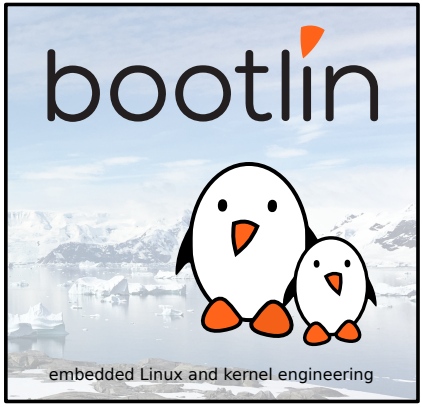
Multi-port and Multi-PHY Ethernet interfaces

Maxime Chevallier
maxime.chevallier@bootlin.com



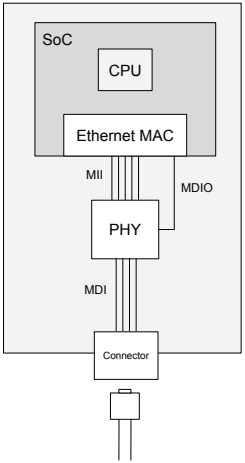
Maxime Chevallier

- Embedded Linux engineer at Bootlin
  - Embedded Linux **expertise**
  - **Development**, consulting and training
  - Strong open-source focus
- Open-source contributor
- Living near **Toulouse**, France

What is this all about ?

- Low-level aspects of an ethernet link, down to hardware layout of a machine
- Most of this comes from a variety of embedded use-cases
- NICs here are very not-smart and all internal components are configured by the kernel
- The Ethernet link isn't always the most important part of the product
- When it is, the requirements can be very specific

Typical embedded hardware design



- MAC : Represented by a Network Interface : `struct net_device`
- PHY : Handled by **phylib** : `struct phy_device` through `netdev->phydev`
- Port (Connector) : Information about it in `phy_device.port` and through `link_modes`

Variants

- The PHY can be integrated in the SoC or MAC
- The PHY might not exist at all
- The PHY can be handled by a firmware
- The Port isn't always BaseT (twisted copper pairs)
- The Port can be internal (backplane ethernet)

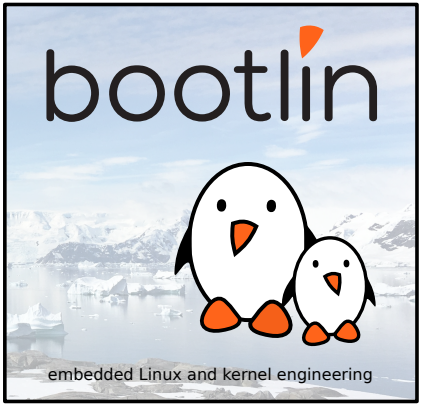
Userspace interaction

Configuring the link parameters and operations

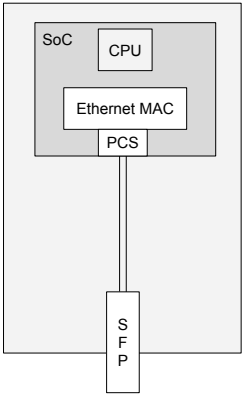
- Through `netlink` or `ioctl`, the PHY is hidden behind the interface
  - `ethtool --cable-test eth0`
- Modern PHYs can do more than transmit data
- `plca`, `cable_test`, `link_state`, `ts_info`, `pse` and statistics gathering needs PHY access.
  - Such commands rely on the `net_device.phydev` pointer
- The connector is also mostly abstracted away, and only the protocol is considered
  - We don't know much about the physical connector type
- The port can in theory be selected by switching between `PORT_FIBRE` and `PORT_TP`

Multi-PHY support

Maxime Chevallier
maxime.chevallier@bootlin.com

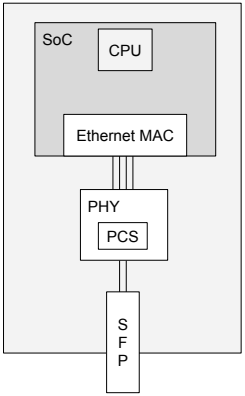


Typical embedded hardware design, with SFP



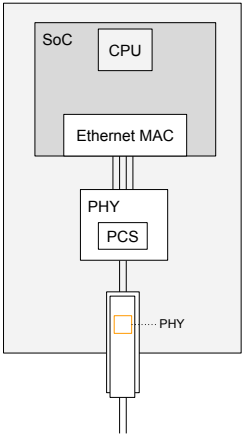
- We see more and more design with SFP cages in embedded
- SFP needs a serialized input (SGMII, 1000BaseX, BaseK)
- Serialization and encoding is done through a PCS component
- Some SoCs include a PCS within the Ethernet controller
- No PHY needed in that case

Media converter



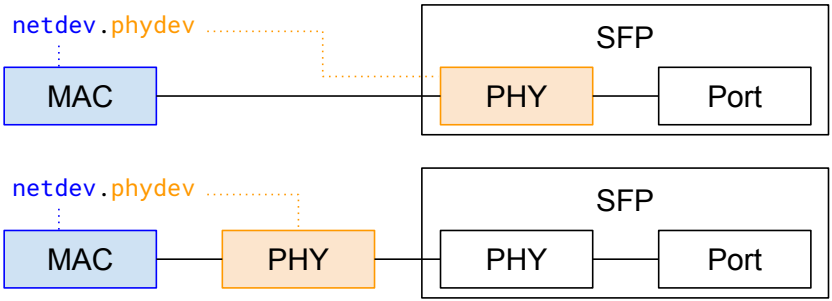
- Some SoCs don't have a PCS and output interfaces like `RGMII`
- Some PHYs can be used as a **media converter** to leverage their internal PCS
  - This not the same as a standalone `PCS`, which can also sit on an MDIO bus
- The media converter can be seen as an Ethernet PHY and is handled by **phylib**

SFP modules



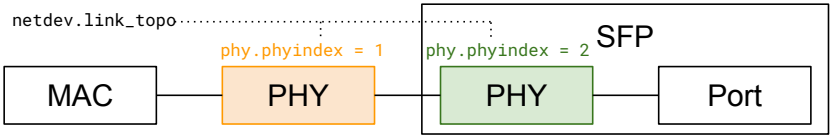
- SFP modules can embed a PHY
- The SFP module's PHY is also handled by **phylib**
- It has a reference to its **upstream** component
  - If there's a media converter : upstream is a PHY
  - If not, upstream is **phylink** : a MAC with a PCS
  - We can have in total 2 PHYs on the link

Limitations



- `net_device.phydev` points to the **innermost** PHY
- We can't always perform PHY-specific operations on the SFP PHY

link_topology and userspace API



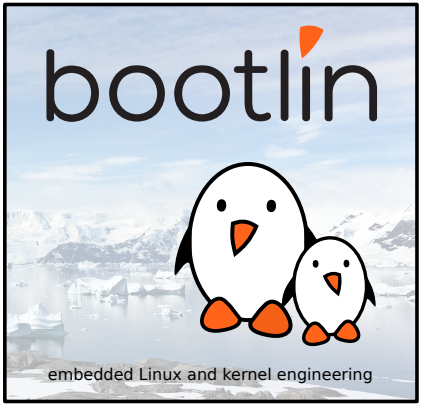
- Keep track of PHY devices and the overall topology
- Assign a unique identifier to each PHY, similar to `ifindex`
- This index is used for PHY-specific netlink commands
- Fallback to `netdev.phydev` if no PHY index is set
- Allow passing **PHY index** in the **ethnl** header.

Kernel API

- `link_topology` lists `phy_device_node` that represents phys
  - `phy_device_node` references the PHY and its parents
  - It avoids maintaining the topology info within the `phy_device`
- They are attached to a netdev but can be used without
  - Some PHYs aren't attached to any `net_device` : DSA shared ports
- `link_topo_add_phy`, `link_topo_del_phy`, `list_topo_get_phy`
- Hooks into `phy_attach_direct` and `sfp_connect_phy`

Multi-port support

Maxime Chevallier
maxime.chevallier@bootlin.com

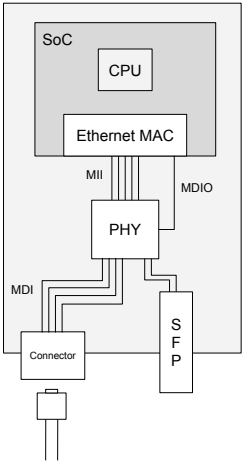


Focus on the port

The Port designs an output of the MAC or the PHY, it can be `internal` of `external`

- In can be connected to a physical connector (`8P8C`, `SFP`, `coax`, ...)
- A port can only support a number of protocols, serialized or not
  - Serialized interfaces have a number of `lanes`
- A port might have some status LEDs
- A port can also perform auxilliary functions, such as `PoE`

Multi-port interfaces

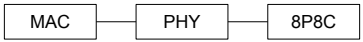


- PHYs can expose multiple physical interfaces
- The first that has the link, gets the link
  - Sometimes called **auto-media selection**
- Supported on the Marvell 88x3310 PHY, the 88e6390X switch, and much more
- Still per-driver behaviour, and few information and control

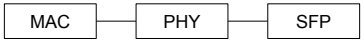
Internal API

- Introduce `phy_port` to represent one port
  - Internal or External
  - link_modes and link_state
  - LEDs ? PoE ?
- Sane default is for a PHY to have one port, but they can register more
- Ethernet drivers can also register ports
- An SFP cage is also a port
  - Switches from `external` to `internal` when inserting a RJ45 transceiver
- Also use `link_topology` to track ports and their location
- `link_topo_add_port`, `link_topo_del_port`, `link_topo_get_port`
- Exposed through **netlink**

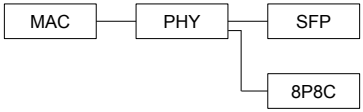
Devicetree port representation today




```
ethernet-phy@0 {
    reg = <0>;
};
```




```
ethernet-phy@0 {
    reg = <0>;
    sfp = <&sfp0>;
};
```




```
ethernet-phy@0 {
    reg = <0>;
    sfp = <&sfp0>;
    // No indication about the 8P8C presence
};
```

phy_port in devicetree

- We assume PHYs have one BaseT port by default for compatibility
- Having an SFP phandle isn't enough to know how many ports are physically connected
- devicetree description of the ports are needed
- Also useful for PHYs that have multiple possible configurations
  - e.g. Marvell's 88e1543 can either be a Quad-PHY or a Dual dual-port PHY
  - Also avoids vendor-specific media-converter settings
- It can be also used for LED description and PoE

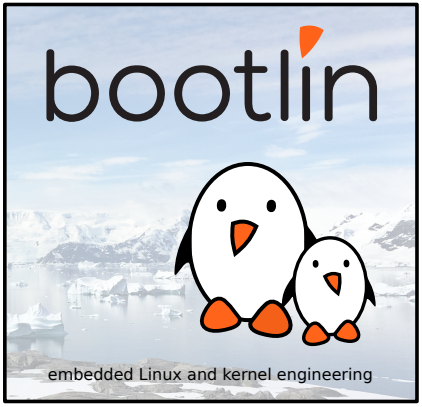
Devicetree example - WIP

example.dts

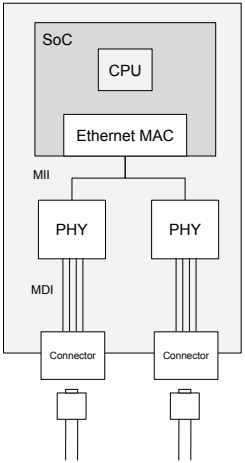
```
ethernet-phy@0 {
 ...

  mdi {
    port@0 {
      media = "10baseT", "100baseT", "1000baseT";
    };

    port@1 {
      sfp = <&sfp>;
    };
  };
};
```

Multiplexing support

Maxime Chevallier
maxime.chevallier@bootlin.com

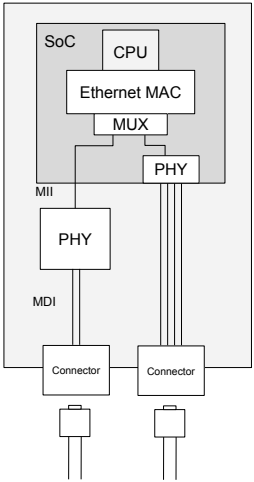


Multi-PHY Link, for redundancy

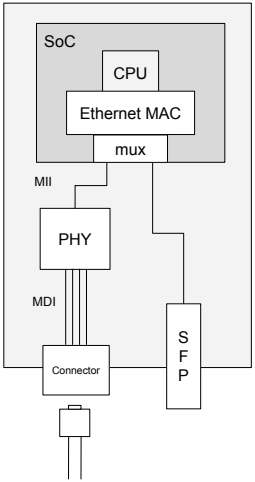


- There are other use-cases where multiple PHYs are on the link
- Multiple PHYs connected on the same `MII`
  - Non standard but found in the wild
  - The Kernel is in charge of managing PHYs
  - Isolate mode or Poweroff
- Achieve link redundancy
- Use multiple link standards
  - BaseT1 + BaseT4
- We model a **Software MUX**

Hardware Muxing

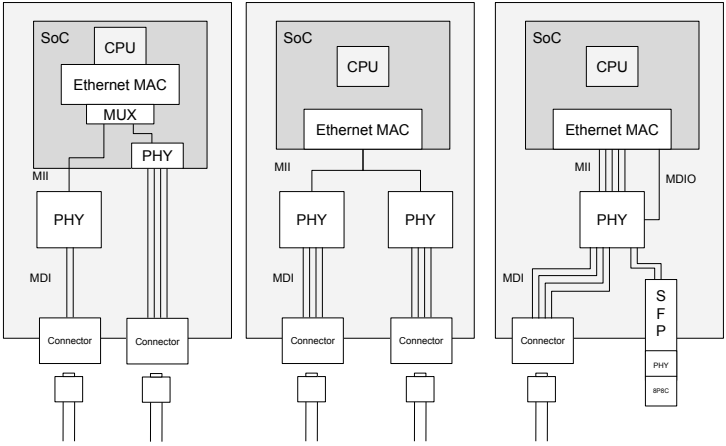


- Internal PHY
- Internal Mux
- External PHY

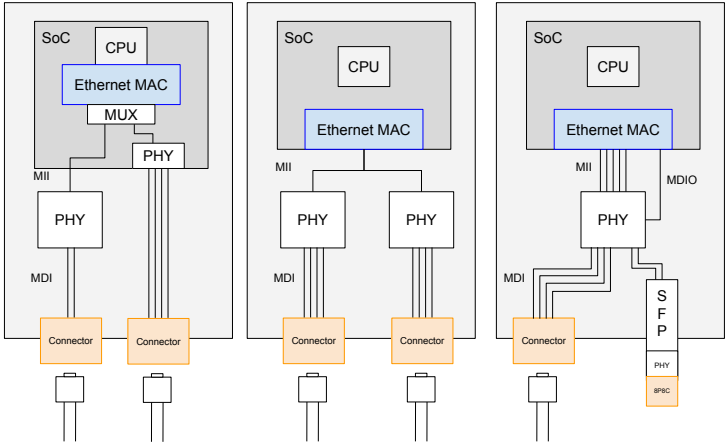


- Internal Mux
- Control via
  - Registers
  - GPIO
- External Lanes
- External MII

Userspace API design



Userspace API design



User cares about which front-facing-ports are on a given interface

Port Multiplexer

- The port multiplexer has a reference to all ports attached to an interface
- It's the component userspace interacts with
- Implements the logic for the port switchover
  - First that has link
    - Use PHY's **Auto media detection**
  - Preferred port
  - Userspace only (no automatic port enabling)
  - Can be extended (speed based...)

Internal APIs

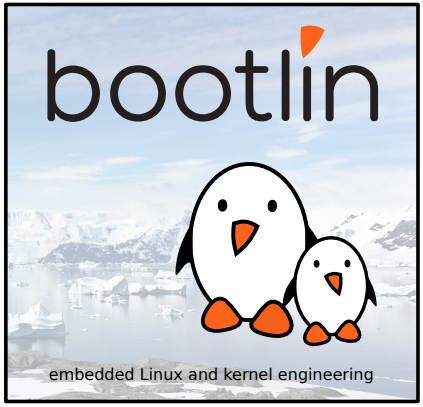
- PHYs that have multiple ports implement muxing ops
- We need to make sure the PHY configuration is coherent
  - We can't enable TS offload on one PHY and not on the other
  - However, cable-testing is fine
- We also need to ensure link configuration coherency
  - Autoneg or speed settings can differ between ports
  - Per-port configuration ?

Userspace API

- From userspace, what's important is mostly which **port** is selected
- The API therefore leverages the `phy_port_index`
- Introduce a netlink command set `PORT_SELECT_GET` / `PORT_SELECT_SET`
  - auto
  - manual
  - speed ?
- Indroduce the `phy_port.preferred` attribute, set to `false` by default
- Indroduce the `phy_port.enabled` attribute, set to `true` by default

Upstream Status

Maxime Chevallier
maxime.chevallier@bootlin.com



Where's the code ?

- PHY enumeration series : Very basic RFC V1 sent, V2 incoming
- Port enumeration series : RFC ready
- Isolate-mode support : Still testing
- Multiplexing support : Still testing

What's next ?

- Timestamping :
  - Leverages `multi-phy` support for SFP case
  - Select the hardware timestamping source
- PoE :
  - PoE controllers assign power-budgets per-port
  - Leverages `phy_port` support
- DSA : There are some blind spots left, such as shared DSA port PHYs

Questions? Suggestions? Comments?

Maxime Chevallier
maxime.chevallier@bootlin.com

https://bootlin.com/pub/conferences/2023/netdev/multi-port-multi-phy-interfaces.pdf