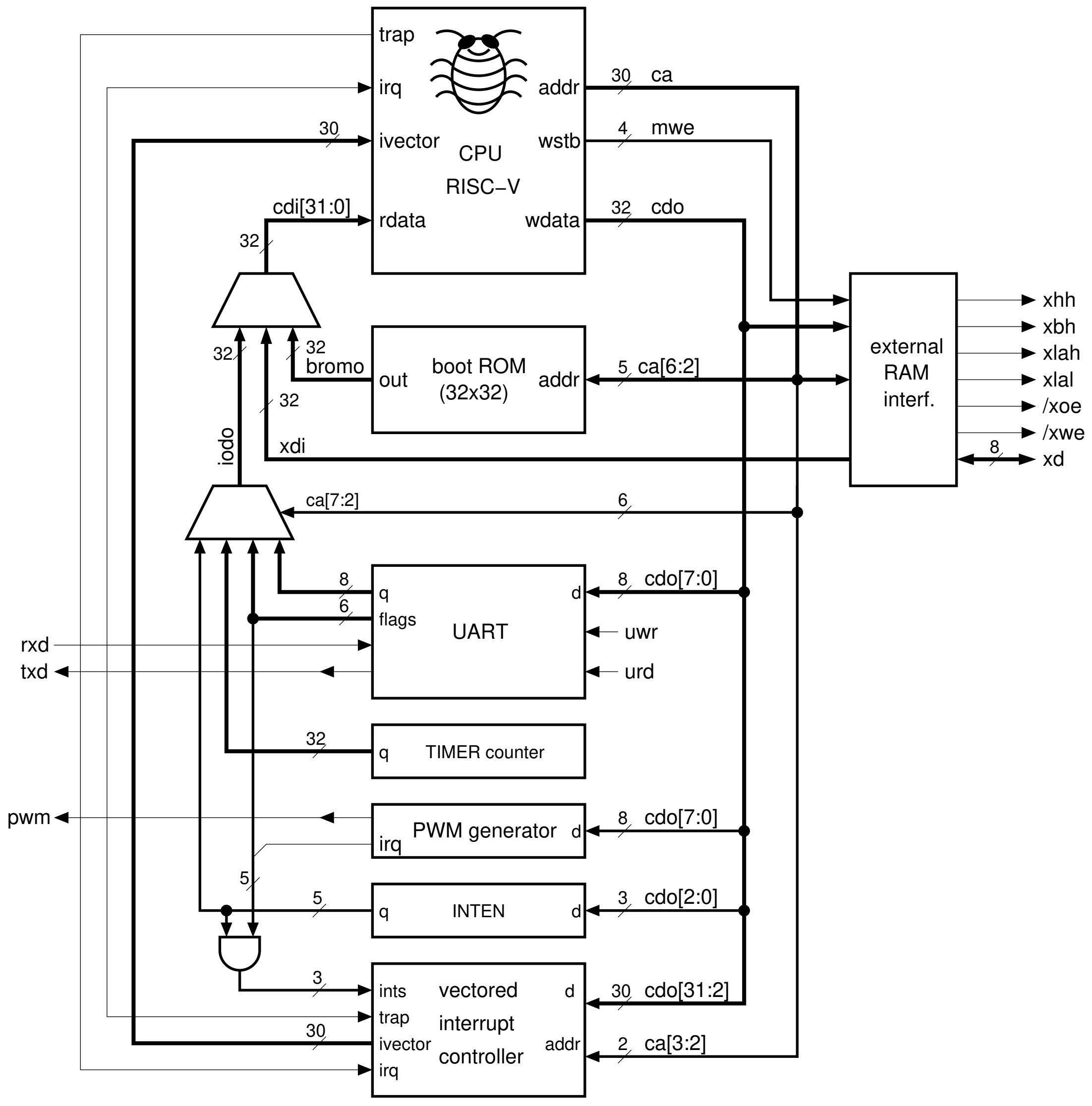
LaRVa's Tiny Tapeout prototype

Jesús Arias

1 System description



The above figure shows the block diagram of the system, that basically includes a laRVa CPU, a tiny boot ROM, an external memory interface, 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.
- A PWM generator.
- An interrupt enable register with one enable bit for each interrupt source.

- 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 memories and the peripheral registers are mapped to the address space by means of some address decoding logic (mainly for writings) and multiplexers (for reads).

1.1 Memory map

In the RiscV processor data buses are 32 bits wide but addresses are for bytes, thus, the two lowest address bits are always 00 and they aren't included in the address bus. Four write lane enable signals, "mwe[3:0]", are provided instead because we have to select which bytes to write during Store-Byte and Store-Halfword instructions. The memory map for writes is the following:

strobe	address bits	byte lane	Base address (x=A=0)
RAM	`001x.xxxx.xxxx.xxAA.AAAA.AAAA.AAAA.AA--`	1,2,or 4 lanes	0x20000000
UARTwr	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x0--`	xxx1	0xE0000000
UARTrd	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x0--`	0000	0xE0000000
clrdirty	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x1--`	xxx1	0xE0000004
PWM	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.001x.xx--`	xxx1	0xE0000020
INTEN	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.1110.xx--`	xxx1	0xE00000E0
vector	`111x.xxxx.xxxx.xxxx.xxxx.xxxx.1111.AA--`	1111	0xE00000F0

The above table is mainly for writes, where a write strobe signal is generated for each register to be written. The exception is "UARTrd", a read strobe for the UART that is used to clear some flags (received data valid, for instance), so it also writes something. Also notice that the interrupt vectors only allows the Store-Word instruction (32 bit write). And, finally, the "clrdirty" strobe is used to clear a debug flip-flop on writes to the 0xE0000004 address (the data written is ignored).

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

address bits	data read	Base address	Data width
`000x.xxxx.xxxx.xxxx.xxxx.xxxx.xAAA.AA--`	boot ROM	0x00000000	D[31:0]
`001x.xxxx.xxxx.xxAA.AAAA.AAAA.AAAA.AA--`	external RAM	0x20000000	D[31:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x0--`	UART RX	0xE0000000	D[7:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.000x.x1--`	PFLAGS	0xE0000004	D[4:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.011x.xx--`	Timer counter	0xE0000060	D[31:0]
`111x.xxxx.xxxx.xxxx.xxxx.xxxx.111x.xx--`	INTEN	0xE00000E0	D[2:0]

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 External memory

The LaRVa core is a 32-bit processor, meaning its address and data buses are 32-bit wide, but the number of available pins for Tiny Tapeout chips is much less than that. Therefore the external memory interface is designed to pass all address and data through 8 bidirectional pins, and this requires 6 clock cycles (see figure 1). First, the external RAM address is stored into two external 8-bit latches, controlled by "xlal" and "xlah".

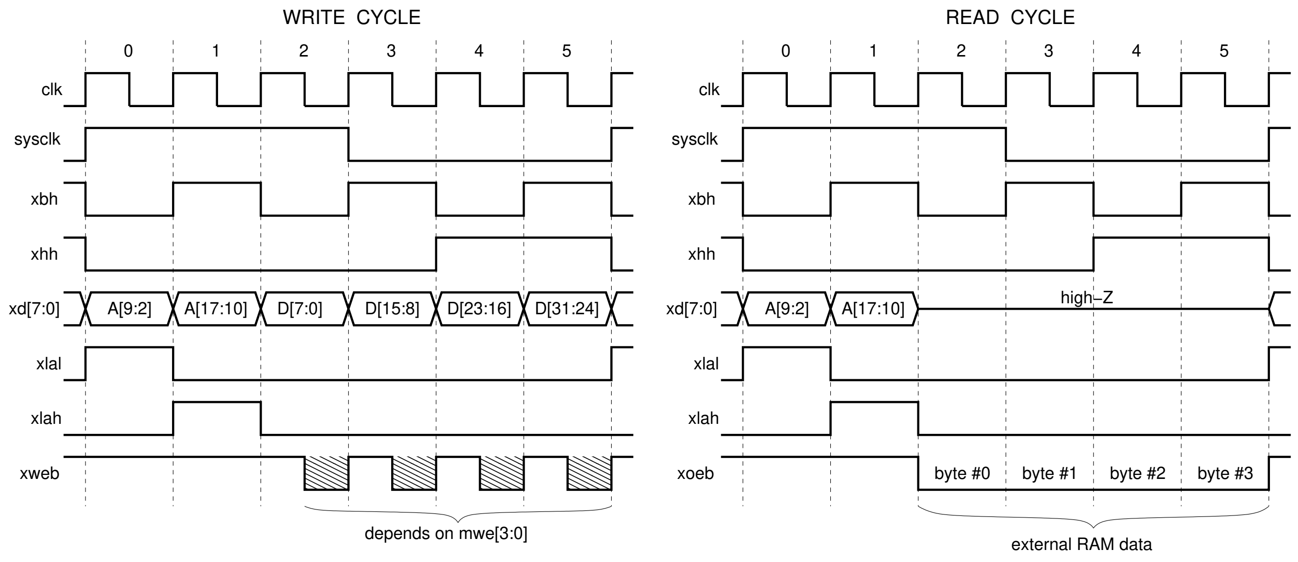


Figure 1: External memory timings

This allows for a maximum 256 KB of external RAM. And, then, another 4 cycles are used for data transfers, totaling 32 data bits. The "xbh" (eXternal Byte High) and "xhh" (eXternal Halfword High) signals can be used as the two lower address signals for the external memory.

Then for write cycles, the byte-write lanes, "mwe[3:0]" are presented as write pulses at the proper time in order to write only the intended bytes into the external RAM.

For read cycles three temporary 8-bit registers are used in order to store the 24 LSBs of the data before presenting a complete 32-bit data to the CPU input data bus on the last clock cycle.

The core clock is thus 1/6 of the main clock frequency (in our case a 24 MHz clock will result in a 4 MHz instruction cycle)

3 Boot ROM

The boot ROM code is executed after reset and it implements a simple serial bootloader. This code is further simplified thanks to the UART blocking data registers. This means that a read of the RX data register actually stops the CPU clock until a character is received, and therefore no polling loops are required in the boot ROM. The bootloader first wait until an 'L' character is received, then reads three 32-bit values (LSB first) with the number of bytes to be loaded, the loading memory address for the data, and an entry address for the execution of the loaded code. This results in the following image format:

#nbytes	Data (LSB first)	Comments
4	0x4CFFFFFF	0xFF is a byte-sync character for UART RX
4	Loading address	
4	Size (bytes)	not including this header
4	Entry address	0x0000 means enter bootloader again
Size	Code to load	

The first 0xFF bytes aren't really needed, but they give a multiple of 4 size for the header and ease the correct reception of the following data as none of their data bits could be misunderstood as an start bit. Also, the 'Entry Address' word can be made zero and the bootloader jumps to itself after loading a data block. In this way more than a single data block can be loaded before jumping to the program code.

4 Peripherals

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

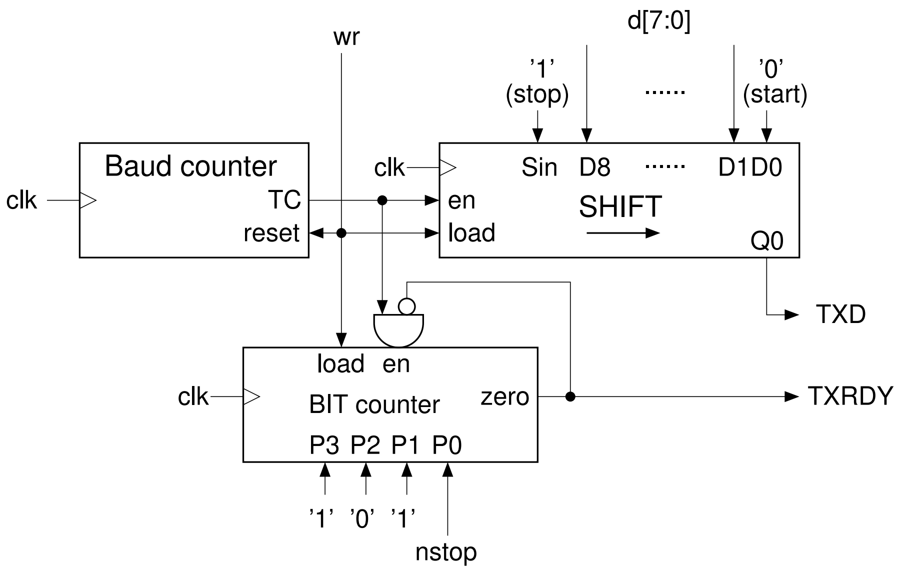
- A simple UART with fixed data format (8-bit, no parity) and speed (115200 bps). The UART can request two different interrupts: one for data received and other when ready to transmit. Its data registers have blocking reads and writes: The CPU clock is stopped if there is no data to read in the receiver or if we write to the transmitter while busy.
- A free-running timer. Mainly intended for time measurement.
- An 7.5-bit (182 different levels) PWM generator with a sampling rate about 22.05 kHz.
- A vectored interrupt controller with 4 vectors (one trap for ECALL / EBREAK, and 3 interrupt sources)

4.1 UART

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

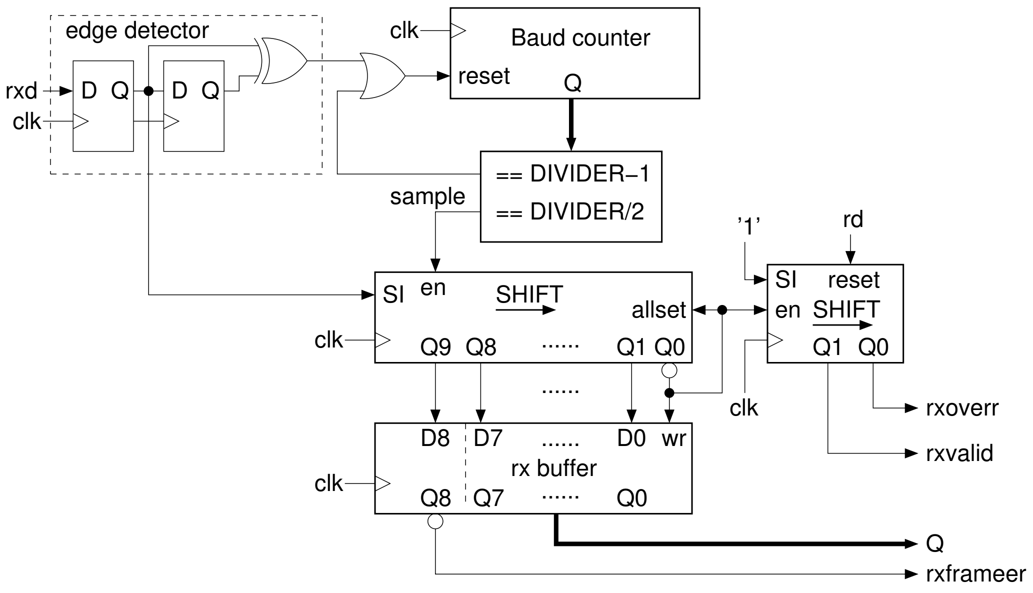
- 8 data bits, No Parity. 1 or 2 Stop bits.
- Fixed baud rate with no 1/16 prescaler. Baud rates as fast as $f_{CLK}/6$ are possible. Baud rate is defined with a clock divider parameter, 'DIVIDER', during synthesis.
- Clock resynchronization on every data transition.
- No buffer register for TX. One 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 the PFLAGS register at address 0xE0000004 in the following way:

Register	...	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PFLAGS	...	PWM irq	DIRTY	CTRLC	RX overrun	RX Framing error	TX ready	RX valid

Bits #0 to #3 are the UART flags. Bits #4 and #5 are two flags intended for debugging: 'CTRLC' gets set when the UART receiver has a 'control-C' character (0x03) in its buffer and 'RX valid' is set. This allows to detect a break condition without actually reading the UART's data register. 'DIRTY' is a flip-flop that gets set every time a data is written to the UART transmitter, and cleared when writing any data to the address of the PFLAG register. This flag can tell to a debugger program is the user code has written something to the terminal since the last breakpoint.

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

The read and write accesses to the receiver and transmitter data registers can stop the CPU clock and this feature is exploited in the bootloader code. But the usual polling routines won't result in clock stalls, that is also desirable if there are interrupts enabled.

4.2 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.

4.3 PWM

A PWM modulator is included with the intended use of audio generation. Its diagram is shown in figure 2, where the main building block is a binary counter that increments from zero up to 180 (its modulus is 181). The counter value is compared with the PWM level stored in a holding register, and, when the two values

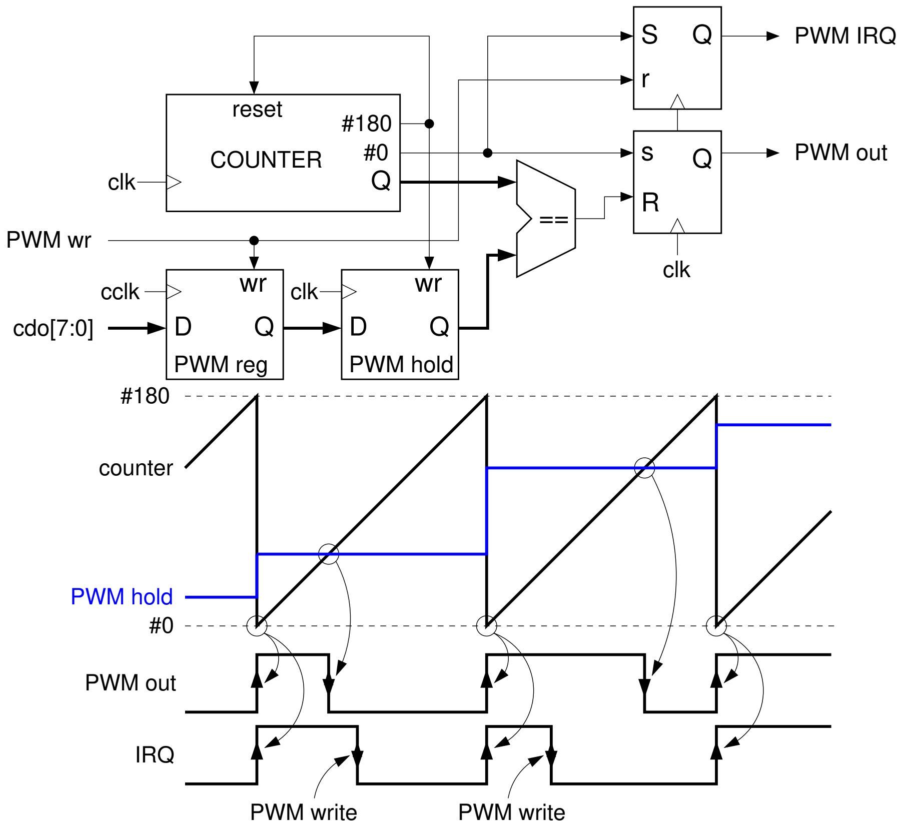


Figure 2: PWM modulator and related waveforms

match the output flip-flop gets reset. This flip-flop is also set when the counter value is zero, thus generating the expected PWM waveform at the output. There are two special cases to analyze:

PWM level = 0. In this case the output flip-flop gets reset (its reset input has more priority than set) and the PWM output is always low.

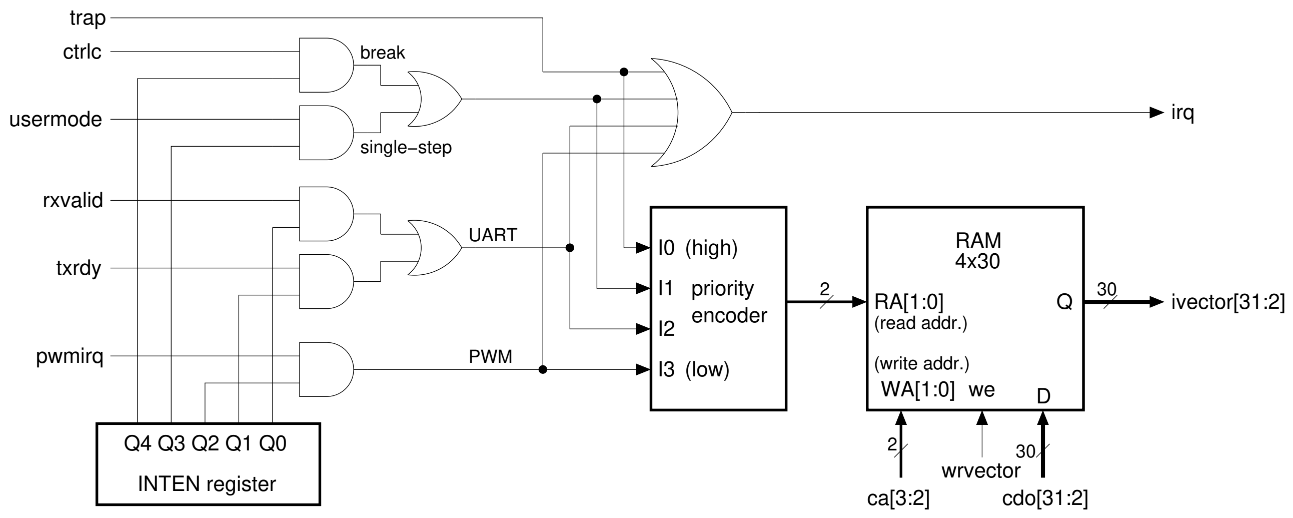
PWM level $\geq$ 181. In this case the flip-flop is set but never reset, and the output is always high.

A second flip-flop is used as an interrupt flag. It gets set when the counter equals zero and reset when a new value is stored in the PWM register. This flag can request an interrupt to the core when high and it can also be read at the bit #6 of the 'STATUS' register..

And, finally, there is a holding register between the PWM register and the comparator block. Thanks to this register the output waveform is undisturbed when new PWM values are written. These values are copied to the holding register when the counter changes from its maximum count to zero. The PWM modulator is a write-only peripheral.

The PWM carrier frequency is 4 MHz / 181 = 22099 Hz (only a 0.22% above the usual 22050 Hz sampling rate), and the modulation resolution is $\log_2(182)$ = 7.49 bits.

5 Vectored Interrupts



The diagram of the simple Vectored Interrupt Controller is shown in the above figure. The system has 5 possible interrupt sources (not counting traps) and 4 interrupt vectors (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
Control-C	bit #4	1		0xE00000F4
Single-Step	bit #3			
UART RX valid	bit #0	2		0xE00000F8
UART TX ready	bit #1			
new PWM cycle	bit #2	3	Lowest	0xE00000FC

The software interrupts (instructions ECALL and EBREAK) can't be masked. These are always serviced. Also notice the two interrupt sources from the UART share a common interrupt vector, and the same happens with the 'Control-C' and 'Single-step" interrupts. The Single-step interrupt is entered as soon as the bit #3 of INTENT is set, but on returning from its ISR the 2-cycle interrupt latency allows a single instruction to be executed in user mode before jumping again to the vector #1 ISR. 'Control-C' can also stop an user program and redirect the execution to a debugger code

6 Summary

The system presented here was synthesized for an ICE40HX4K FPGA, along with the required external memory interface, and a JTAG debug port, resulting in 3223 logic cells (and 0 BRAMs).

When synthesized for the 130nm CMOS technology of the Tiny Tapeout chips it required 6 tiles ($480 \times 220\ \mu m^2$)