Open Hardware USB E1 interface (WIP)

Sylvain Munaut

OsmoCon, October, 2018

Introduction

E1/T1/TDM

- Good old ISDN technology
- 2 Mbits/s (E1) or 1.54 Mbits/s (T1) synchronous, full-duplex
  - Focus on E1 here
- Not used much anymore in telephony (everything moves to SIP/IP)
- Still used quite a bit in 2G/3G cellular networks, even in 2018!

E1 in 2G/3G networks today

Traditionally all interfaces over E1/T1

- Abis (RSL/OML over LAPDm) from BTS to BSC
  - Back-haul networks increasingly switch TDM to IP as 4G is co-located with 2G
  - BUT: Lots of BTSs still have physical E1
- A (BSSAP/SCCP/MTP) from BSC to MSC
  - TDM on the decline, moved over to 3GPP AoIP
- Core network
  - TDM based core network connections still prevalent
  - Lots of legacy switches (MSCs) and STPs around

E1 interface: Use cases

- Many E1/T1 based BTSs decommissioned around the world
- Refurbished traders have quantities in stock for **very** low price
- Using those BTSs with OsmoBSC + friends is an inexpensive way of
  - Deploying carrier-grade tier-1 BTS equipment
  - With excellent environmental, RF sensitiviy, RF power and high MTBF
  - For very low cost

E1 interface: Existing options

- PCI / PCIe cards available
- Still fairly expensive
  - Fine for use un the Core network
  - Not so much next to each BTS
- Requires fairly large computers to get full size PCI/PCIe slots

E1 interface: Building our own

- Wishlist
  - In 2018, you just want a very small E1/USB or E1/Ethernet adapter
  - Can be used with laptop for on the road debugging
  - Can be used with off-the-shelf cheap/small SBCs
  - Stable/Precise clock for the BTS to use as reference
  - Cheap-ish
- Existing chips
  - Hard to find, many already EOL, the rest to follow
  - Impractical bus interfaces (large parallel bus)
  - Impractical high pin count packages
  - Unreasonably expensive
- "Software Defined" TDM
  - Lower level LIU chips (still 'ACTIVE' status) for electrical interface
  - Serialize / Deserialize from some microcontroller
  - Implement the rest in software (either host or mcu firmware)

Harware options: Initial thoughts

- TI PRU (Programable Realtime Unit)
  - TI processors like the AM335x on the Beagleboard have PRU cores
  - PRU allows high-speed "real time" bit banging and provides buffers to the ARM
  - Beaglebone could actually run entire Osmo stack
- XMOS
  - RISC CPU core @ 500 MHz with programmable serdes
  - USB + Ethenet as "Hard IP"
  - Everything else (SPI, I2C, ...) implement in software
- Programmable Logic
  - Not so "software" defined
  - Overkill for a 2Mbit/s signal
  - Toolchains can be an issue

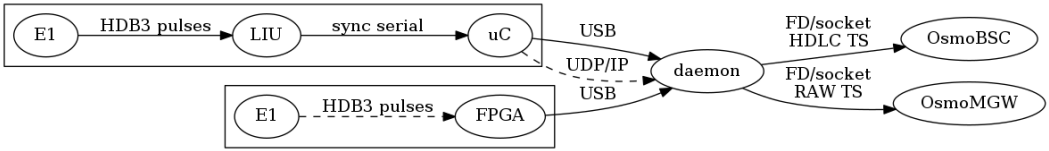
Harware options: What we went with

- Atmel SAM4S + IDT 82V2081
  - Probably the smart option here
- ice40 UltraPlus
  - Skip the LIU
  - TBH ... I mostly thought it'd be fun

Software Interface

Software Interface: Host

- Existing support for E1/T1 in `libosmo-abis`
  - Supports mISDN & DAHDI drivers
  - One `fd` per timeslot concept
- Use a daemon to handle USB (or ethernet) communication with the interface
  - Handles hardware / timeslot muxing / HLDC / ...
  - Exposes one fd per time-slot to external processes
  - Important to share the device between different processes (OsmoBSC for signalling and OsmoMGW for traffic)
  - RPC method ? Buffer sizes ?
- No work done as of yet



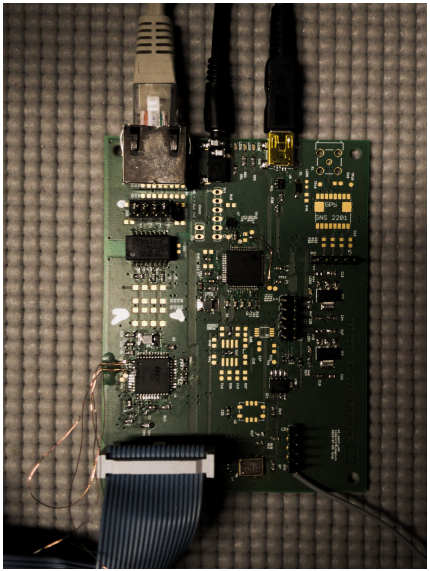
Software Interface: USB

- USB protocol
  - Vendor-specific USB device/interface for control/status
  - Isochronous IN and OUT endpoints for traffic
  - Exchanges multiples full 32-byte octet-aligned E1 "basic frames"
  - Flow control ?
  - Exact specs still need to be defined

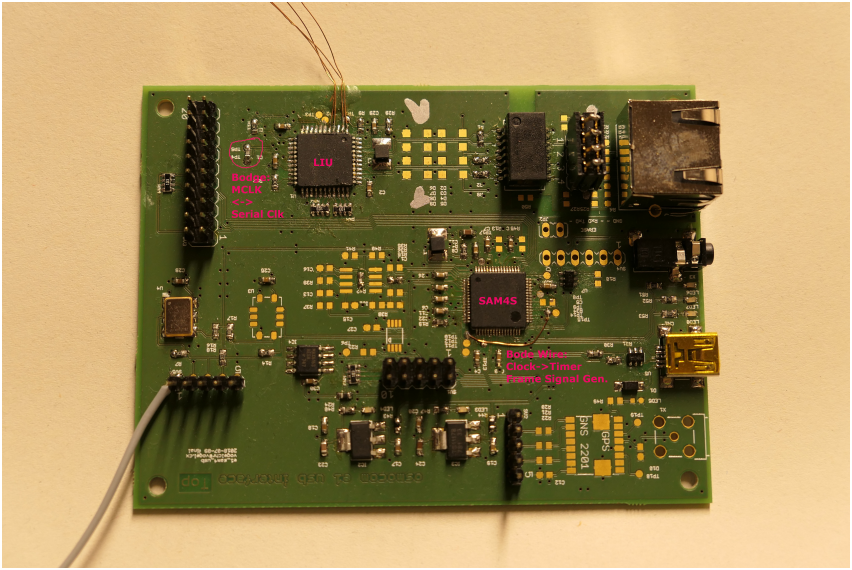
Hardware: SAM4S + LIU

SAM4S: Overview

- Work led by Christian Vogel
- LIU: IDT 82V2081
  - Already had a board to experiment with that LIU
  - Alternative considered, but no real gain
- uC: Atmel SAM4S
  - Hardware USB Full-Speed
  - SSC controller to interface the LIU



SAM4S: Dev Board



SAM4S: Clocking

- 30.72 MHz VCXTO directly clocking the SAM4S
  - Common frequency
  - Divide by 8, Multiply by 25 -¿ 96 MHz for the USB
  - Divide by 15 gives 2.048 MHz for the LIU
- Means no USB SAM-BA :(
  - UART it is then
- Classic GPSDO FLL
  - Count clock cycles between 2 PPS
  - Integrate error to adjust a DAC

SAM4S: E1 interface

- IDT 82V2081
  - Handles electrical interface
  - Equalization, clock recovery, HDB3 coding, ...
  - Simple Clk/Data interface for TX & RX
- Master clock provided by SAM4S
  - Therefore derived from 30.72M locked to GPS
- Interfaced via SAM4S SSC
- TX:
  - Easy, just send the bits. Receiver has to align.
- RX:
  - Search for proper alignement in the raw bit stream
  - Use the Timer/Counter unit to generate a framing signal
  - Once aligned, no need to do bit-shuffling in software

SAM4S: USB

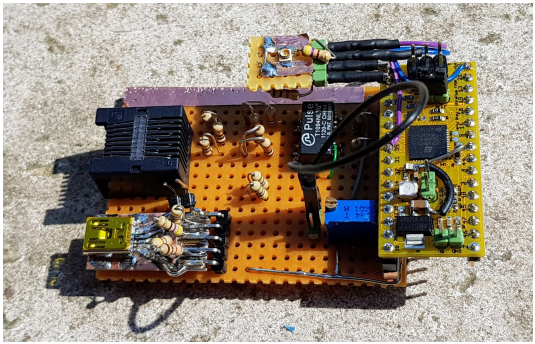
- SAM4S has Hardware USB Full Speed SIU
- Still requires software "drivers"
- Not supported in libopencm3
- Vendor stack is a mess
- Rewrite our own
  - WIP: Enumeration working

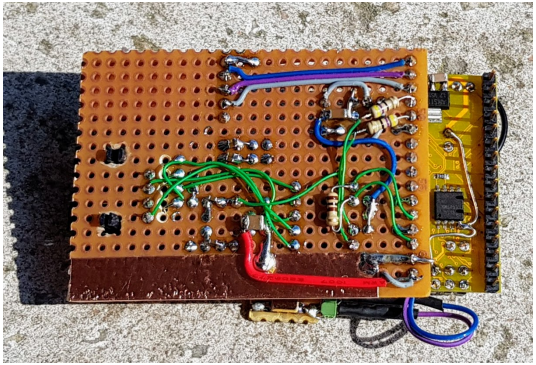
Hardware: ice40

ice40: Overview

- Small FPGA by lattice
  - Cheap
  - Convenient (QFN48) package
  - FOSS toolchain available
  - Mostly I thought it'd be fun
- Goal is to minimize chip count

ice40: Dev Board





ice40: Clocking

- Same idea as in SAM4S
- 30.72 MHz VCTXO
  - Use FPGA PLL to go to 48M for USB
  - Use FPGA logic to divide by 15 for nominal baud rate
- DAC
  - PDM from FPGA pins
  - If too much noise, will use a "real" DAC

ice40: E1 interface

- Electrical interface
  - Avoid LIU
  - Abuse ice40 differential IO as comparator to detect analog voltages
  - Could also avoid magnetics with capacitive coupling
  - ... but it's a bit finicky, not worth it
- E1 HDB3/Framing/Sync in FPGA logic
  - RX all done and working
  - TX mostly done, still need framer, but that's trivial

ice40: USB interface

- Electrical interface
  - For Full Speed, classic CMOS drivers are fine
- Implement a SIU in FPGA logic
- Control with a high level soft core
  - RISC-V most likely
- Very much WIP ...

Final words

Resources

- Project: `http://osmocom.org/projects/e1-t1-adapter`
  - including links to all relevant specs :
  - `http://osmocom.org/projects/e1-t1-adapter/wiki/E1_Specifications`
- Hardware:
  - `http://git.osmocom.org/osmo-e1-xcvr`
  - `https://github.com/vogelchr/e1_sam4_usb`
  - `https://github.com/vogelchr/e1_sam4_usb_fw`

Thanks

- Christian "sigwinch" Vogel
- Harald "LaF0rge" Welte
- Dieter Spaar