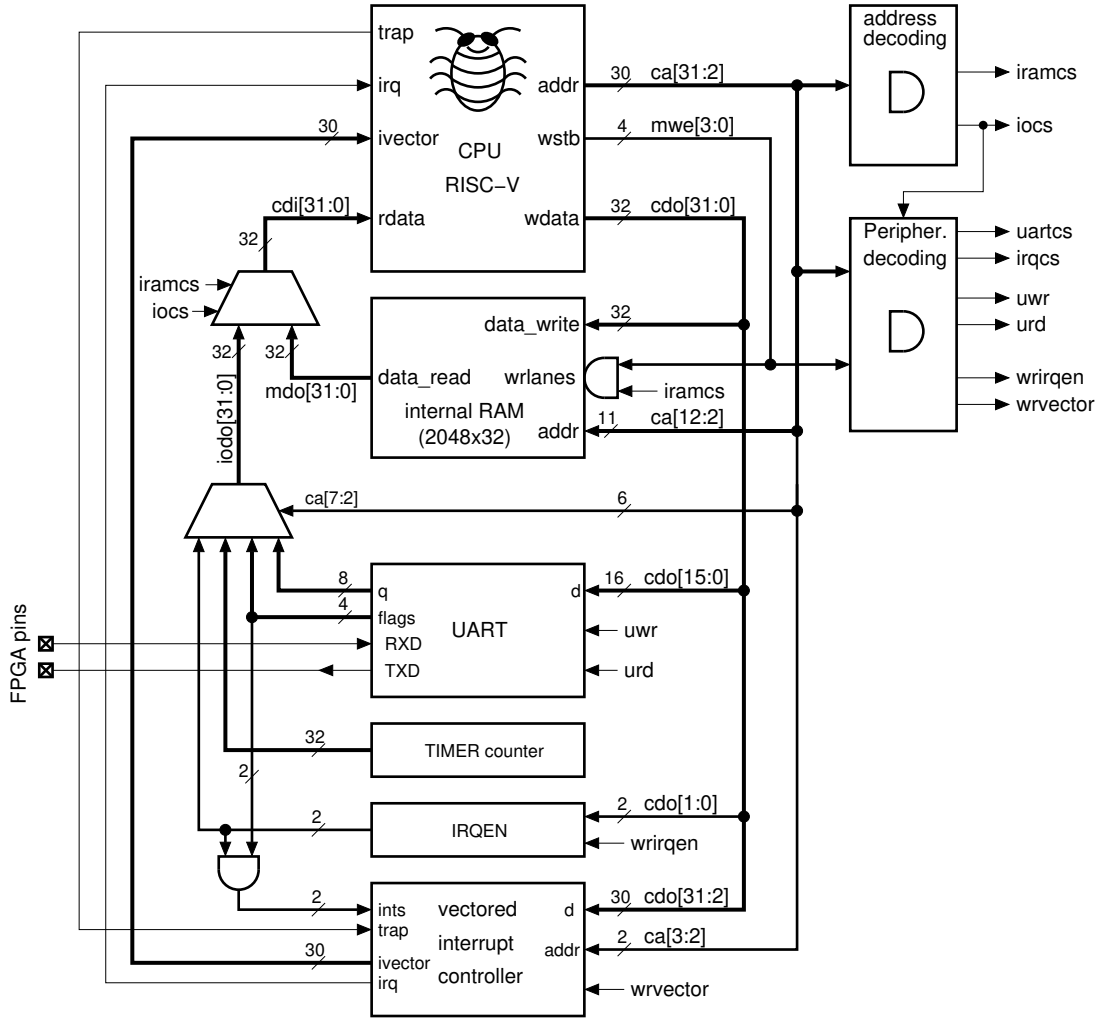
LaRVa's Small System

Jesús Arias

1 System description

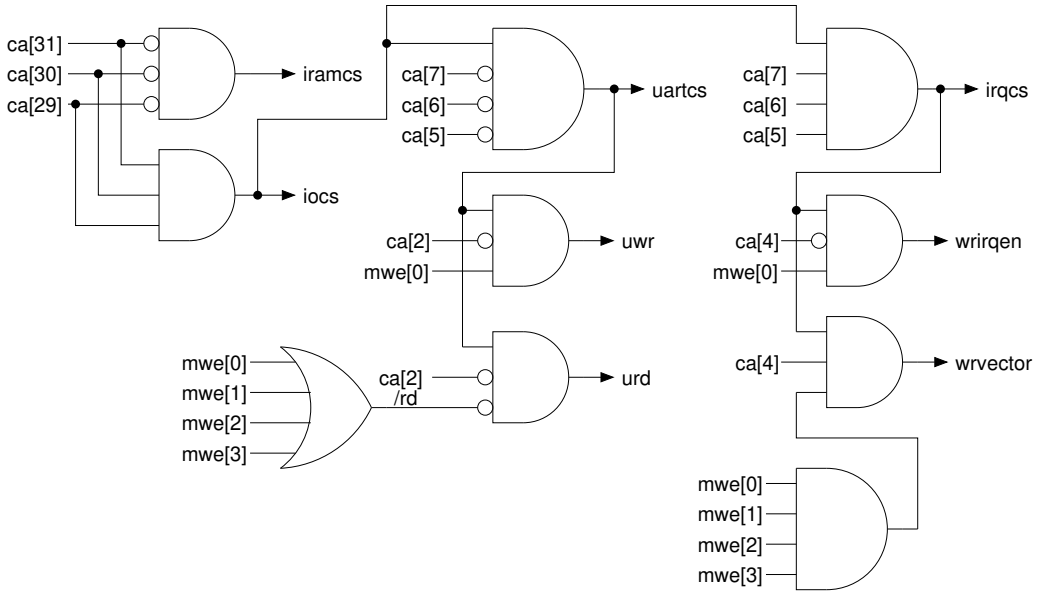


The above figure shows the block diagram of the system, that basically includes a laRVa CPU, 8KB of internal RAM with a known initial content, and an small number of peripherals:

- An Universal Asynchronous Receiver/Transmitter, UART, that provides a bidirectional communication channel with the outside World.
- A free running timer counter for time measurement.
- An interrupt enable register with one enable bit for each interrupt source: Bit 0: UART RX, Bit 1: UART TX.
- A Vectored Interrupt Controller with up to 4 different programmable interrupt vectors. These vectors are presented to the CPU when some interrupt is requested (including the execution of ECALL and EBREAK opcodes)

The RAM and the peripheral registers are mapped to the address space by means of the address decoding blocks (mainly for writings) and multiplexers (for reads).

1.1 Address decoding and memory map



A more detailed schematic of the address decoding logic is shown in the above figure, where "ca[]" are the address lines coming from the CPU and "mwe[]" are the memory write enable lanes for bytes. As data buses are 32 bits wide but the address is for bytes, the two lowest address bits are always 00 and they aren't included in the address bus. The four write lane enable signals are provided instead because we have to select which bytes to write during Store-Byte and Store-Halfword instructions. With this logic the memory map is:

strobe	address bits	R/W	Base address (x=A=0)
iramcs	`000x.xxxx.xxxx.xxxx.xxxA.AAAA.AAAA.AA00`	x	0x00000000
iocs	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xx00`	x	0xE0000000
uartcs	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.xx00`	x	0xE0000000
uwr	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x000`	mwe[0]	0xE0000000
urd	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x000`	rd	0xE0000000
irqcs	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.111x.xx00`	x	0xE00000E0
wrirqen	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.1110.xx00`	mwe[0]	0xE00000E0
wrvector	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.1111.AA00`	mwe[3:0]	0xE00000F0

The above table is mainly for writes, where a write strobe signal is generated for each register to be written. The exception is "urd", a read strobe for the UART that is used to clear some flags (received data valid, for instance). Also notice that the interrupt vectors only allows the Store-Word instruction (32 bit write).

On the other hand, the mapping for reads is performed in the multiplexers for the CPU data input bus, resulting in the following table:

address bits	data read	Base address	Data width
`000x.xxxx.xxxx.xxxx.xxxA.AAAA.AAAA.AA00`	internal RAM	0x00000000	D[31:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.xxxx.xx00`	peripherals	0xE0000000	D[31:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x000`	UART RX (Q)	0xE0000000	D[7:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x100`	UART flags	0xE0000004	D[3:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.011x.xx00`	Timer counter	0xE0000060	D[31:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.111x.xx00`	IRQEN	0xE00000E0	D[1:0]

Notice that, apart from the timer which requires a Load-Word, all the peripherals can also be read with Load-Byte or Load-Halfword instructions.

2 Peripherals

The peripheral address space is located at the upper 512MB area (0xE0000000 to 0xFFFFFFFF) . The included peripherals are:

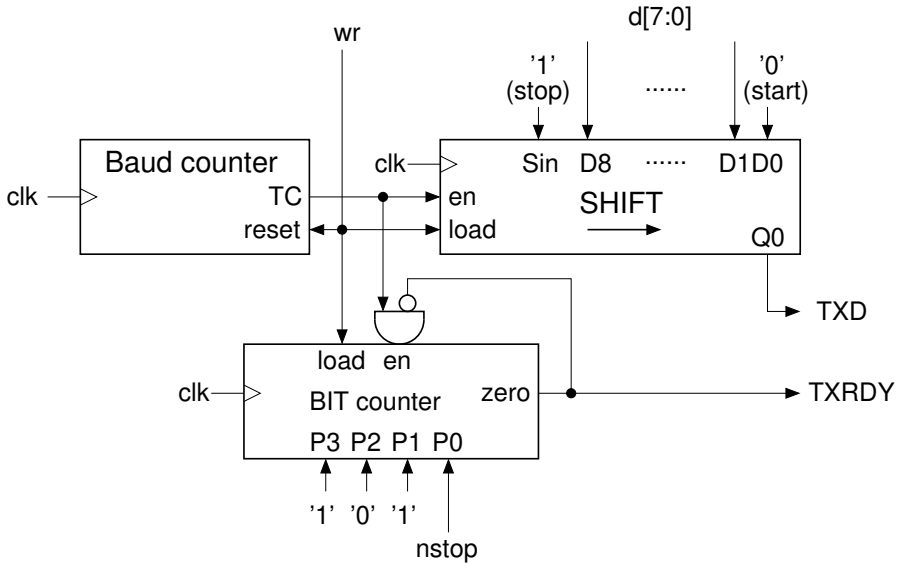
- A simple UART with fixed data format (8-bit, no parity) but now with a programmable baud rate. The UART can request two different interrupts: one for data received and other when ready to transmit.
- A free-running timer.
- A vectored interrupt controller.

2.1 UART

A minimal UART was designed for interfacing. It has the following characteristics:

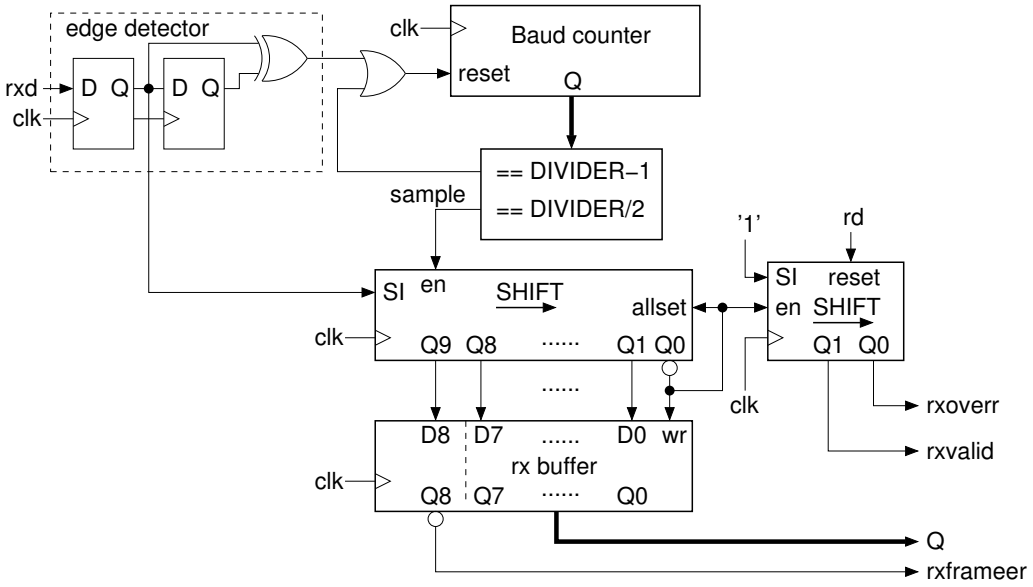
- 8 data bits, No Parity. 1 or 2 Stop bits.
- Fixed baud rate with no 1/16 or 1/8 prescaler. Baud rates as fast as $f_{CLK}/6$ are possible. Baud rate is defined with a clock divider parameter, 'DIVIDER', during synthesis.
- Clock resynchronization on every data transition.
- No buffer register for TX. One byte buffer for RX.

A simplified diagram of the transmitter is shown next. It includes a clock divider, a 9-bit shift register, and a bit counter. The clock divider (baud counter) gets reset when a data is written into the shift register or when it reaches its maximal count, and provides an strobe pulse for data shifting and bit counting. The bit counter is loaded on writes with ten or eleven, depending on the number of Stop bits, and downcounts until it reaches zero. When in zero state the tx_ready flag is asserted. This is all the logic needed for a functional UART transmitter.



The diagram of the receiver is shown next. It also includes a baud counter that get reset when it reaches its maximal count (DIVIDER-1) or when an edge is detected in the incoming data stream. Also, a sampling strobe is asserted at the middle of the bit time, enabling the shifting of the incoming bits. The shift register has

10 bits and when the start bit of the data arrives at Q0 all its bits are preset as ones while the rest of the register is copied to the output buffer, including the received Stop bit. A value of zero in that bit means a framing error. Another two flags are also present: an 'rxvalid' flag is asserted every time a data is stored in the buffer, and a 'rxoverr' flag can also be set if 'rxvalid' is already active when data is stored. These two flags are reset by means of an external read strobe, 'rd'.



The UART module has all its flags available as individual outputs. These signals were grouped into an status register at address 0xE0000004 in the following way:

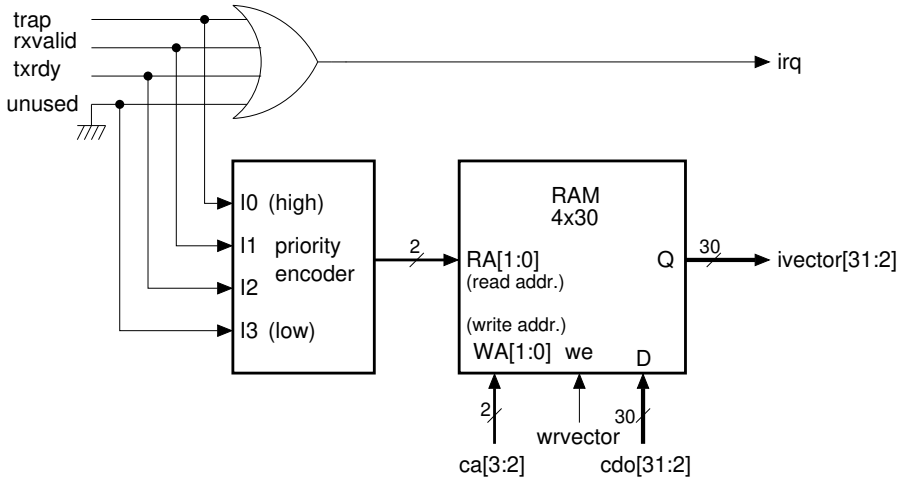
Register	...	bit 3	bit 2	bit 1	bit 0
STATUS (read)	0	RX overrun	RX Framing error	TX ready	RX valid

On the other hand, while the UART core can transmint one or two stop bits, in this system its 'nstop' control input is fixed as '0', so, only one stop bit is transmitted.

2.2 Timer

The timer is simply a 32 bit counter that gets incremented each clock cycle. Its value can be read at address 0xE0000060. This is a read-only peripheral.

3 Vectored Interrupts



The diagram of the simple Vectored Interrupt Controller is shown in the above figure. The system has 3 possible interrupt sources, each of them with an specific interrupt vector (write-only) where the memory address of the corresponding interrupt service routine is stored. There is also an interrupt enable register that allows the selective masking of interrupts:

Interrupt Cause	INTEN bit mask (0=masked)	Vector #	Priority	Vector address
ECALL, EBREAK	non-maskable	0	Highest	0xE00000F0
UART RX valid	bit #0	1		0xE00000F4
UART TX ready	bit #1	2	Lowest	0xE00000F8

The software interrupts (instructions ECALL and EBREAK) can't be masked. These are always serviced.

4 Summary

The system presented here was synthesized for an ICE40HX4K FPGA with the following results:

Logic cells	2813
BRAMs	16
Max. clock frequency	22 MHz