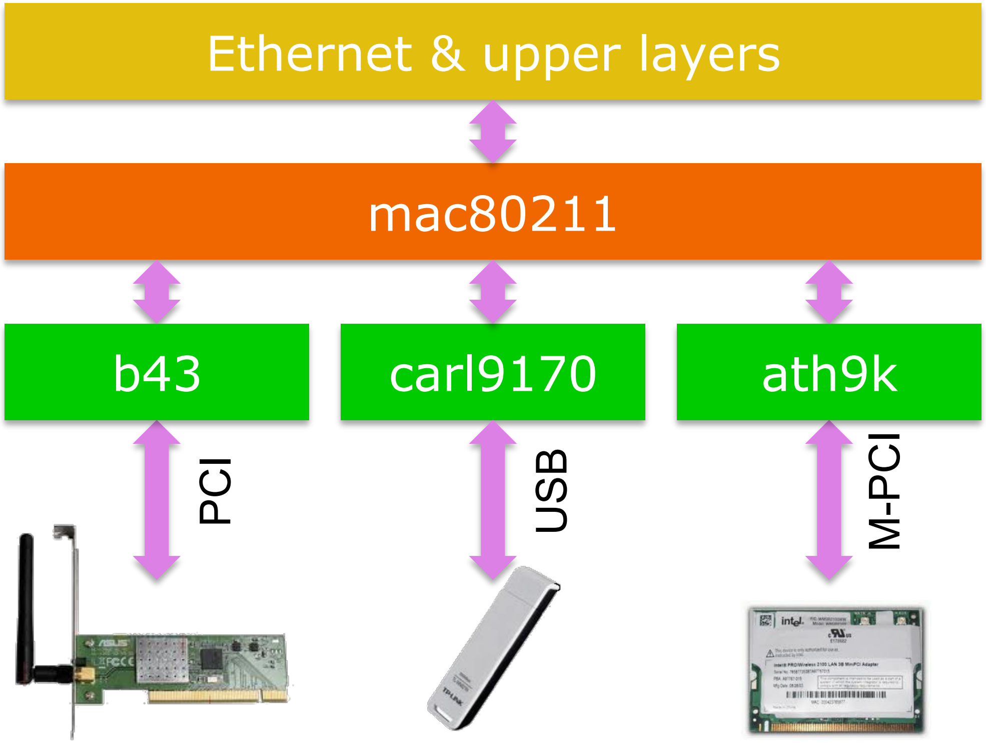
Descent to layer 2 and below An open firmware

A glimpse into the Linux Kernel Wireless Code

Part 2

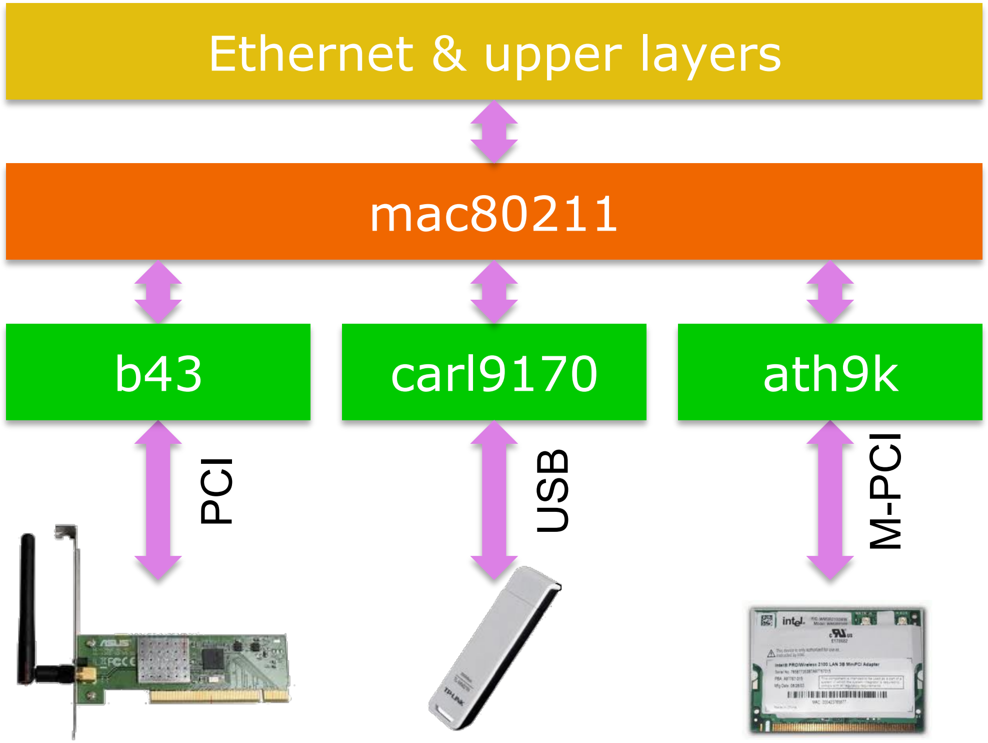
Linux & 802.11
Modular architecture

Wrapper for all hw
Find interface;
remove eth head;
add LLC&dot11 head;
fill (sa;da;ra;seq);
fill(control;duration);
set rate (from RC);
fill (rate;fallback);



Linux & 802.11
Modular architecture/2

Set up hw regs;
Fill hw private fields;
Send frame on DMA;
Get stats;
Reports to mac80211
Several MAC primitives missing!
Who takes care of ack?



Linux & 802.11
Modular architecture/3

Ethernet & upper layers

For sure

We will see the firmware in this course
but first…
Let's check why we should do that ☺

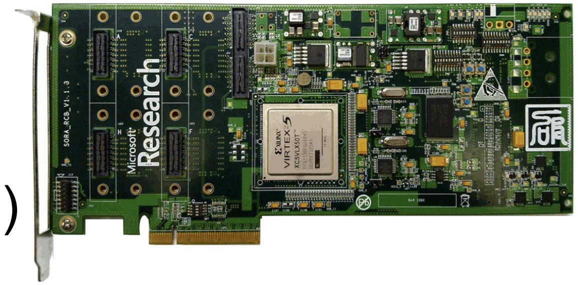
Firmware does

Why/how playing with 802.11

- Radio access protocols: issues
  - Some are unpredictable: noise & intf, competing stations
- Experimenting with simulators (e.g., ns-3)
  - Captures all “known” problems
    - Testing changes to back-off strategy is possible ☺
  - Unknown (not expected)?
    - Testing how noise affects packets not possible ☹
- **In the field testing is mandatory**
  - Problem: one station is not enough!

Programmable Boards

- Complete platforms like
  - RICE-WARP: Wireless open-Access Research Platform
  - NI-RIO2940
  - Microsoft SORA
  - Based on FPGA
  - Everything can be changed
    - MAC + PHY (access to OFDM symbols!)
  - Two major drawbacks
    - More than very expensive
    - Complex deployment
  - **If PHY untouched: look for other solutions!**

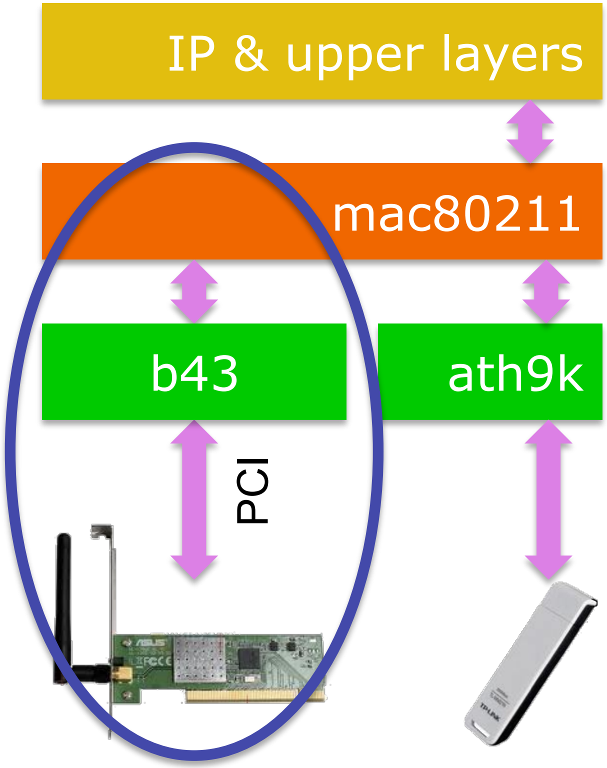


Off-the-shelf hardware

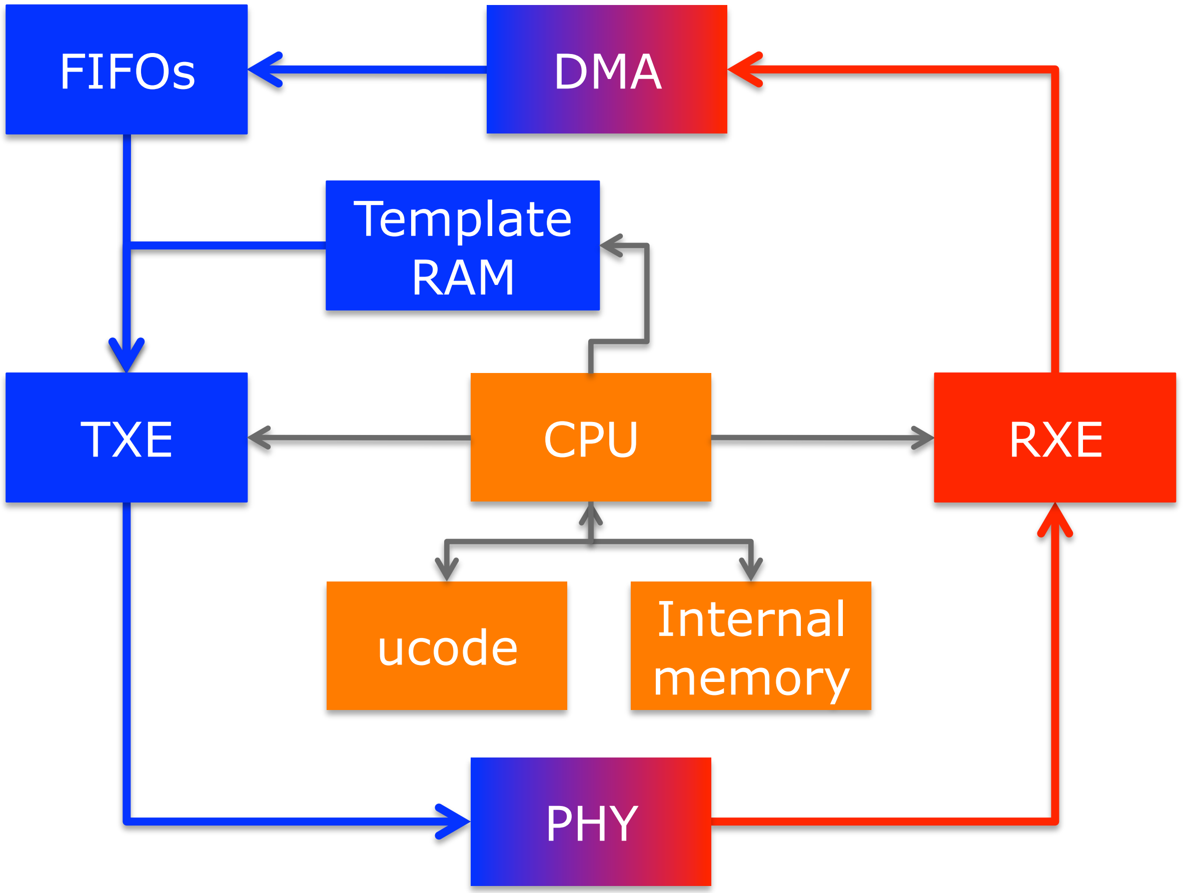
- Five/Six vendors develop cheap WiFi hw
  - – Hundreds different boards
  - – Almost all boards load a binary firmware
    - MAC primitives driven by a programmable CPU
  - – Changing the firmware ➔ Changing the MAC!
- Target platform:
  - – Linux & 802.11: modular architecture
  - – Official support prefers closed-source drivers ☹
  - – Open source drivers && Good documentation
    - Thanks to community! ☺

Linux & 802.11
Broadcom AirForce54g

- Architecture chosen because
  - Existing asm/dasm tools
    - A new firmware can be written!
  - Some info about hw regs
- We analyzed hw behavior
  - Internal state machine decoded
  - Got more details about hw regs
  - Found timers, tx&rx commands
  - Open source firmware for DCF possible
- We released OpenFWWF!
  - OpenFirmWare for WiFi networks



Broadcom AirForce54g Basic HW blocks



Description of the HW

- CPU/MAC processor capabilities
  - 88MHz CPU, 64 general purpose registers
- Data memory is 4KB, direct and indirect access
  - From here on it's called Shared Memory (SHM)
- Separate template memory (arrangeable > 2KB)
  - Where packets can be composed, e.g., ACKs & beacons
- Separate code memory is 32KB (4096 lines of code)
- Access to HW registers, e.g.:
  - Channel frequency and tx power
  - Access to channel transmission within N slots, etc…

TX side

- Interface from host/kernel
  - Six independent TX FIFOs
  - DMA transfers @ 32 or 64 bits
  - HOL packet from each FIFO
    - can be copied in data memory
      - Analysis of packet data before transmission
      - Kernel appends a header at head with rate, power etc
    - can be transmitted “as is”
    - can be modified and txed
      - Direct access to first 64 bytes

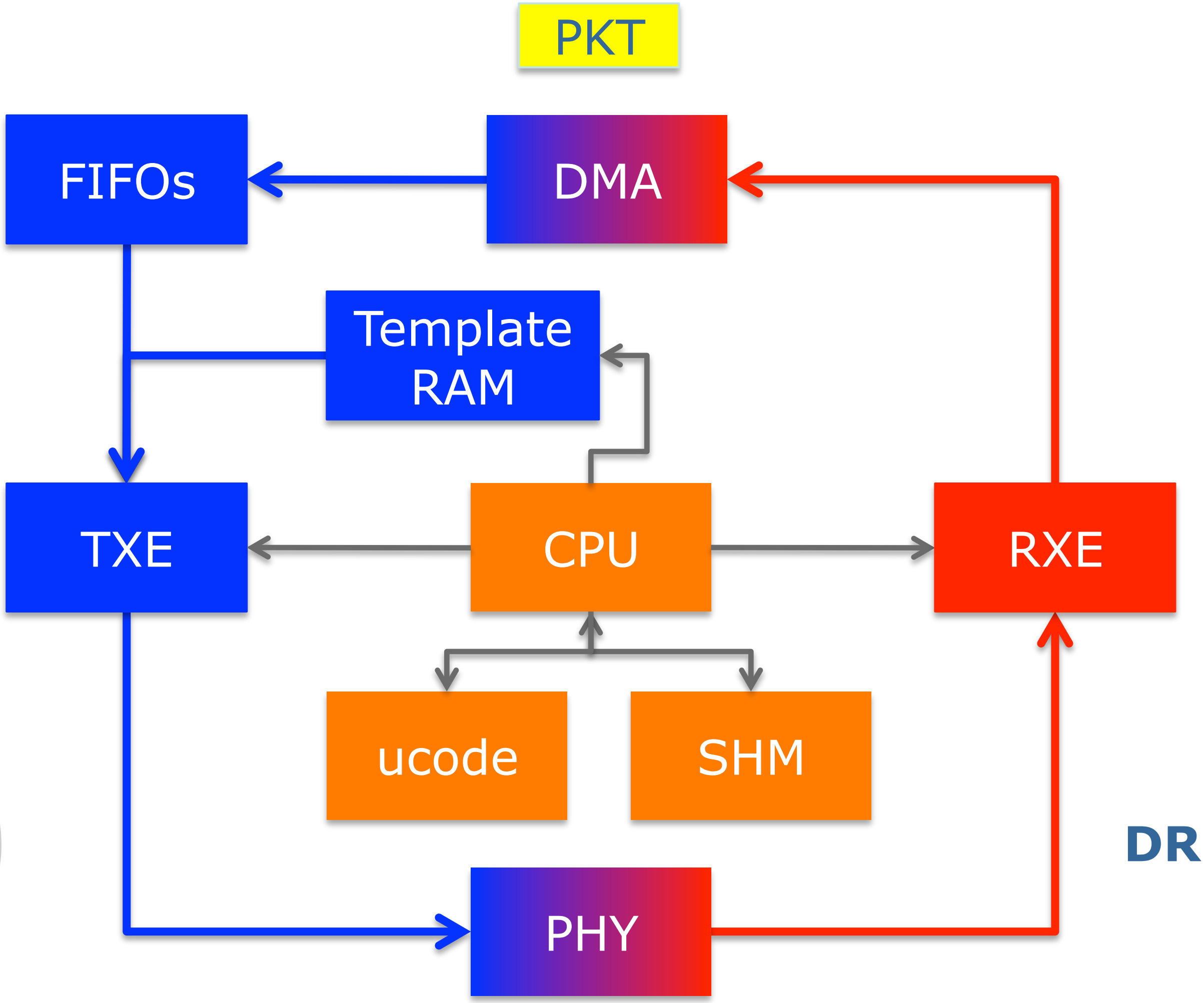
TX side/2

- Interface to air
  - Only 802.11 b/g supported, soon n/ac
  - Full MTU packets can be transmitted (~2300bytes)
    - If full packet analysis is needed, analyze block-by-block
  - All 802.11 timings supported
    - Minimum distance between Txed frames is 0us
      - Note: channel can be completely taken by such firmware!!
  - Backoff implemented in software (fw)
    - Simply count slots and ask the HW to transmit

RX side

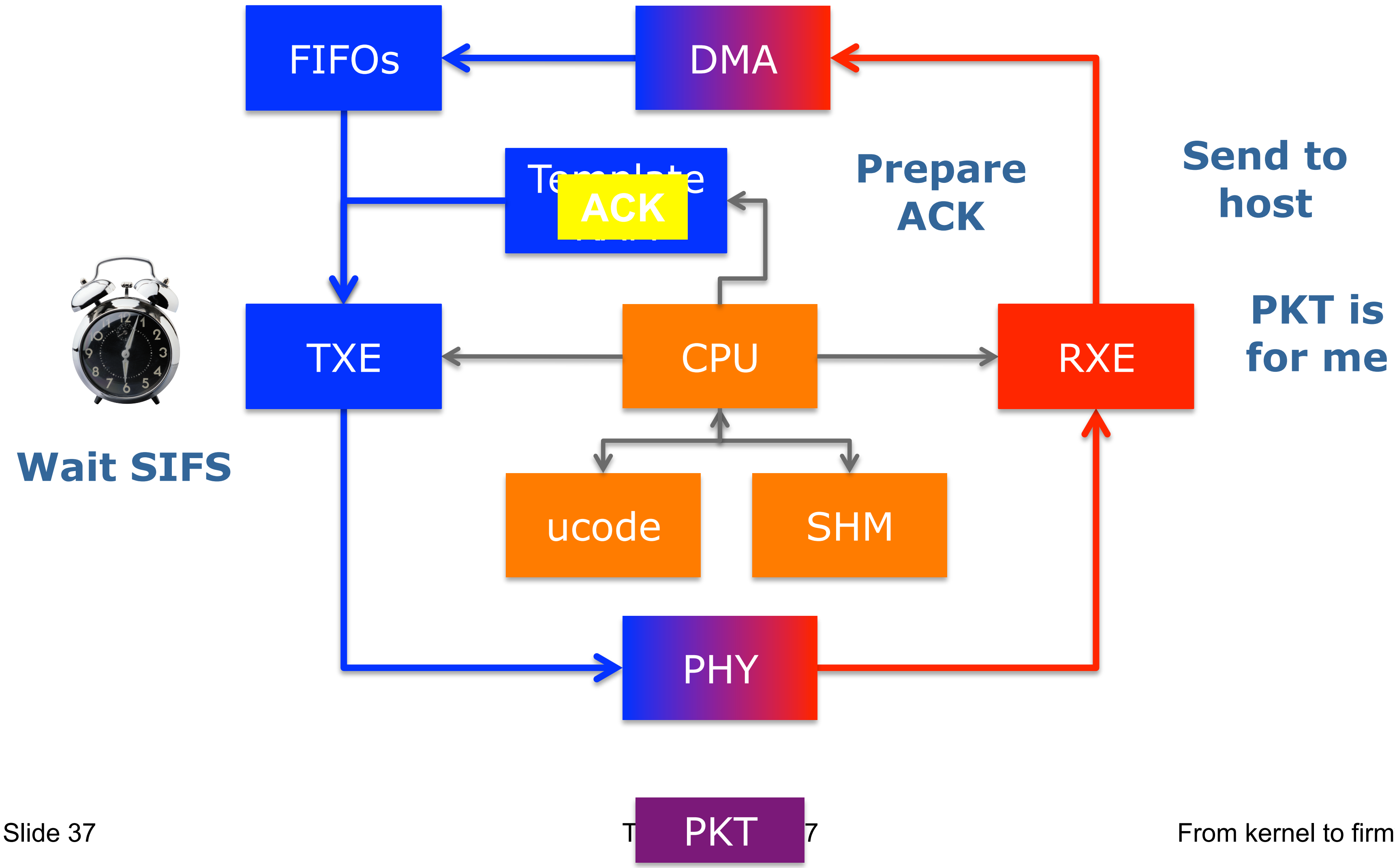
- Interface from AIR
  - HW acceleration for
    - PLCP and global packet FCS - Destination address matching
  - Packet can be copied to internal memory for analysis
    - Bytes buffered as soon as symbols is decoded
  - During reception and copying CPU is idle!
    - Can be used to offload other operations
    - Try to suggest something
  - Packets are pushed to host/kernel
    - If FW decides to go and through one FIFO ONLY
    - May drop! (e.g., corrupt packets, control…)

Example:
TX a packet, wait for the ACK



DROPPED!

Example:
RX a packet, transmit an ACK



Prepare ACK

Send to host

PKT is for me

Wait SIFS

What lesson we learned

- From the previous slides
  - Time to wait ack (success/no success)
  - Dropping ack (rcvd data not dropped, goes up)
  - And much more
    - When to send beacon
    - Backoff exponential procedure and rate choice
  - Decided by MAC processor (by the firmware)
- Bottom line:

Hardware is (almost) general purpose

From lesson to OpenFWWF Description of the FW

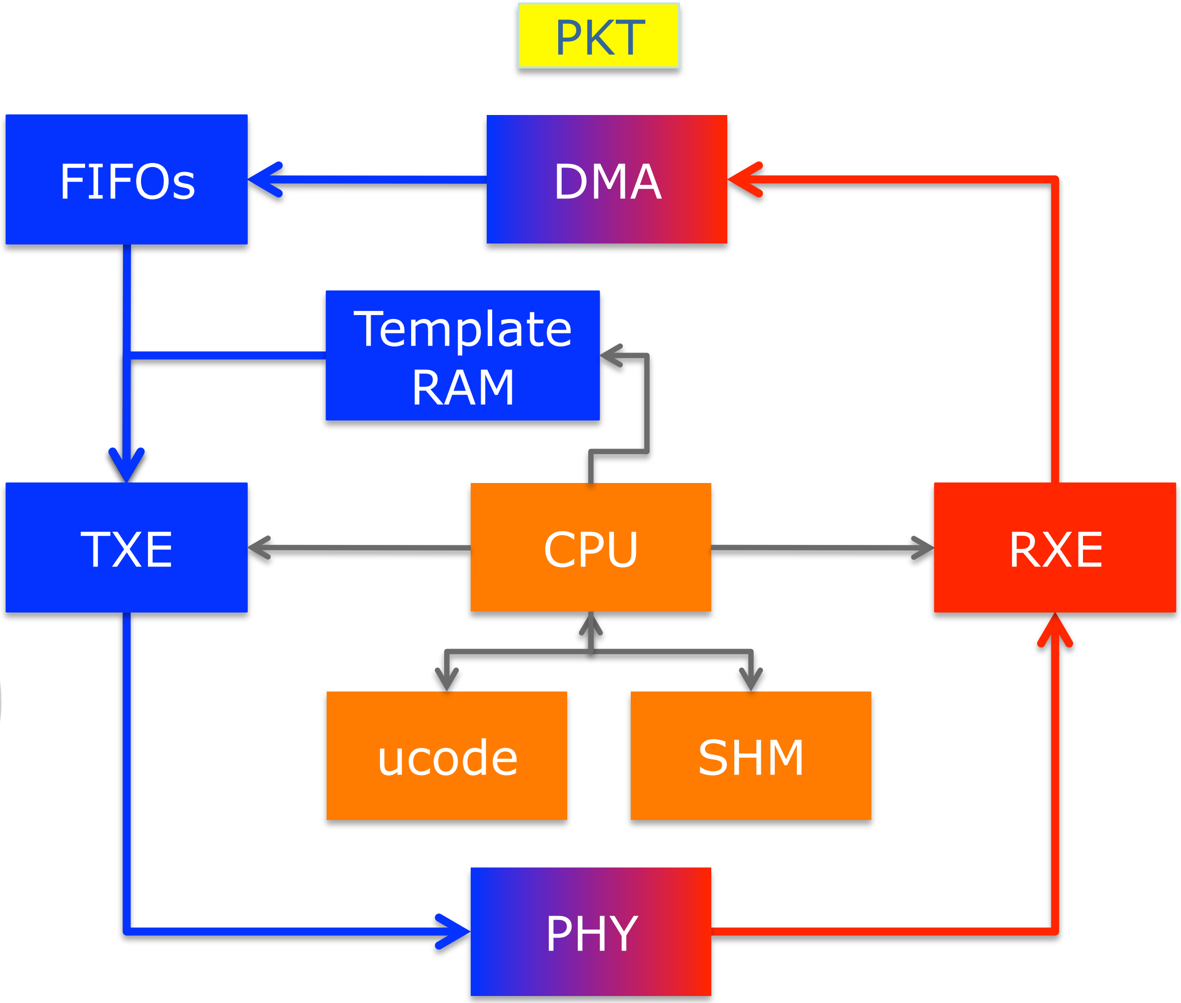
- OpenFWWF
  - It’s not a production firmware
  - It supports basic DCF
    - No RTS/CTS yet, No QoS, only one queue from Kernel
  - Full support for capturing broken frames
  - It takes 9KB for code, it uses < 200byte for data
    - **We have lot of space to add several features**
- Works with 4306, 4311, 4318 hw
  - Linksys Routers supported (e.g., WRT54GL)

Broadcom AirForce54g Simple TDM

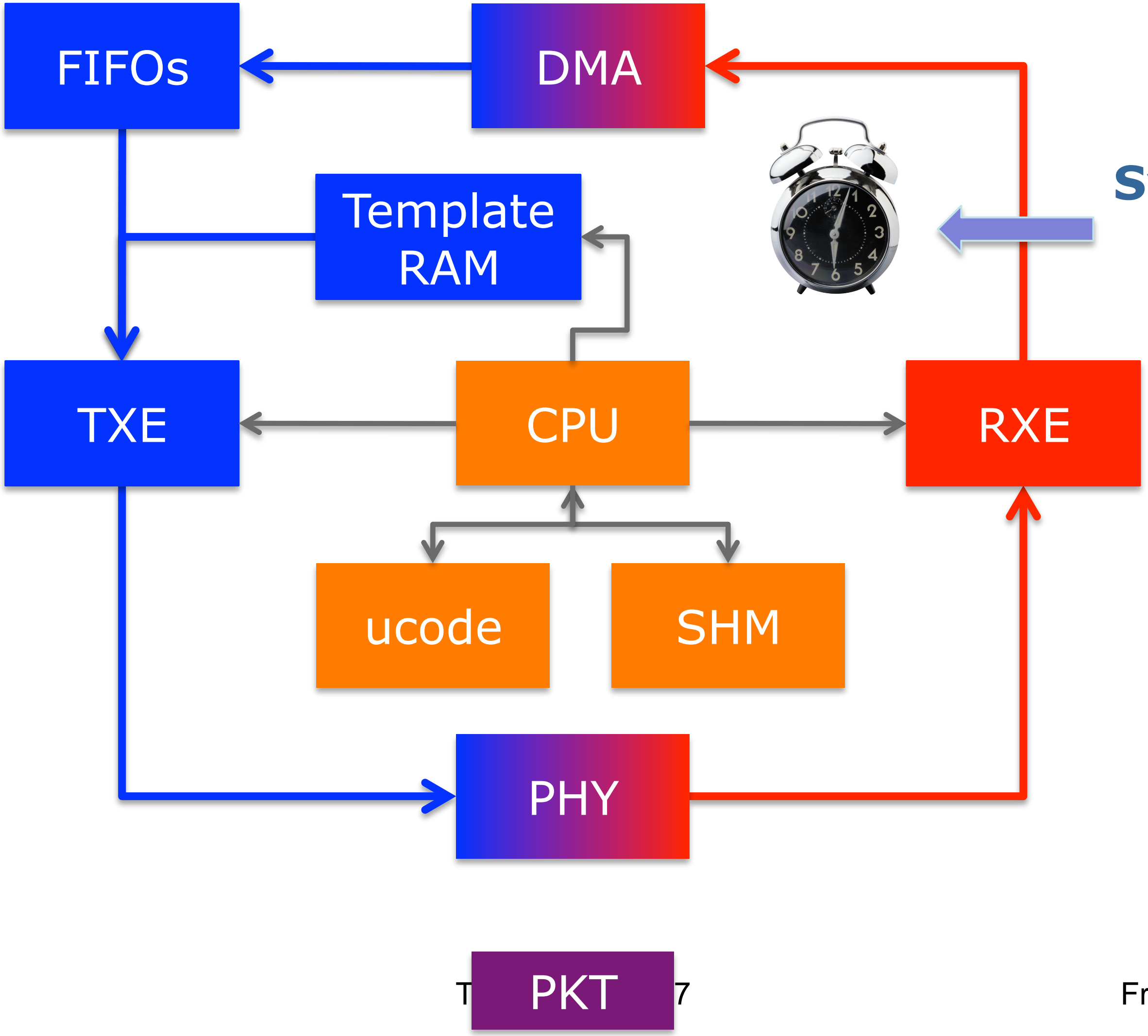
TDM needed! Waiting turn

3

GO!



Broadcom AirForce54g Simple TDM/2



Sync the clock

PKT from TDM domain