WRPC integration into WR Node

White Rabbit
COLLABORATION

Training material

2024/12/4

Main Topics

What you will learn:

- What should be added on the PCB for a WR node
  - Oscillators
  - SFP
  - eeprom for parameters and MAC address
- Clocking structure
- How to instantiate the WR core
- Troubleshooting

Vocabulary

WR: White Rabbit – the fully deterministic Ethernet-based network for general purpose data transfer and synchronization.

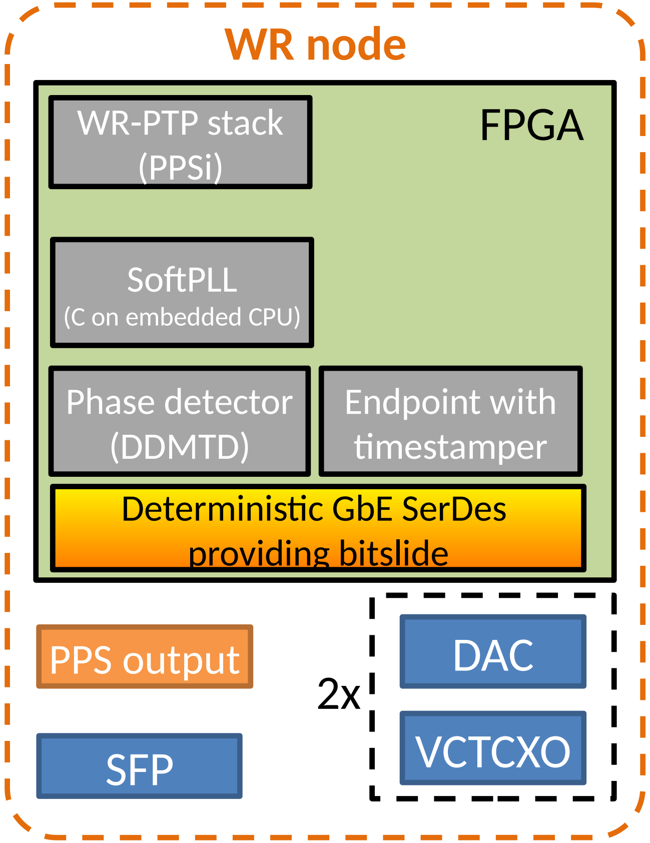
WRPC: WR PTP Core – HDL + SW IP implementing WR. In general, it’s a slave (the WR switch is master/grandmaster/boundary clock). Sometimes we use the redundant name ‘WRPC core’

WR Node: A device which includes the WRPC core, as opposed to the WR switch

White Rabbit PTP Core (WRPC)

Main HDL components:

- 1G Ethernet endpoint (with timestamps)
- DDMTD to extract phase between clocks
- Timetag + PPS generation
- GPIOs for storage, LEDs
- UART
- CPU + ram for control



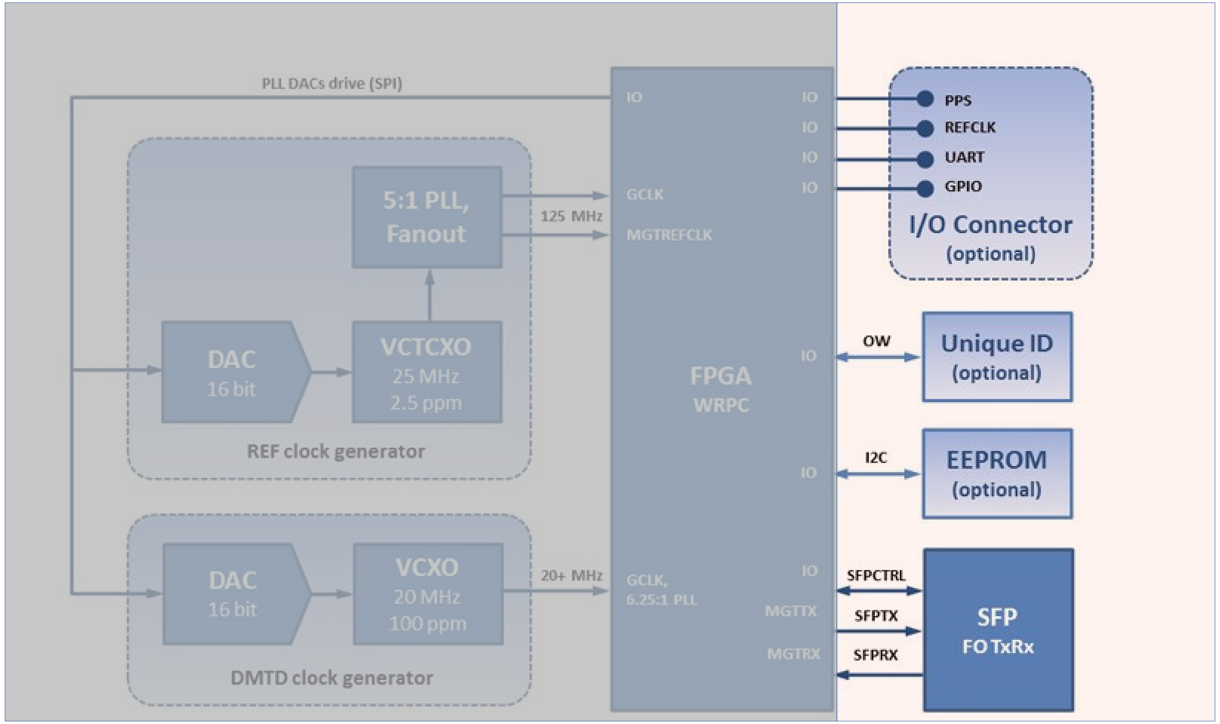
WR node
FPGA
WR-PTP stack (PPSi)
SoftPLL (C on embedded CPU)
Phase detector (DDMTD)
Endpoint with timestamper
Deterministic GbE SerDes providing bitslide
PPS output
SFP
2x
DAC
VCTCXO

White Rabbit PTP Core (WRPC)

Main software components:

- SoftPLL to discipline clocks (topic for another talk)
- PTP (ppsi) for delay
- Remote management (BOOTP, ARP, SNMP, LLDP, …)
- Local storage (MAC address, calibration parameters)
- CLI

Non-clocking external hardware



Storage

The core needs non-volatile storage (fallback if absent):

- Calibration parameters
- MAC address
- SFP database

This is fully handled by software, and quite flexible:

- SPI flash
- I2C eeprom
- One wire

Storage

For MAC address, unique id chip like 24AA02E48 could be used

For non-SoC FPGA, the bitstream flash could also be used as non-volatile storage (different page!)

SPI and I2C are done in software (bit-banging)

Misc

SFP cage

- Follow vendor recommendations
- AC capacitors

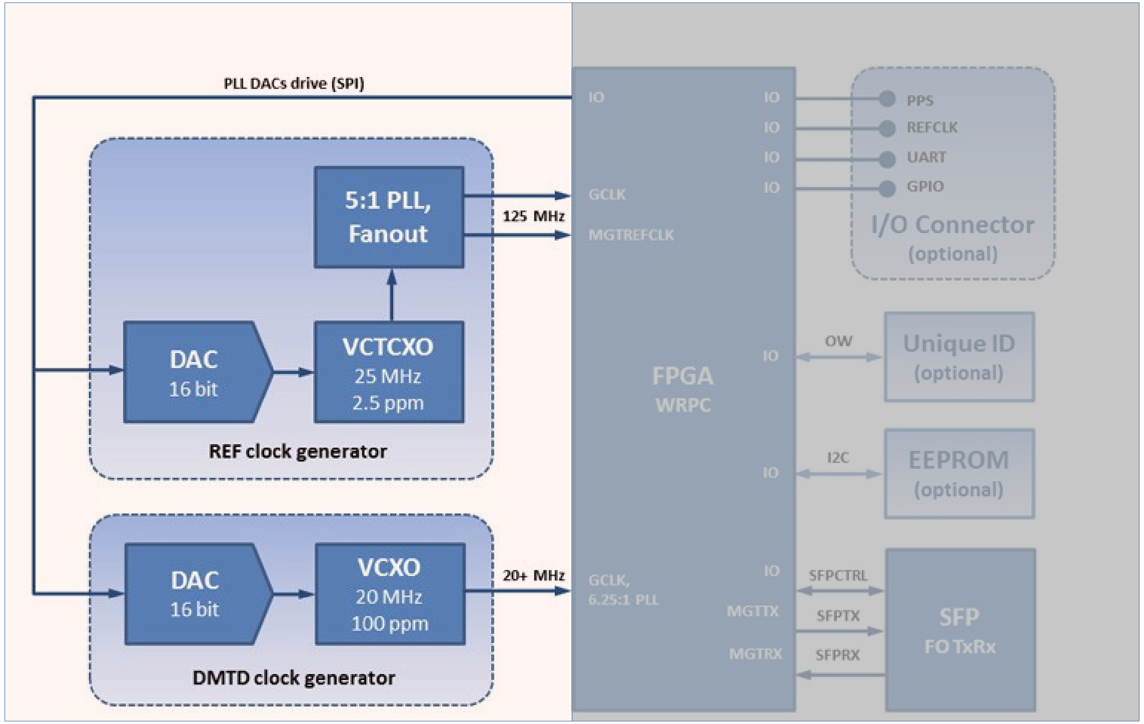
Optional:

Sensors (temperature)

UART for command line interface: USB adapter

Anything you want to add (and you are ready to write the software for)!

Clocking Structure



WR main purpose

What does White Rabbit ?

- It disciplines an oscillator from time information provided over ethernet

(Probably you have seen this question many times, and also different answers…)

What does discipline mean ?

- Same frequency (or an integer multiple)
- Fixed phase offset

How to discipline an oscillator ?

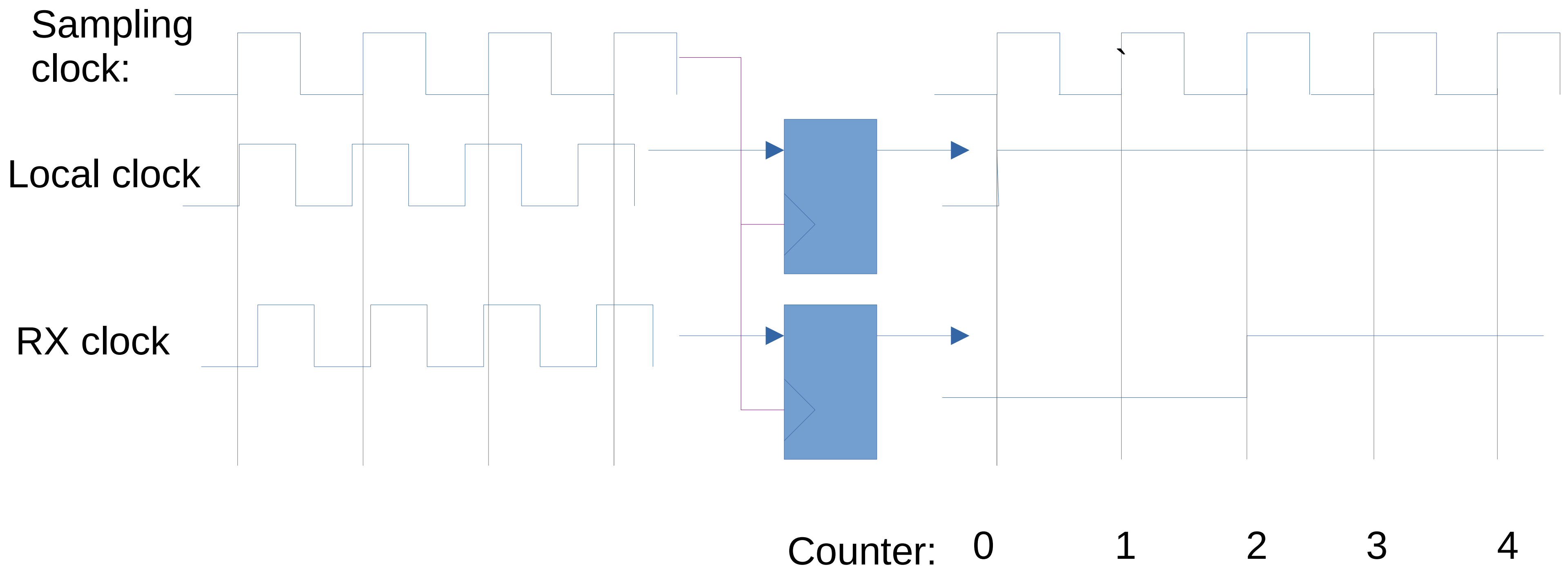
Local oscillator ~~frequency~~ phase is compared with the phase of a virtual oscillator.

Not so virtual, it’s the clock recovered from ethernet (in fact, a divided one). This clock represents the master clock.

From phase comparison, a PI controller (the SoftPLL) disciplines a local oscillator.

How to compare two clocks ?

The solution: DDMTD – Digital Dual Mixed Time Difference



External oscillators

WR needs 2 tunable clocks:

- The disciplined clock (the reference clock)
- The DDMTD clock

WR is also able to discipline other clocks (auxillary clocks)

External oscillators: frequency

Mostly defined by the transceiver

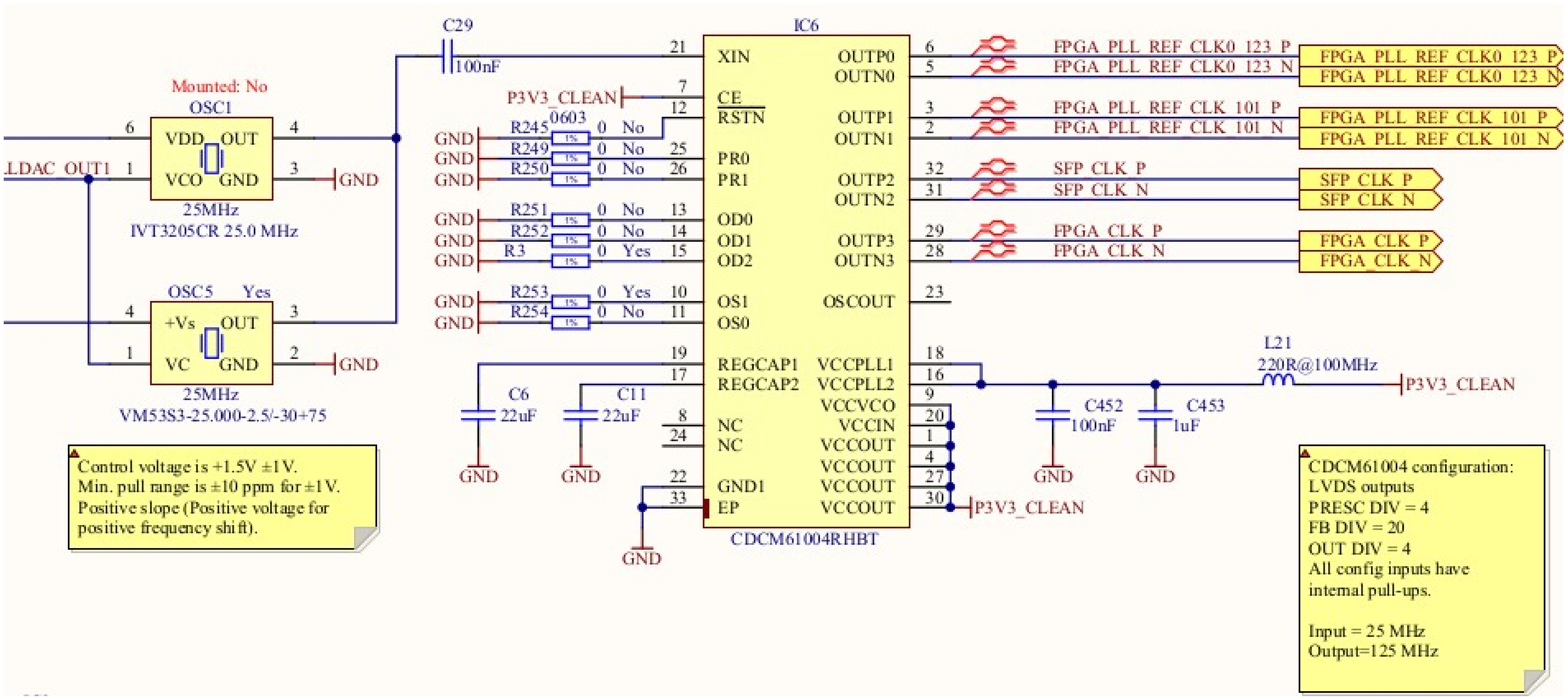
- Which provide the WR frequency

We are using 1G ethernet, but 1GHz is too high for an FPGA clock,
The transceiver use a 16b parallel interface (8b on Spartan6 or Virtex5).

So the data clock is 1GHz/16 = 62.5Mhz (125Mhz on Spartan6 or Virtex5)
For DDMTD, we use $n=2^{14}=16384$, so $F_{ddmtd} \approx 62.496$MHz

Note:
1G ethernet use 8B/10B encoding, so serial clock is 1G*10/8 = 1.25GHz
This recovered clock is then divided by 20.

External Clocking (SPEC - ohwr) 1/2



Control voltage is +1.5V ±1V.
Min. pull range is ±10 ppm for ±1V.
Positive slope (Positive voltage for positive frequency shift).

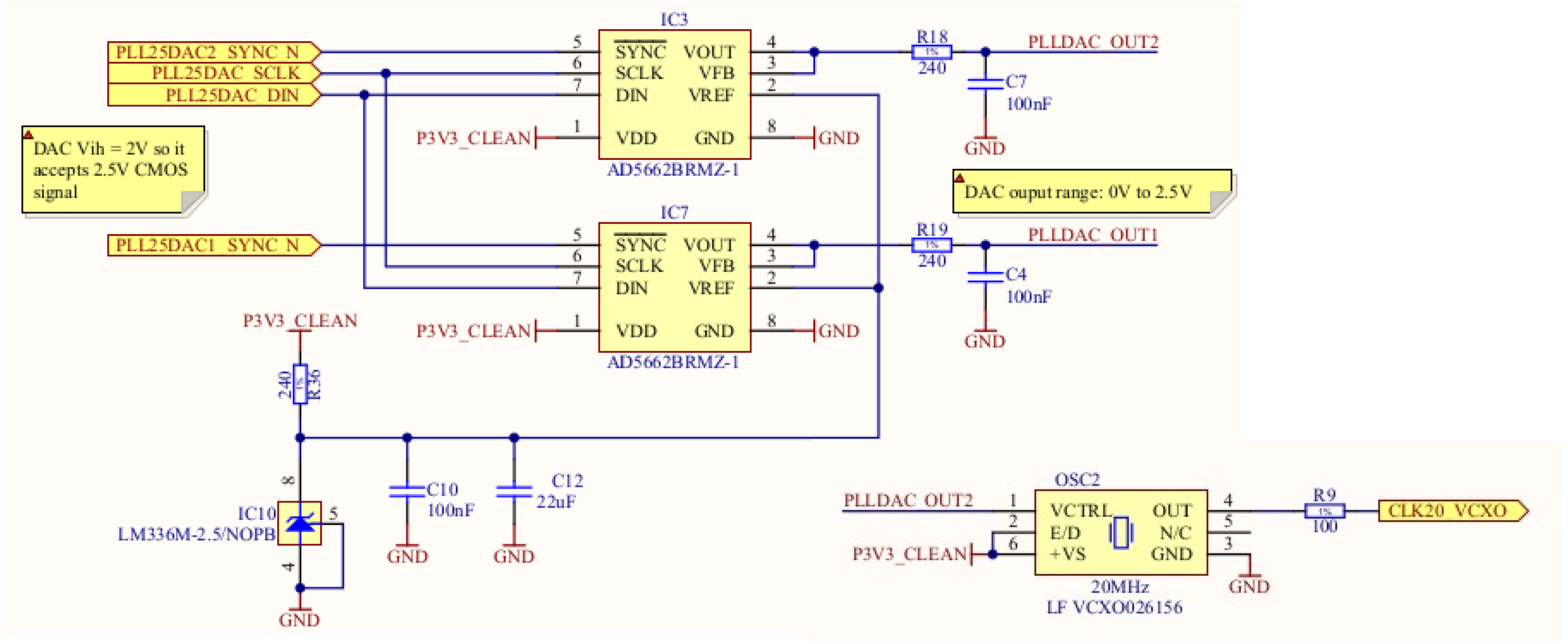
CDCM61004 configuration:
LVDS outputs
PRESC DIV = 4
FB DIV = 20
OUT DIV = 4
All config inputs have internal pull-ups.

Input = 25 MHz
Output=125 MHz

SPEC clocking comments 1/2

- Reference clock is provided by a 25MHz VCTCXO, small pull range
- External low-jitter PLL to provide 125MHz (better than the internal PLL)
- Double footprints for the oscillator
- External DACs to tune the oscillators
- PLL drives the transceivers and the FPGA
- Clean 3.3V power supply + DACs vref

External Clocking (SPEC) 2/2



SPEC clocking comments 2/2

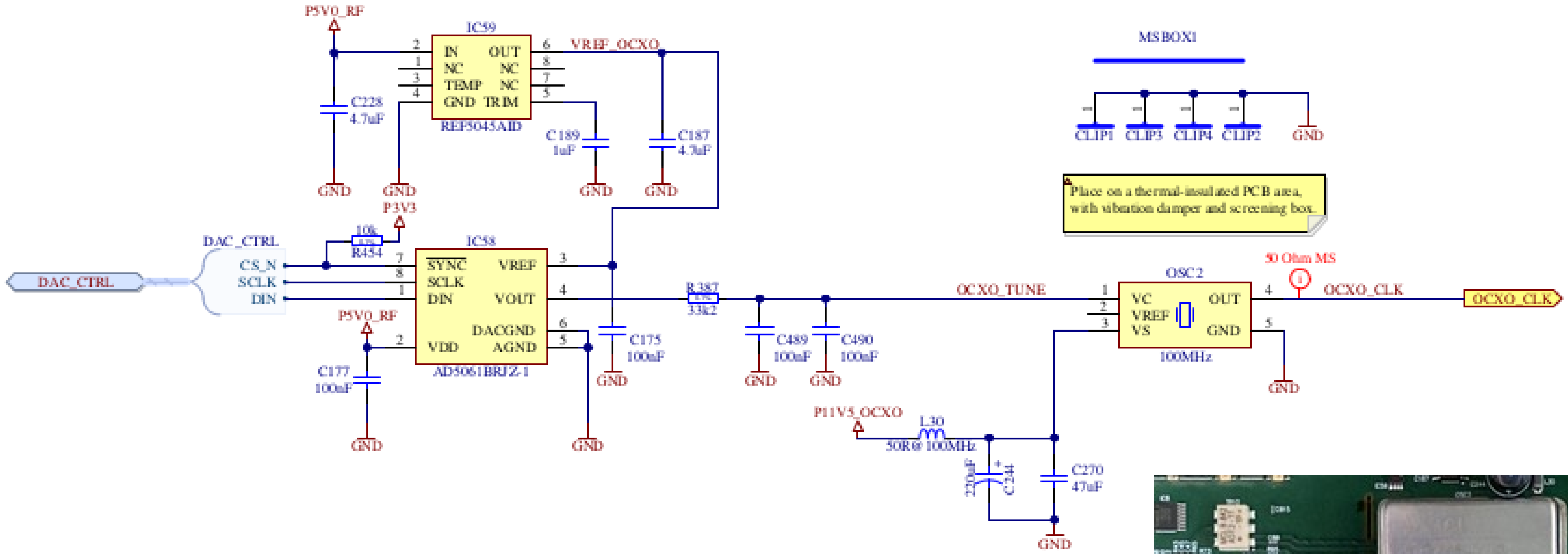
- 20Mhz VCXO for DDMTD (with +-100ppm pullability)
  - Use internal PLL to get ~62.5Mhz
  - n/(n+1) ≈ 0.999_938 so need > 60ppm of pullability

- Some components may not be available anymore

- Quality of the clock

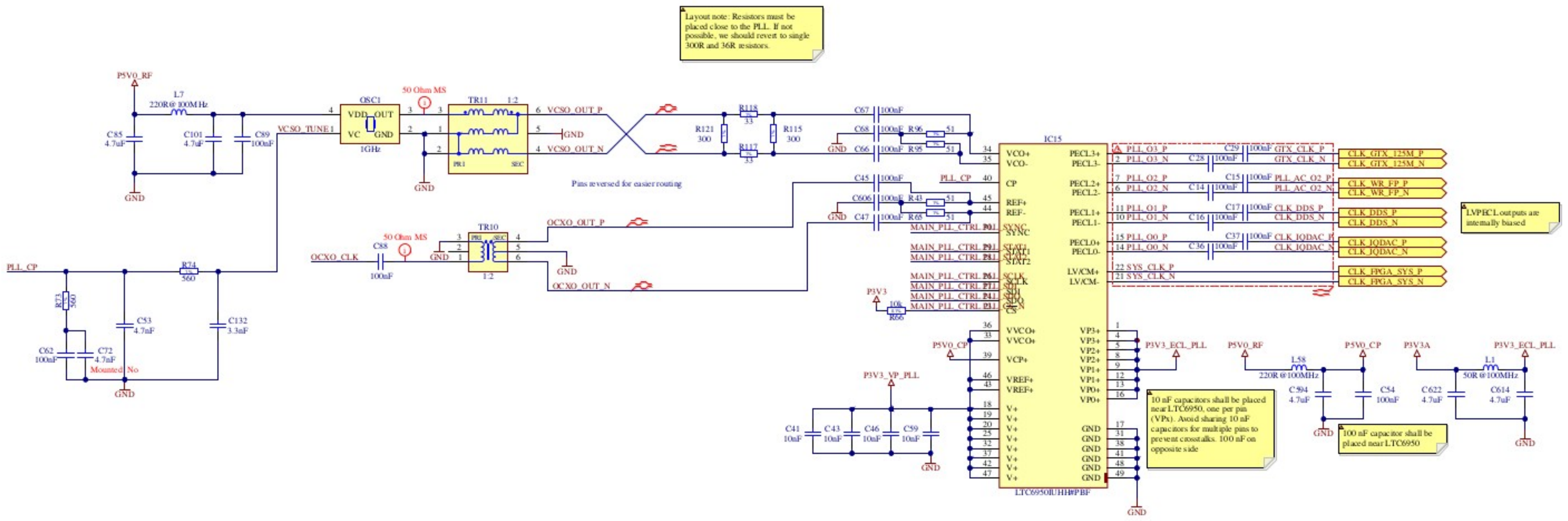
- Many components!

WR2RF (CERN) clocking (1/2)



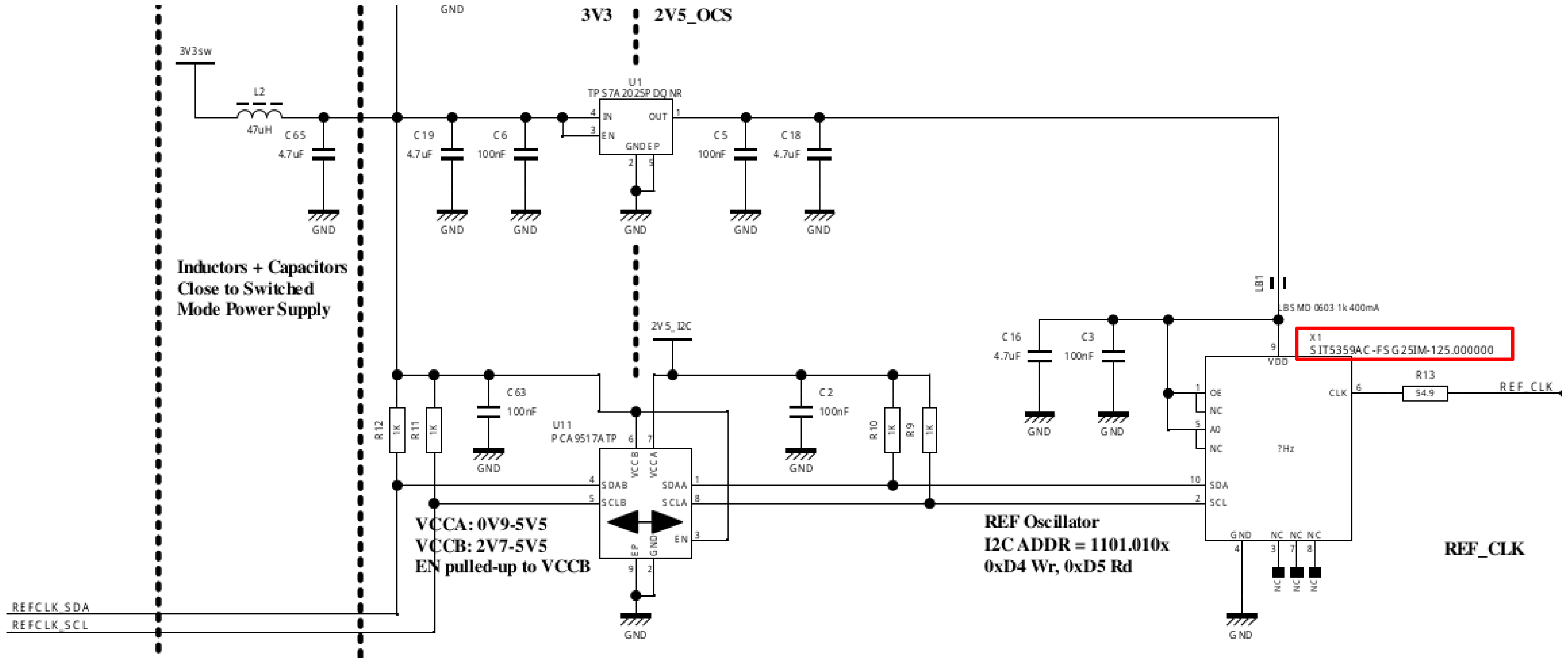
OCXO provides a very stable clock, small pullability!

WR2RF clocking (2/2)


Layout note: Resistors must be placed close to the PLL. If not possible, we should revert to single 300R and 36R resistors.
P5V0_RF
L7 220R@100MHz
OSC1 1GHz
VCSO_TUNE
TR11 1:2
VCSO_OUT_P
VCSO_OUT_N
Pins reversed for easier routing
R121 300
R115 300
IC15
LTC6950IUHH#PBF
TR10 1:2
OCXO_OUT_P
OCXO_OUT_N
OCXO_CLK
PLL_CP
50 Ohm MS
CLK_GTX_125M_P
CLK_GTX_125M_N
CLK_WR_FP_P
CLK_WR_FP_N
CLK_DDS_P
CLK_DDS_N
CLK_IQDAC_P
CLK_IQDAC_N
CLK_FPGA_SYS_P
CLK_FPGA_SYS_N
LVPECL outputs are internally biased
10 nF capacitors shall be placed near LTC6950, one per pin (VPx). Avoid sharing 10 nF capacitors for multiple pins to prevent crosstalks. 100 nF on opposite side
100 nF capacitor shall be placed near LTC6950
P3V3_ECL_PLL
P5V0_CP
P3V3A
P3V3_VP_PLL
GND

PLL discipline a VCSO (SAW) to provide a very stable high frequency clock!
Much higher price
Cf SPEC7 for another design with very low phase noise clock

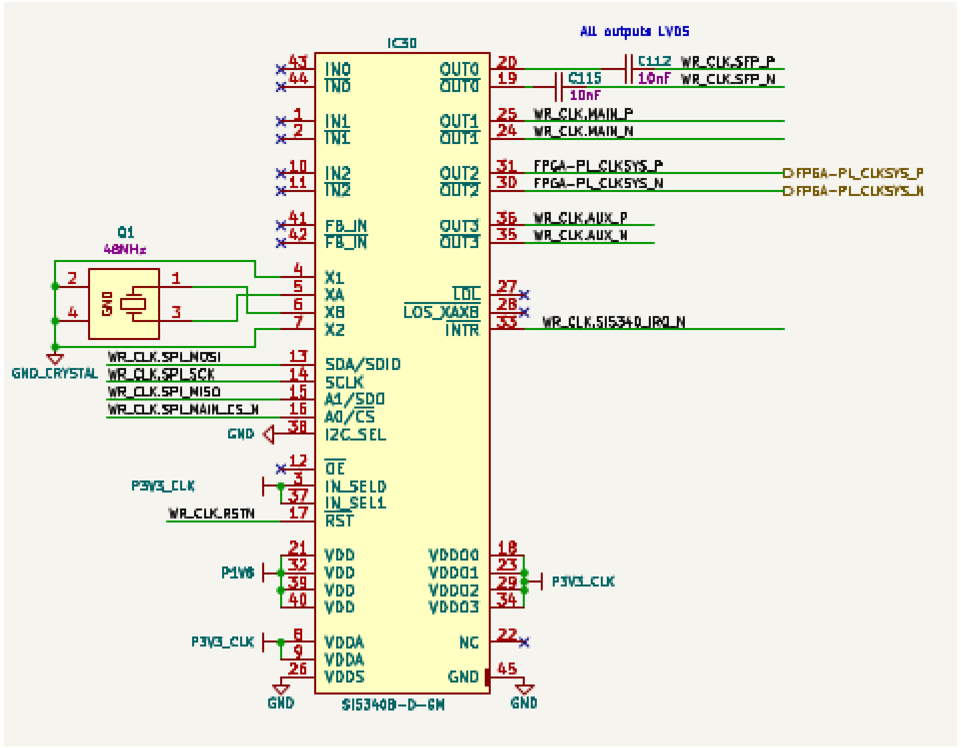
BabyWR (Nikhef) clocking



BabyWR clocking comments

- Clocks (Ref and DDMTD) provided by 2* Sit5359
- Integrated oscillator + fractional PLL
- Small package
- Single ended signal
- Tuned through I2C

WREN (CERN) clocking



- Si5340 fractional PLL
- NOT for ultra-low phase noise
- But very flexible
- ‘Tuned’ through SPI

Fractional PLL vs VCO+DAC

With fractional PLL, fixed oscillators can be used.

This simplifies the BOM (less components).

But according to (some) crystal vendors, fractional PLLs have more phase noise…

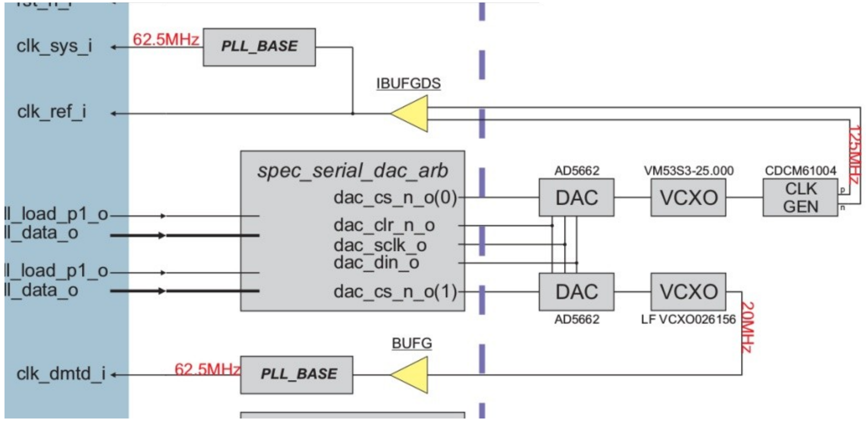
Some FPGA (eg: ZynqUS+) even have fractional PLLs inside with enough precision bits (eg: QPLL of GTHE4).

See branch `mle/upstream/zcu102` for an example.

Oscillator tuning

WRPC (SoftPLL) tunes the oscillators
The interface is generic:
 data port + pulse

There are many ways to tune:
 DAC (SPI-like), I2C, SPI, …



The repository provides many ‘adapters’:

- `modules/wr_dacs`: serial interface (SPI-like) + arbiter
- `modules/wr_sit5359_interface`: for Sit5359
- `modules/wr_si57x_interface`: for Si57x

You can design you own adapter (eg: WREN)

Clocking Structure (inside)

Clocking Structure

The FPGA requires one or more clock for the logic, and a reference clock for the transceiver

Bootstrap: a clock must be present to start the FPGA...

Current FPGA can clock the logic from the transceiver reference clock (save some pins)

The WRPC also needs a `sys_clk` (in general 62.5Mhz).

If you want your node to also be grantmaster, you need to provide 10MHz+PPS input

clk_ref: WR reference clock (1/2)

THE clock which is generated and disciplined by WR

- Must derive from a tunable source
- Must be compared with the RX clock
- Must be transmitted by TX (as the master will also compare it)

Note: the terms ‘reference clock’ are also used in other contexts (like PLL), or at other places in the WR core (like reference clock for GT). Do not confuse!

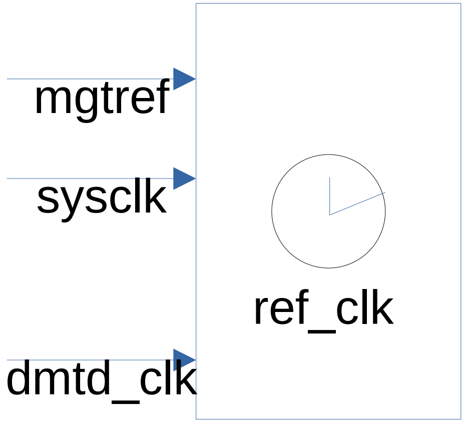
clk_ref: WR reference clock (2/2)

clk_ref:

- Clock time ports (`tm_tai`, `tm_cycles`, …) and pps.
- Might not be present during reset…
- Might not be in sync with `sys_clk`. Might be an issue if you need to send it over the network (fabric is clocked by `sys_clk`).
- It’s an internal clock. Can be output, but not with a high quality

Question: who generates this clock ?

Clocking Structure

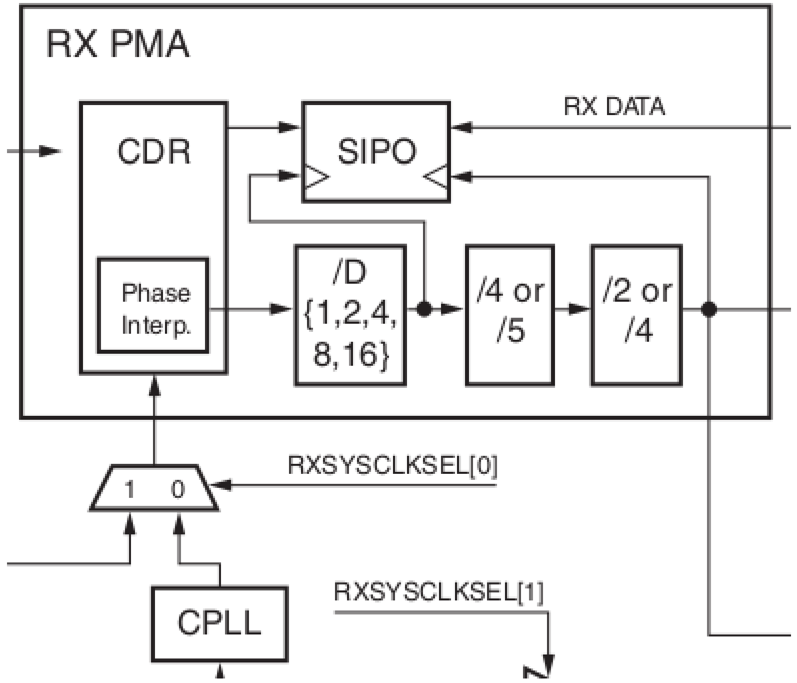


The key element is the transceiver:

- RX recover the clock from the master. The phase is compared to the local reference.
- TX transmit the local clock to the master. The master also measure the phase.

(The dmtd clock is used only for measuring the phase)

Clocking Structure



(From Xilinx ug476 – 7Series GTX/GTH)

Transceiver have PLLs to generate serial clocks (1.25Ghz)

And use dividers to generate ‘parallel’ clocks (there are RX and TX dividers)

Dividers create uncertainties of the phase (n times UI, so 8/16ns!)

All these clocks are not necessarly aligned.

Clock non-determinism

For RX: the phase non-determinism can be lifted by looking for the comma character. The phase offset corresponds to the number of bits shifted during comma alignment (bit-sliding)
Well, hopefully. That's the theory and might be more complex…

Clock non-determinism

For TX: we don’t know as we don’t have any readback!
Solution: the TX clock is the reference clock. (Or a clock in phase). This is the clock to be aligned with the master. This is the clock of the timestamp.

Or: It is possible to use (or synchronize or align) a clock with the TX clock. That’s deterministic and this could be the reference clock.

The latter is enforced by LPDC. This allows ref_clk to be synchronized with an external source, which could be then also used to locally distribute a ‘WR clock’ (with a fixed phase)

Quizz

- What if $F_{ddmtd} > F_{ref}$?
- Do we need an external Ref clock ? Can't we instead use the recovered clock ?
- Can the transceiver clock be a fixed clock ?
- Can F_{ddmtd} be directly derived from F_{ref} ?
- What about using $n=2^{15}$ or 2^{16} for DDMTD offset instead of 2^{14} ?

HDL: Instantiation

The Structure

- `modules/wrc_core/xwr_core.vhd`: the bare core, should be the reference
- `modules/wrc_core/wr_core.vhd`: without records (for verilog)

- `board/XX/xwrc_board_XX.vhd`: Wrapper for a board. Core + transceiver + PLL + DAC adapter. Recommended to simplify reuse of WR for a particular board.

- `board/common/xwrc_board_common.vhd`: Common part of the wrapper, define multiple fabric interfaces (Streamer, Etherbone, Loopback, Plain)

The Structure

- `platform/xilinx/wr_gtp_phy`: Transceivers for Xilinx family

- `platform/xilinx/xwrc_platform_xilinx.vhd`: Clocking + Transceiver for Xilinx family (ISE)
- `platform/xilinx/xwrc_platform_vivado.vhd`: Likewise but for Vivado (ZUS+)

Note: The `xwrc_platform_xxx` files are not very flexible. It might be easier to simply inspire from them instead of using them.

Note: See `platform/altera` for Altera FPGA.

The transceivers (Xilinx)

`platform/xilinx/wr_gtp_phy`: Transceivers for Xilinx family

- 7 series: gthe2, gtxe2, gtpe2
- US: gthe3
- US+: gthe4
- Virtex6 (For the switch)
- Spartan6, Virtex5
- *Missing:* gty

Wrapper around Xilinx transceivers

- Mostly deals with bitsliding
- LPDC (topic for another presentation)

Wrpc-sw: the software

Embedded software

It is required.

Mostly C code (+ crt0.S + linker script)

Target Risc-V on wrpc-v5 (and probably on the future too)

Should be compiled on a Linux platform

We provide gcc 11 for Risc-V on ohwr.org. But any Risc-V compiler should work (if not too old).

Generic platforms (8b and 16b), configuration through a menu.

Embedded software

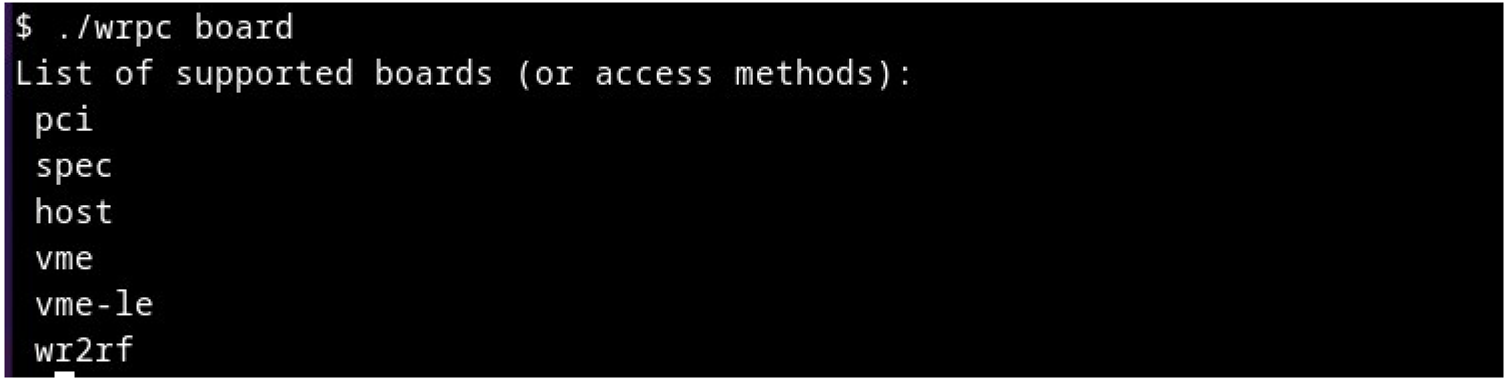
Some parts are shared with the switch (SoftPLL)

Structure: topic for another talk.

Host Software

`wrpc` is now the tool which does everything. The swiss knife.

Could be used only if there is a path to the `slave_i/slave_o` bus port of the core.

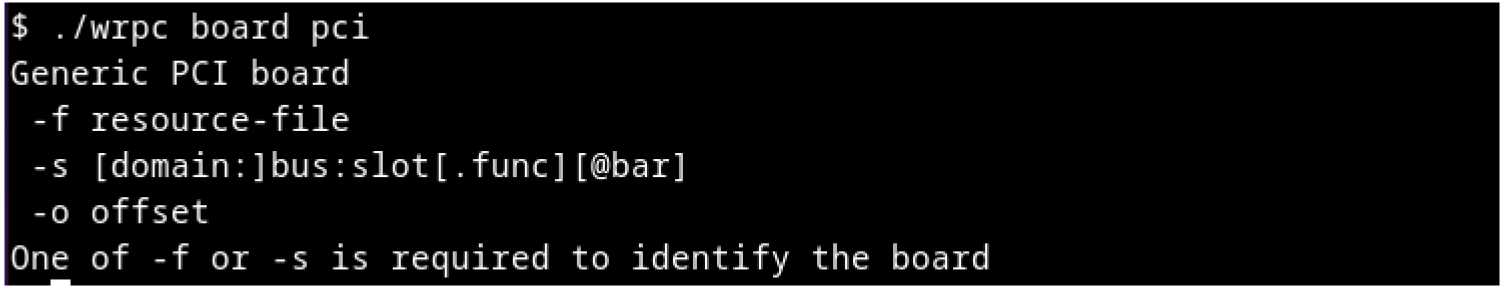


```
$ ./wrpc board
List of supported boards (or access methods):
 pci
 spec
 host
 vme
 vme-le
 wr2rf
```

`pci` and `host` are generic
Hopefully, you don’t have to deal with vme!

PCI based boards

For cores reachable through PCI/PCIe.



```
$ ./wrpc board pci
Generic PCI board
 -f resource-file
 -s [domain:]bus:slot[.func][@bar]
 -o offset
One of -f or -s is required to identify the board
```

The resource file should be something like:

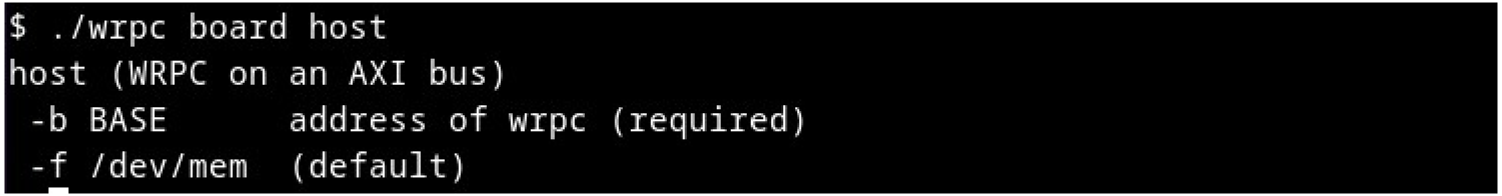
`/sys/bus/pci/devices/0000:02:00.0/resource1`

(The option -s translate bus:slot to a resource file)

The offset is the offset of the wrpc core within the BAR.

Hosted boards

For cores directly reached in memory



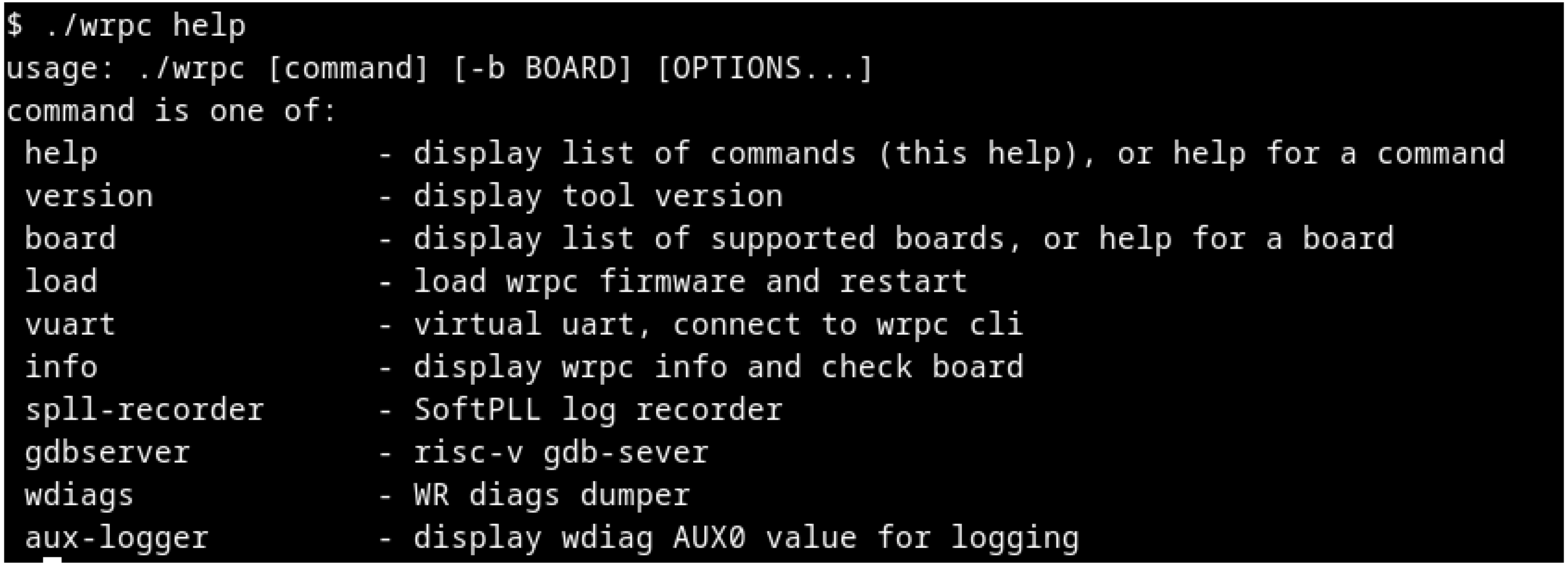
```
$ ./wrpc board host
host (WRPC on an AXI bus)
 -b BASE      address of wrpc (required)
 -f /dev/mem  (default)
```

Useful for SoC like ZynqUS+, when Linux is running on the chip and the wrpc core is connected on an AXI bus

BASE is the physical address of the core

(Strictly speaking, this could also be used on PCI, except the physical address is set only after enumeration)

`wrpc` commands



```
$ ./wrpc help
usage: ./wrpc [command] [-b BOARD] [OPTIONS...]
command is one of:
 help                   - display list of commands (this help), or help for a command
 version                - display tool version
 board                  - display list of supported boards, or help for a board
 load                   - load wrpc firmware and restart
 vuart                  - virtual uart, connect to wrpc cli
 info                   - display wrpc info and check board
 spll-recorder          - SoftPLL log recorder
 gdbserver              - risc-v gdb-sever
 wdiags                 - WR diags dumper
 aux-logger             - display wdiag AUX0 value for logging
```

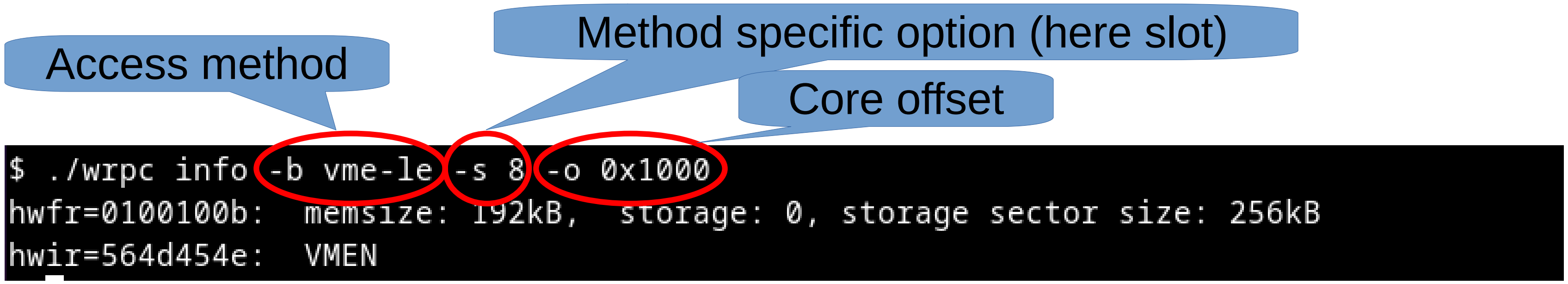
wrpc info

Display static information about the core:

- Features (memory size)
- Identification (arbitrary 4 letter string)

Very useful to check if access is OK (much less intrusive than other commands)

Try before any other command



```
$ ./wrpc info -b vme-le -s 8 -o 0x1000
hwfr=0100100b:  memsize: 192kB,  storage: 0, storage sector size: 256kB
hwir=564d454e:  VMEN
```

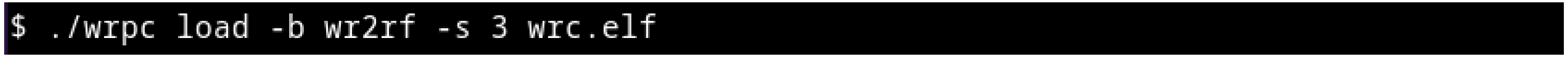
wrpc load

Load the wrpc-sw

Useful during the development to quickly change/update the software

It resets the cpu (which means it has to lock again)

Can load the ELF file (no need to convert to raw binary)

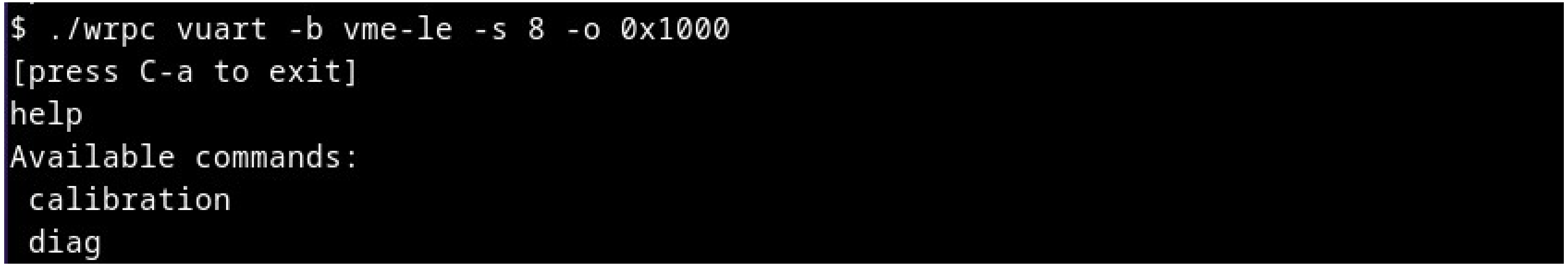


```
$ ./wrpc load -b wr2rf -s 3 wrc.elf
```

wrpc vuart

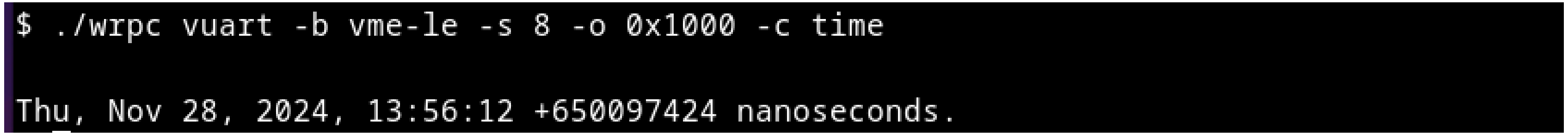
Just connect to the uart, no need for a cable

Press Control-A to exit



```
$ ./wrpc vuart -b vme-le -s 8 -o 0x1000
[press C-a to exit]
help
Available commands:
 calibration
 diag
```

You can also directly execute a command:

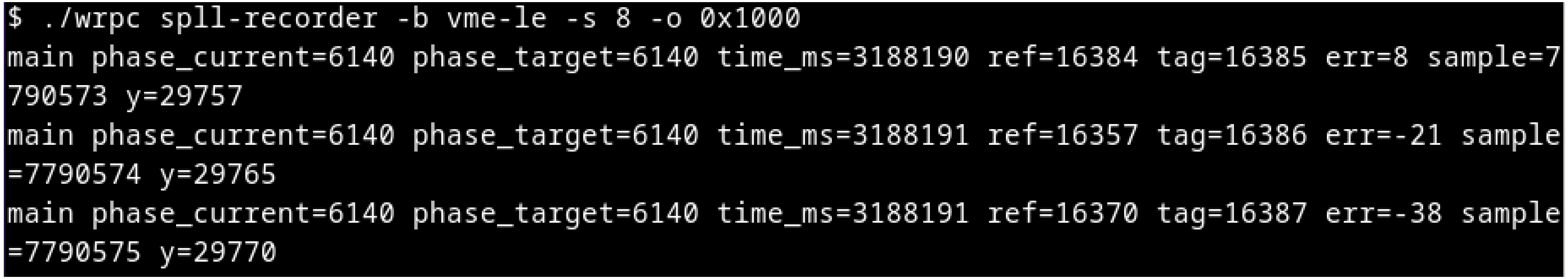


```
$ ./wrpc vuart -b vme-le -s 8 -o 0x1000 -c time

Thu, Nov 28, 2024, 13:56:12 +650097424 nanoseconds.
```

wrpc spll-recorder

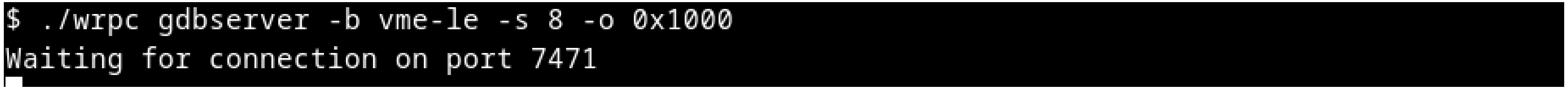
Dump the timetags (which are the primary data for the spll).
Very low-level tool, useful only if you need to debug the spll.



```
$ ./wrpc spll-recorder -b vme-le -s 8 -o 0x1000
main phase_current=6140 phase_target=6140 time_ms=3188190 ref=16384 tag=16385 err=8 sample=7
790573 y=29757
main phase_current=6140 phase_target=6140 time_ms=3188191 ref=16357 tag=16386 err=-21 sample
=7790574 y=29765
main phase_current=6140 phase_target=6140 time_ms=3188191 ref=16370 tag=16387 err=-38 sample
=7790575 y=29770
```

wrpc gdbserver

Connect a gdb server to the Risc-V core
Allows you to debug the software



```
$ ./wrpc gdbserver -b vme-le -s 8 -o 0x1000
Waiting for connection on port 7471
```

Maybe a topic for a talk – you need to know a little bit about gdb

Notes:

- You’d better to build wrpc-sw without LTO and with a low level of optimization
- Before reloading software, do `mon reset` to reset the cpu (in particular, to disable the interrupts)

Troubleshooting...

Troubleshooting

This is a list of problems I had or I have seen.

Some might be completely stupid mistakes

Not comprehensive, but I will try to update the list.

Your contribution is welcome!

IMPORTANT: it is **HIGHLY** recommended to make the WRPC visible on the host bus (connect `slave_i/slave_o`). It allows to easily update the software, have vuart, and the debugger. If not, connect at least the uart (`uart_rxd_i, uart_txd_o`). Have a few free pins you can connect to a scope.

My board doesn’t start !

I boot my board, but nothing appear on the console

- Baudrate ? If you are using the serial console, it should be 115200 baud. Maybe the sys_clk is not correct…
- Address ? If you are using ‘`wrpc vuart`’, maybe the address is not correct. Check the output of ‘`wrpc info`’

- No software loaded ? During development, it makes sense to not embed the wrpc-sw (as Vivado parse it very slowly). It needs to be loaded. Maybe the address is not correct too.

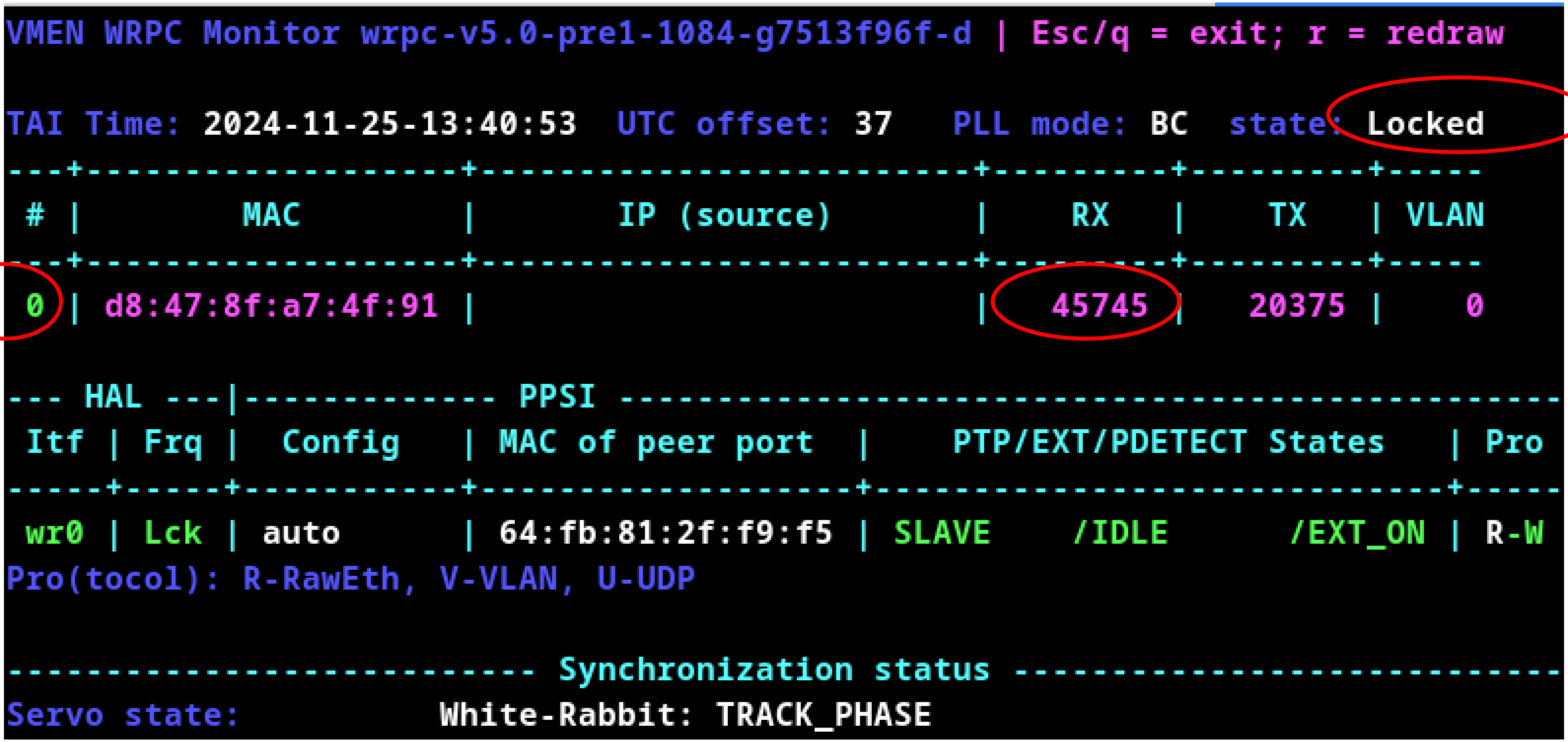
- Last solution: ILA on the cpu instruction bus, it should load instructions.

The software crashes quickly

I can see some startup messages, then nothing

- Maybe incorrect configuration (like addressing a non-existant device) ?
- The software is too large and get truncated or overwritten. Check the configuration with your hardware configuration.
- Enable more traces to see progress
- Debug with `pp_printf`, or ‘`wrpc gdbserver`’

Link issue



```
VMEN WRPC Monitor wrpc-v5.0-pre1-1084-g7513f96f-d | Esc/q = exit; r = redraw

TAI Time: 2024-11-25-13:40:53  UTC offset: 37   PLL mode: BC  state: Locked
---+--------------------+-------------------------+---------+---------+-----
 # |        MAC         |        IP (source)      |    RX   |    TX   | VLAN
---+--------------------+-------------------------+---------+---------+-----
 0 | d8:47:8f:a7:4f:91 |                          |   45745 |   20375 |    0

--- HAL ---|-------------- PPSI -----------------------------------------------
 Itf | Frq |  Config     | MAC of peer port  |     PTP/EXT/PDETECT States    | Pro
-----+-----+-----------+-------------------+-------------------------------+----
 wr0 | Lck | auto       | 64:fb:81:2f:f9:f5 | SLAVE    /IDLE        /EXT_ON | R-W
Pro(tocol): R-RawEth, V-VLAN, U-UDP


--------------------------- Synchronization status ----------------------------
Servo state:           White-Rabbit: TRACK_PHASE
```

No link

I have a prompt, but link is never present

- Check SFP is present, fully pushed in the cage, fibre is present…
- Maybe `sfp_tx_disable` is not driven or incorrectly driven.
- The transceiver is incorrectly clocked. Check presence of clocks.
- Might be a problem after the transceiver (auto negotiation)
- Connections between VHDL and Verilog (depends on Vivado versions…)
- Start with non-lpdc transceiver (and non-lpdc software)

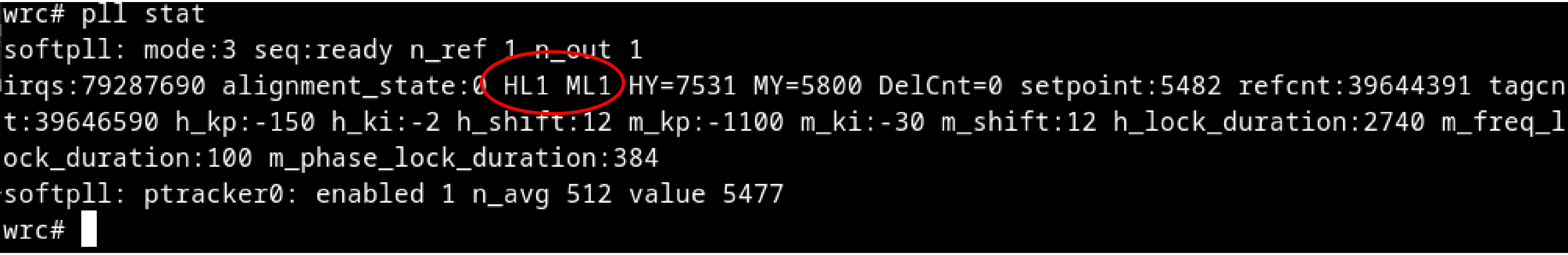
No link

- Mismatching SFPs ?
- Try with an SFP loopback
- (GT near-end loopback is possible but not well supported)
- PCB issue (impedence)
- ILA on the transceiver, in particular the reset FSMs

No lock

Link is present but cannot lock

- Check details with ‘pll stat’



```
wrc# pll stat
softpll: mode:3 seq:ready n_ref 1 n_out 1
irqs:79287690 alignment_state:0 HL1 ML1 HY=7531 MY=5800 DelCnt=0 setpoint:5482 refcnt:39644391 tagcn
t:39646590 h_kp:-150 h_ki:-2 h_shift:12 m_kp:-1100 m_ki:-30 m_shift:12 h_lock_duration:2740 m_freq_l
ock_duration:100 m_phase_lock_duration:384
softpll: ptracker0: enabled 1 n_avg 512 value 5477
wrc#
```

HL1: Helper (DDMTD) locked
ML1: Main locked

- Important: First link, second helper, third main pll

No helper lock

If helper is not locked…

- ddmtd clock not present ?
- ddmtd clock cannot be tuned ?
- ddmtd clock out of range ?
- wrong Ki/Kp sign ?

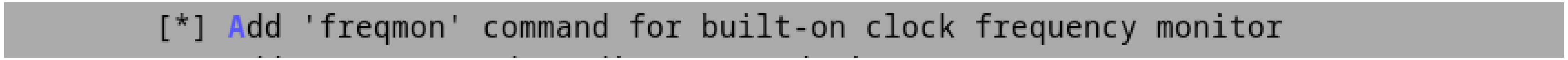
Connect the clock to a pin and check with a scope

Use freqmon command

Freqmon

Optional hardware + software feature

```
g_with_clock_freq_monitor   : boolean                          := true;
```



```
        [*] Add 'freqmon' command for built-on clock frequency monitor
```

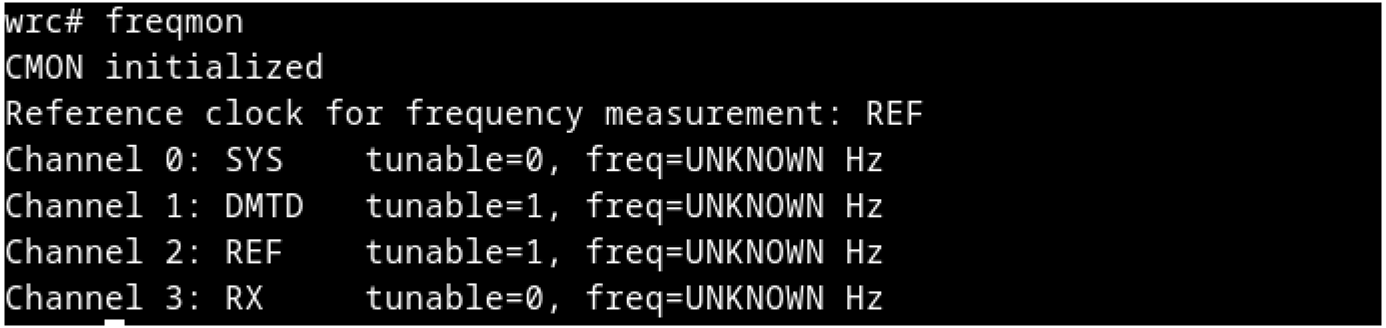
It directly drives the DAC lines and measure the clocks frequency
The best reference is the uplink clock (so link must be present)

Set low and high values, measure frequency and compute pullability

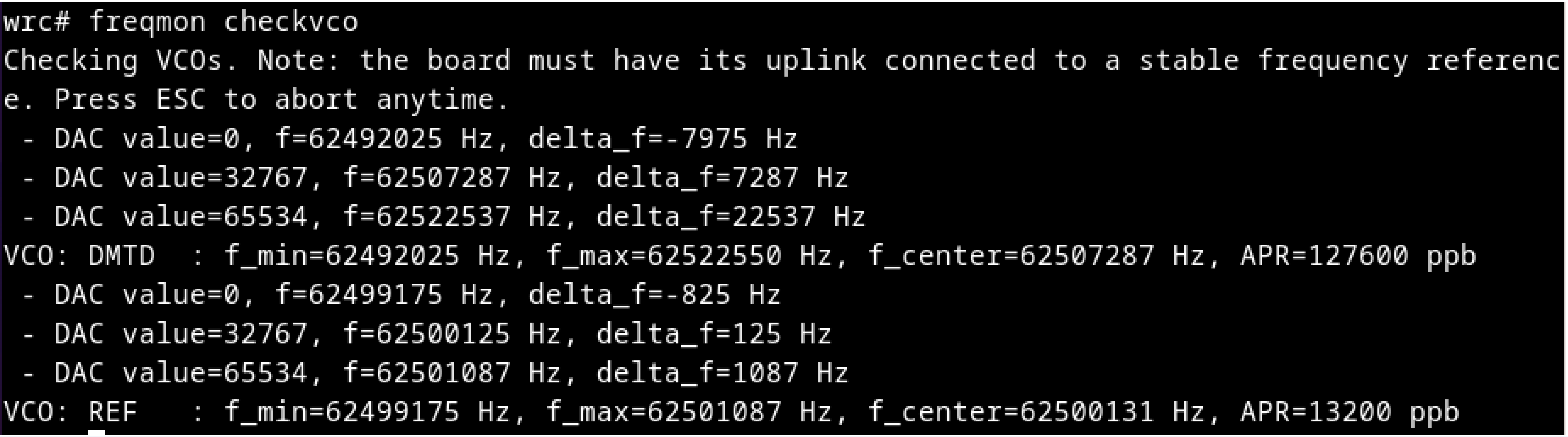
As DAC lines are driven, softpll and ptp are disabled!

- Check Fmin < 62.5MHz, Fmax > 62.5Mhz

Freqmon



```
wrc# freqmon
CMON initialized
Reference clock for frequency measurement: REF
Channel 0: SYS    tunable=0, freq=UNKNOWN Hz
Channel 1: DMTD   tunable=1, freq=UNKNOWN Hz
Channel 2: REF    tunable=1, freq=UNKNOWN Hz
Channel 3: RX     tunable=0, freq=UNKNOWN Hz
```



```
wrc# freqmon checkvco
Checking VCOs. Note: the board must have its uplink connected to a stable frequency referenc
e. Press ESC to abort anytime.
 - DAC value=0, f=62492025 Hz, delta_f=-7975 Hz
 - DAC value=32767, f=62507287 Hz, delta_f=7287 Hz
 - DAC value=65534, f=62522537 Hz, delta_f=22537 Hz
VCO: DMTD  : f_min=62492025 Hz, f_max=62522550 Hz, f_center=62507287 Hz, APR=127600 ppb
 - DAC value=0, f=62499175 Hz, delta_f=-825 Hz
 - DAC value=32767, f=62500125 Hz, delta_f=125 Hz
 - DAC value=65534, f=62501087 Hz, delta_f=1087 Hz
VCO: REF   : f_min=62499175 Hz, f_max=62501087 Hz, f_center=62500131 Hz, APR=13200 ppb
```

No main lock

If the main does not lock...

- Clock stability
- Detuned oscillator (output frequency out of the specification)
- Upstream clock not stable enough for an oscillator with tiny pullability
  - Use a low-jitter switch locked on an atomic clock
- SoftPLL PI parameters (Kp/Ki) incorrect
- `wrpc spll-recorder` can record and display time tags

PPS (or clocks) not aligned

The core is synchronized (TRACK_PHASE), but the PPS output is not aligned with the PPS of the switch (or of the GPS receiver)

- If the PPS of the switch is not aligned with the PPS of the receiver, you first need to align them
- It's about calibration

PPS phase jumps every restart

Each time the core is reset, or powered off and on, or the link goes down and up, the PPS phase (compared to the switch PPS) changes

- So the system is not deterministic!
- Main culprit: the FPGA transceiver
- How important is the jump: more than the UI (800ps) or less ?
- Some transceivers (GTX, Virtex6, GTHE4) have an LPDC version (jump is less than ~20ps). Topic for another talk...

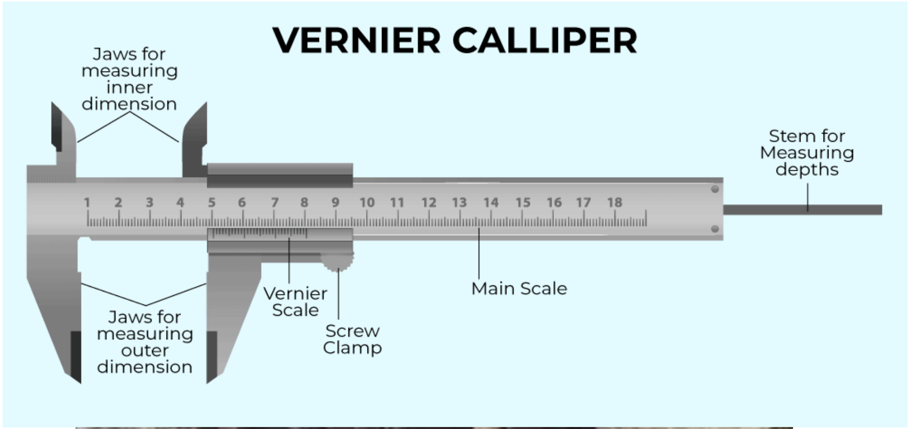
PPS phase depends on the temperature

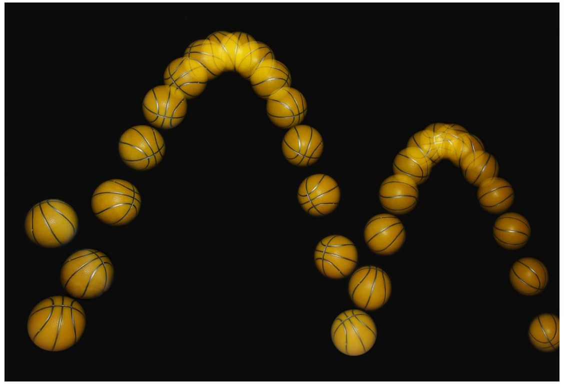
The PPS phase (compared to the PPS of the switch) depends on the temperature

- It’s a feature
- Setting the alpha parameter may help
- There are some temperature sensors

Backup: DDMTD

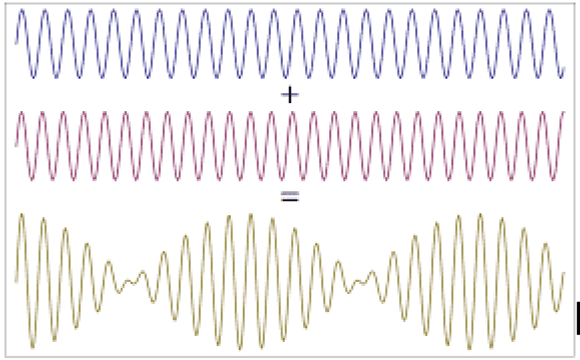
How does DDMTD work ?





Stroboscope

Wheels



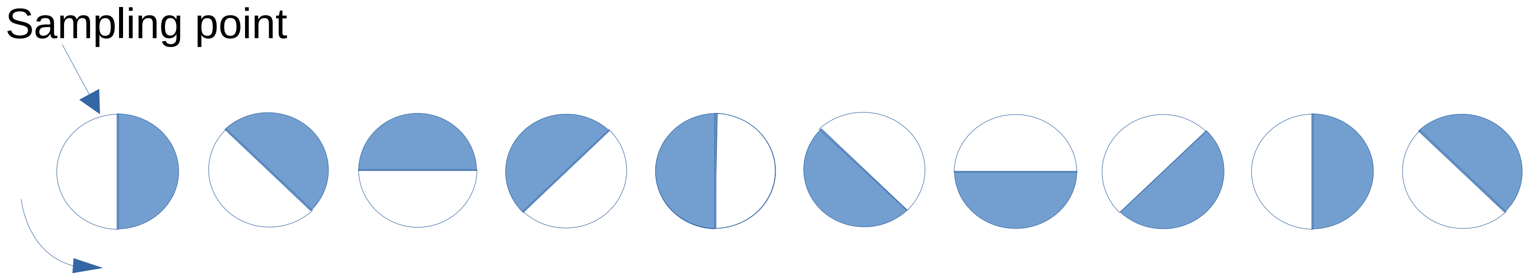
Beat waves

How does DDMTD work ?

Zoom-in effect

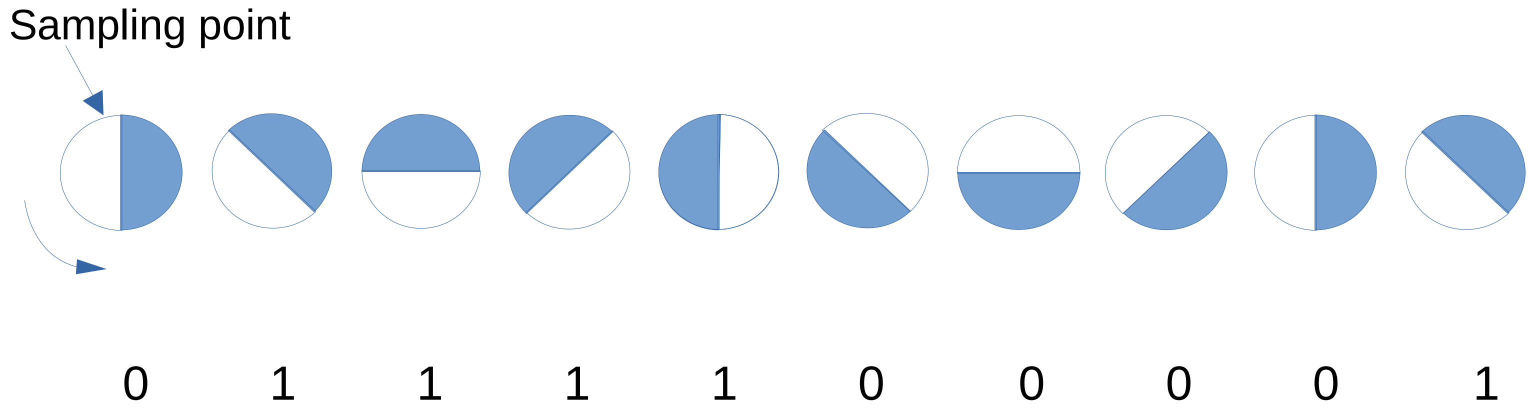
- Input (the clock) is periodic
- Watched at a slightly different frequency

Let use the rotating wheel effect:



How does DDMTD work ?

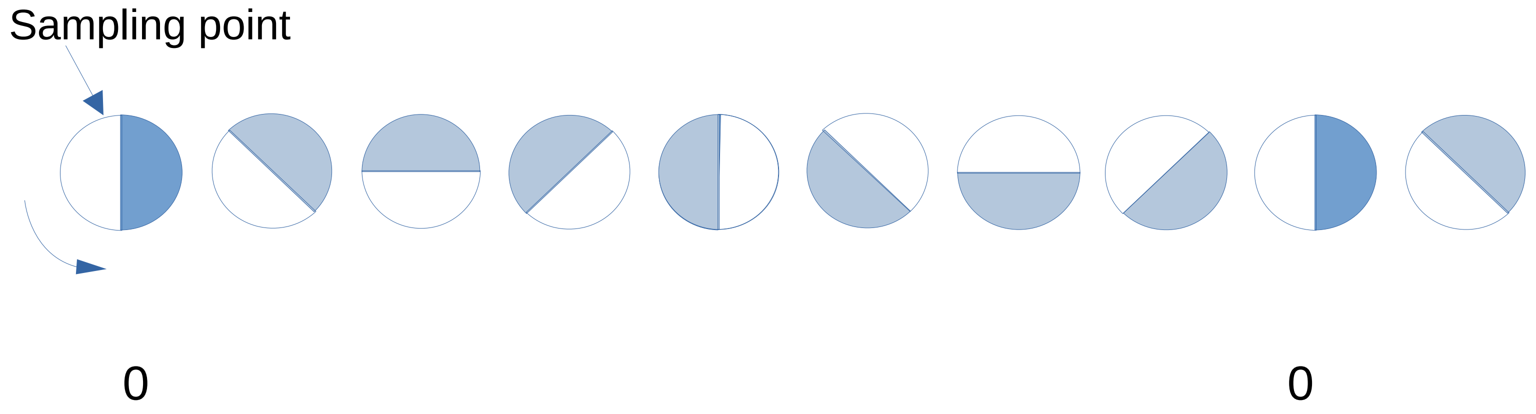
When sampling frequency is much higher than the sampled clock:



No zoom-in effect, low precision (defined directly by the sampling frequency)

How does DDMTD work ?

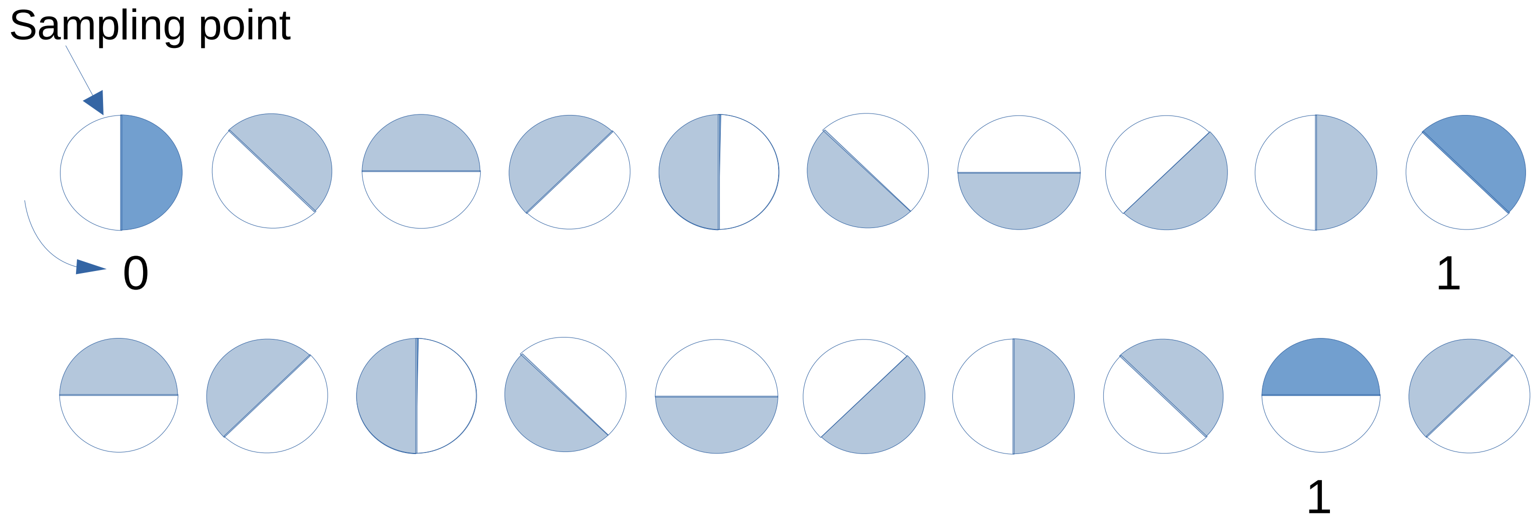
When sampling frequency is equal to the sampled clock:



Always 0 (or always 1), not very interesting!

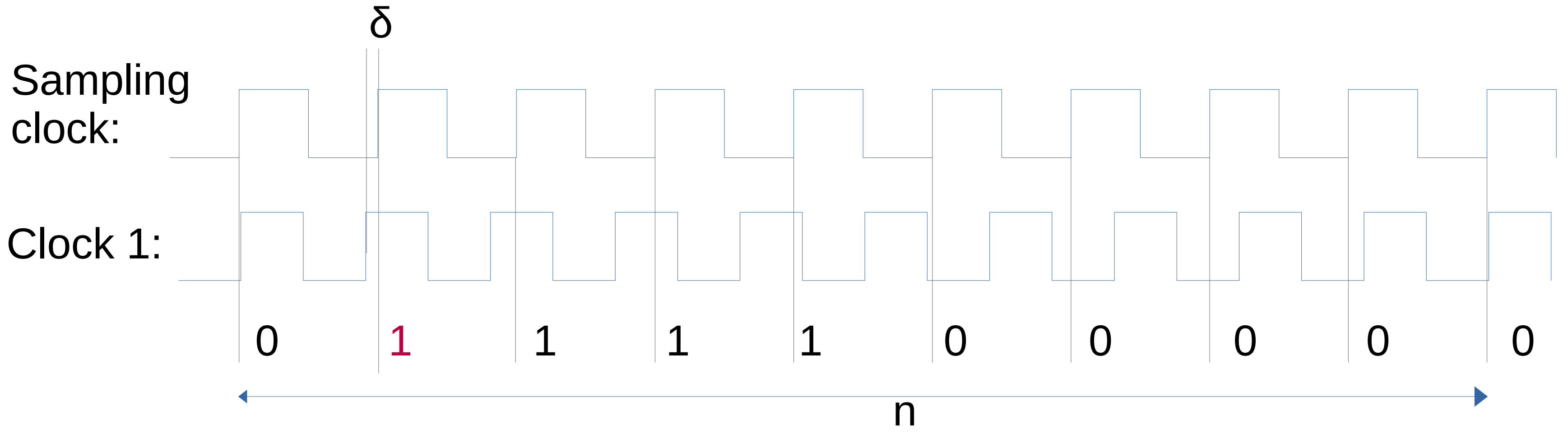
How does DDMTD work ?

When sampling frequency is slightly lower than the sampled clock:



Zoom-in effect!

How does DDMTD work ?

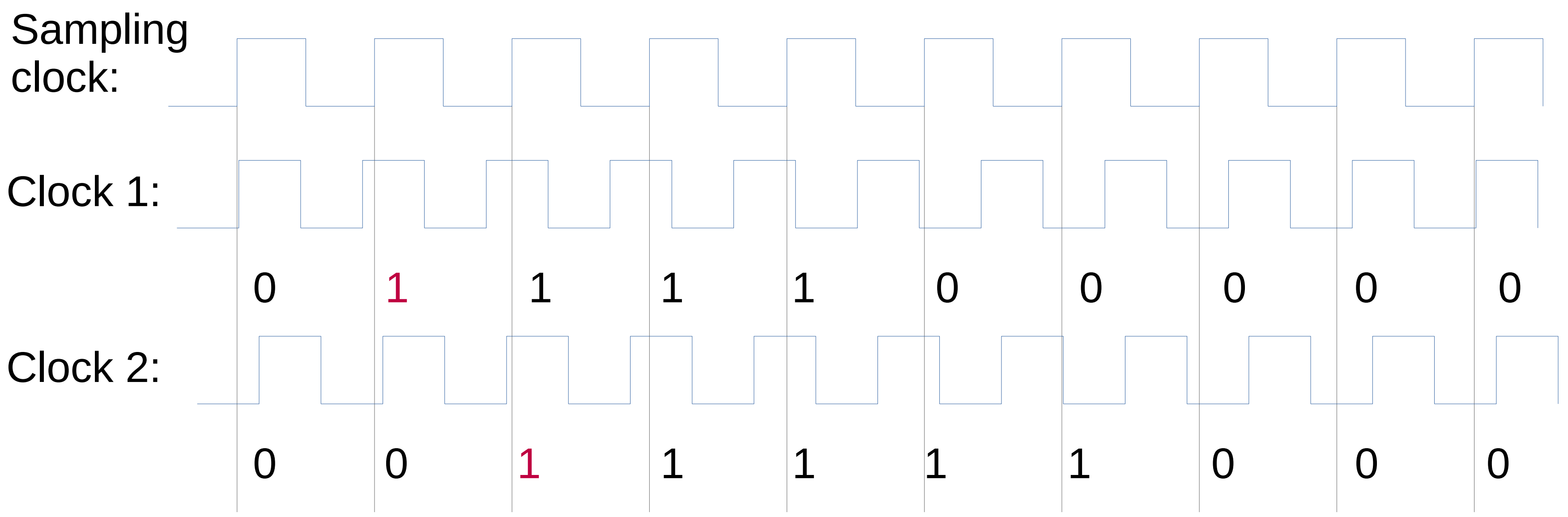


$$\text{If } P_s > P_1,\ \ \delta = P_s - P_1$$

$$n = P_1 / \delta = P_1 / (P_s - P_1) = 1 / (P_s / P_1 - 1) = 1 / (F_1/F_s - 1)$$

$$1/n = F_1/F_s - 1 \ \text{ so } F_s = F_1/(1/n + 1),\ F_s = F_1 * n/(n+1)$$

How does DDMTD work ?



DDMTD provides the phase difference between two clocks

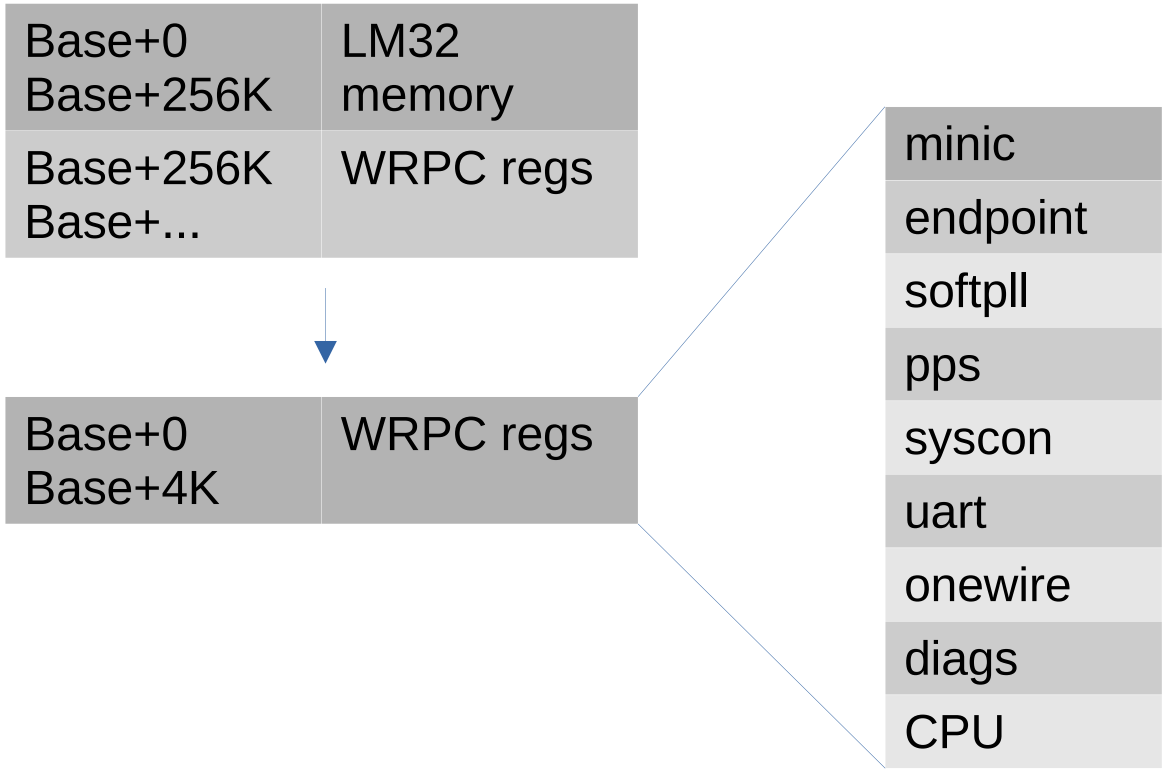
Backup: wrpc-v5 *vs* wrpc-v4

What has changed with wrpc-v5 ?

The cpu has moved from lm32 to risc-v

The cpu memory is not directly visible

Address space is much smaller!


Base+0
Base+256K
LM32
memory
Base+256K
Base+...
WRPC regs
Base+0
Base+4K
WRPC regs
minic
endpoint
softpll
pps
syscon
uart
onewire
diags
CPU