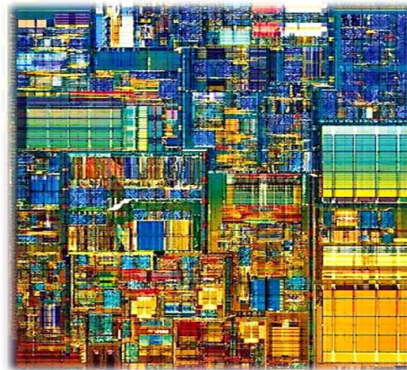




**INSTITUTO POLITECNICO NACIONAL**  
**CENTRO DE INVESTIGACION EN COMPUTACION**  
**LABORATORIO DE MICROTECNOLOGIA Y SISTEMAS EMBEBIDOS**

---

## Alligator\_OS: An embedded OS



Adrian Alonso

<aalonso00@gmail.com>

January 2011



# Rights to copy

## Attribution – ShareAlike 3.0

You are free to copy, distribute, display, and perform the work

to make derivative works

to make commercial use of the work

Under the following conditions

Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

License text:

<http://creativecommons.org/licenses/by-sa/3.0/legalcode>





# Tutorial Committee

---

Dr. Marco Antonio Ramírez Salinas (Director)

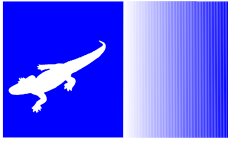
Dr. Hugo César Coyote Estrada (Director)

Dr. Jose Luis Oropeza Rodríguez

Dr. Luis Alfonso Villa Vargas

M. en C. Osvaldo Espinosa Sosa

M. en C. Sergio Sandoval Reyes



## Alligator\_OS: embedded operating system

---

General objective:

Hardware/software co-design approach for re-configurable architectures based on the Xilinx ML507 PowerPC 440 platform.

This thesis aims to find a reliable solution for bringing the Linux operating system on embedded devices, focusing on Xilinx re-configurable architectures as a base platform.



## Alligator\_OS: embedded operating system

---

A field-programmable gate array (FPGA) is a programmable logic integrated circuit designed to be configured after manufacturing, hence "field-programmable".

A common design pattern is to pair an FPGA with a traditional CPU running an operating system like Linux. Applications are run on the CPU while the FPGA implements hardware interfaces and offloads some data processing.



# Alligator\_OS main project task

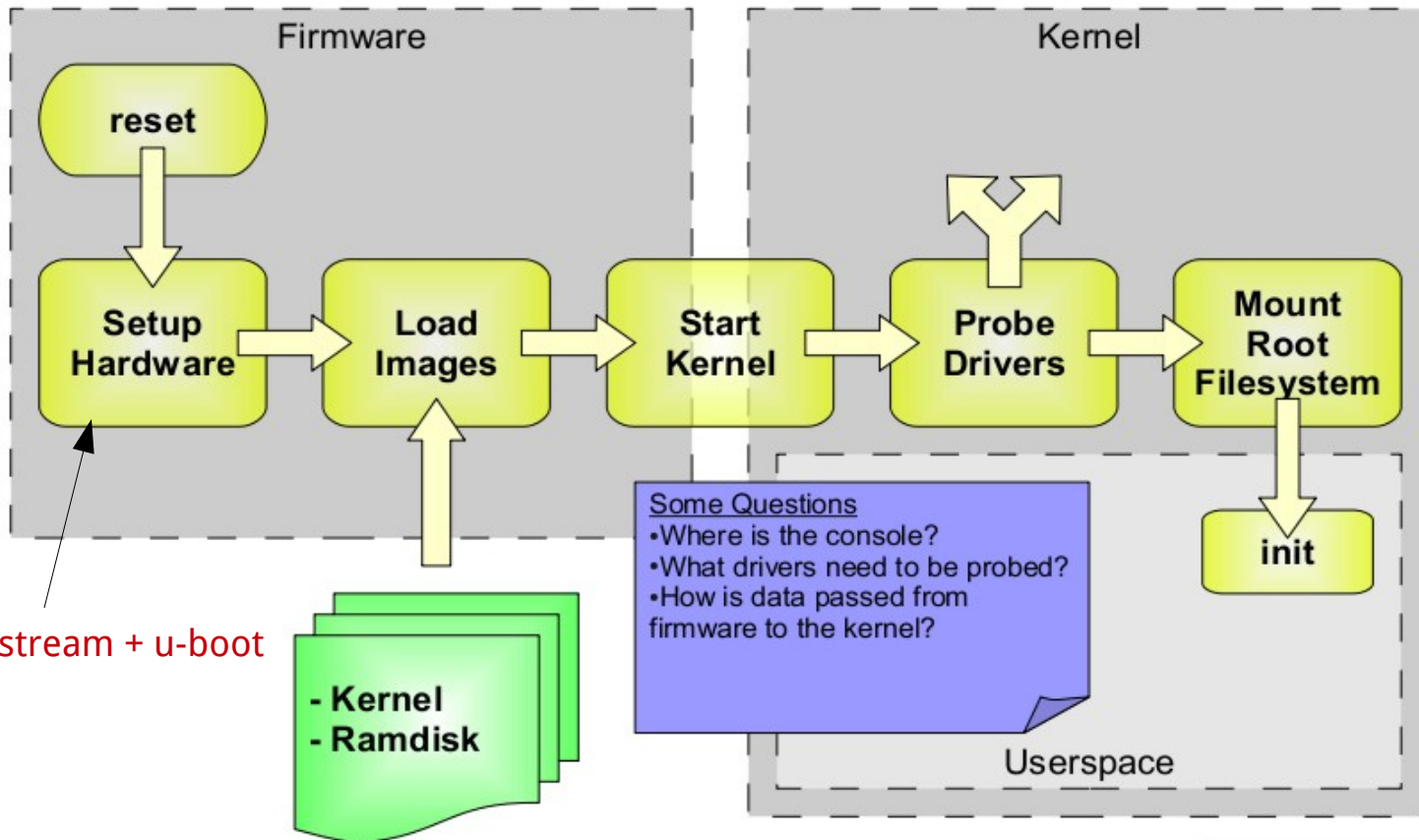
---

- **Part 1:** Embedded Linux board bring up for Xilinx ML507 target platform (PowerPC 440).  
OE/Poky build frameworks
- **Part 2:** Custom hardware implementation and integration to the system bus.
- **Part 3:** Linux device drivers for custom hardware  
Input device, emulate scroll wheel found on standard mice



# Getting to know the ML507 target

- Identify core elements to boot Linux



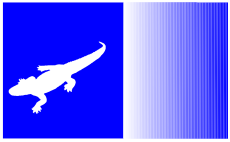
Ace = bitstream + u-boot



# Getting to know the ML507 target

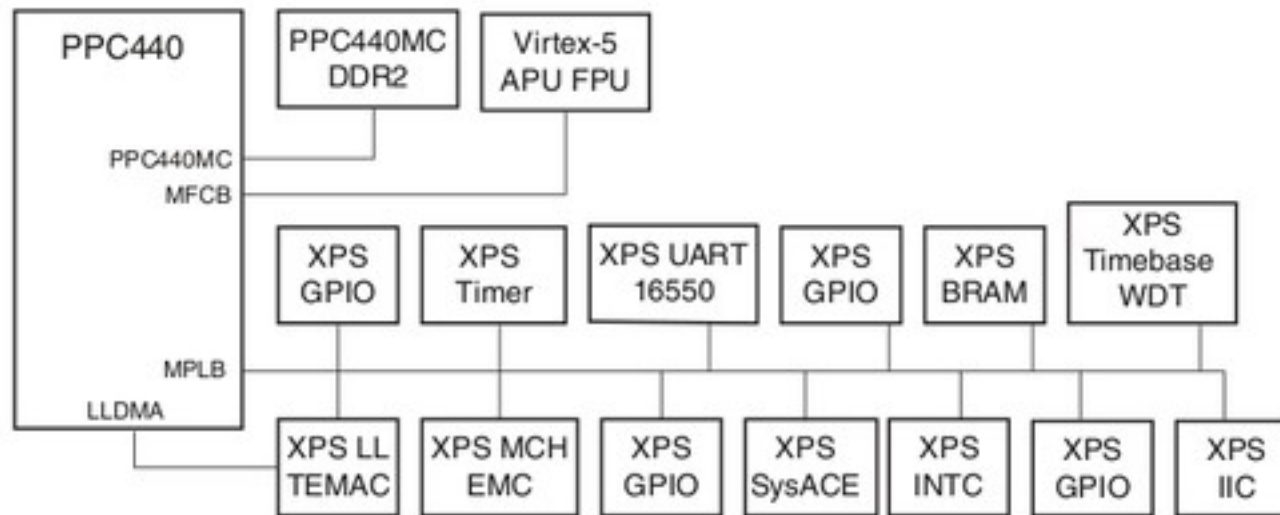
- Design flow, hardware configuration





# Getting to know the ML507 target

- Create base hardware platform with XPS
  - `xps_tft` display; `xps_ps2` keyboard module added manually





# Embedded Linux distributions

- Prebuilt distributions

emdebian

Linaro



LiMo Foundation™

MeeGo™

- Build frameworks

BuildRoot  
Making Embedded Linux Easy



yocto  
PROJECT



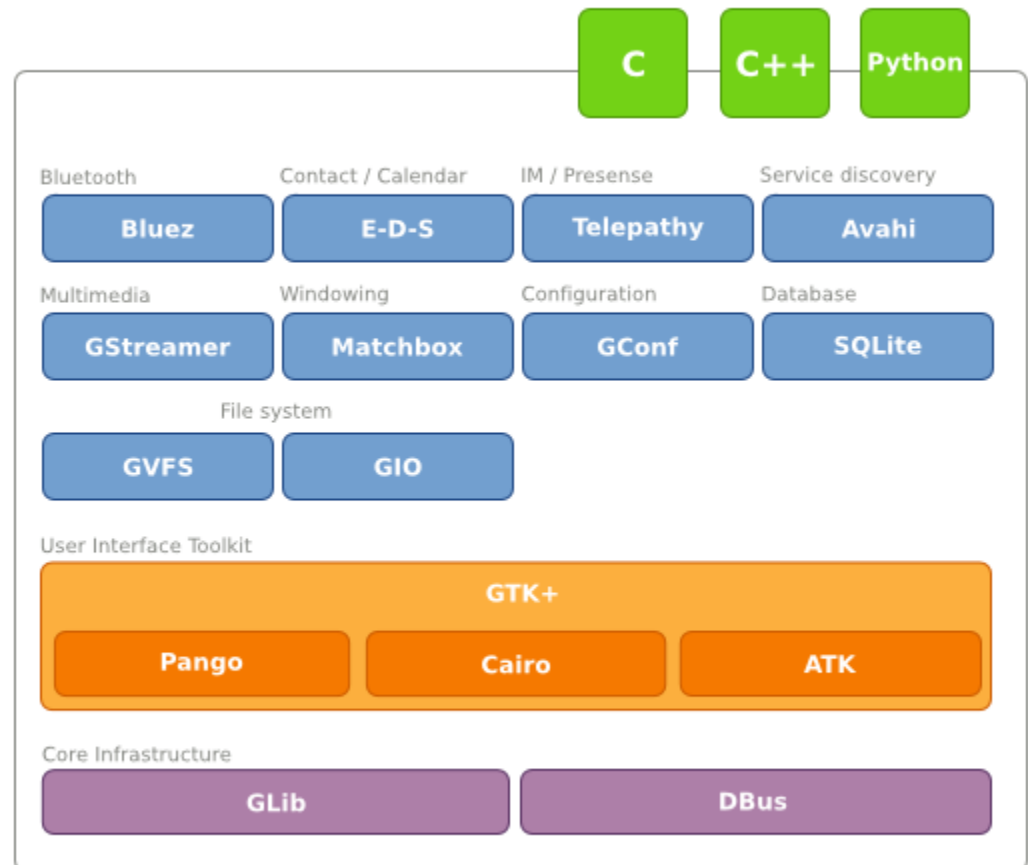
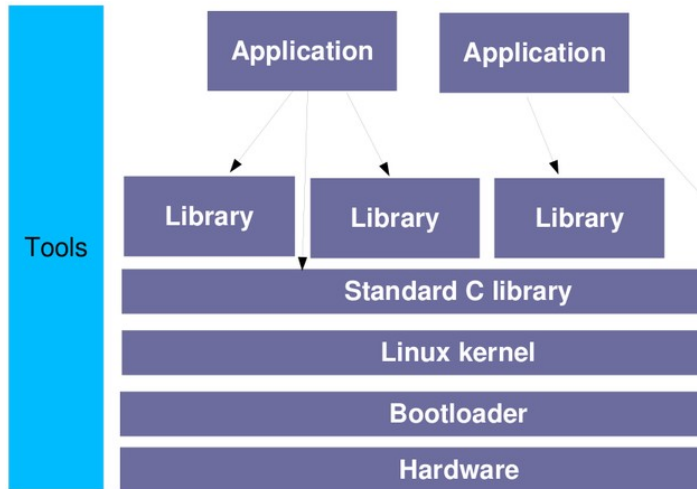
openembedded



OpenBricks  
Embedded Linux Framework



# Embedded Linux distributions





# Embedded Linux distributions

---

Unified build approach using Openembedded.

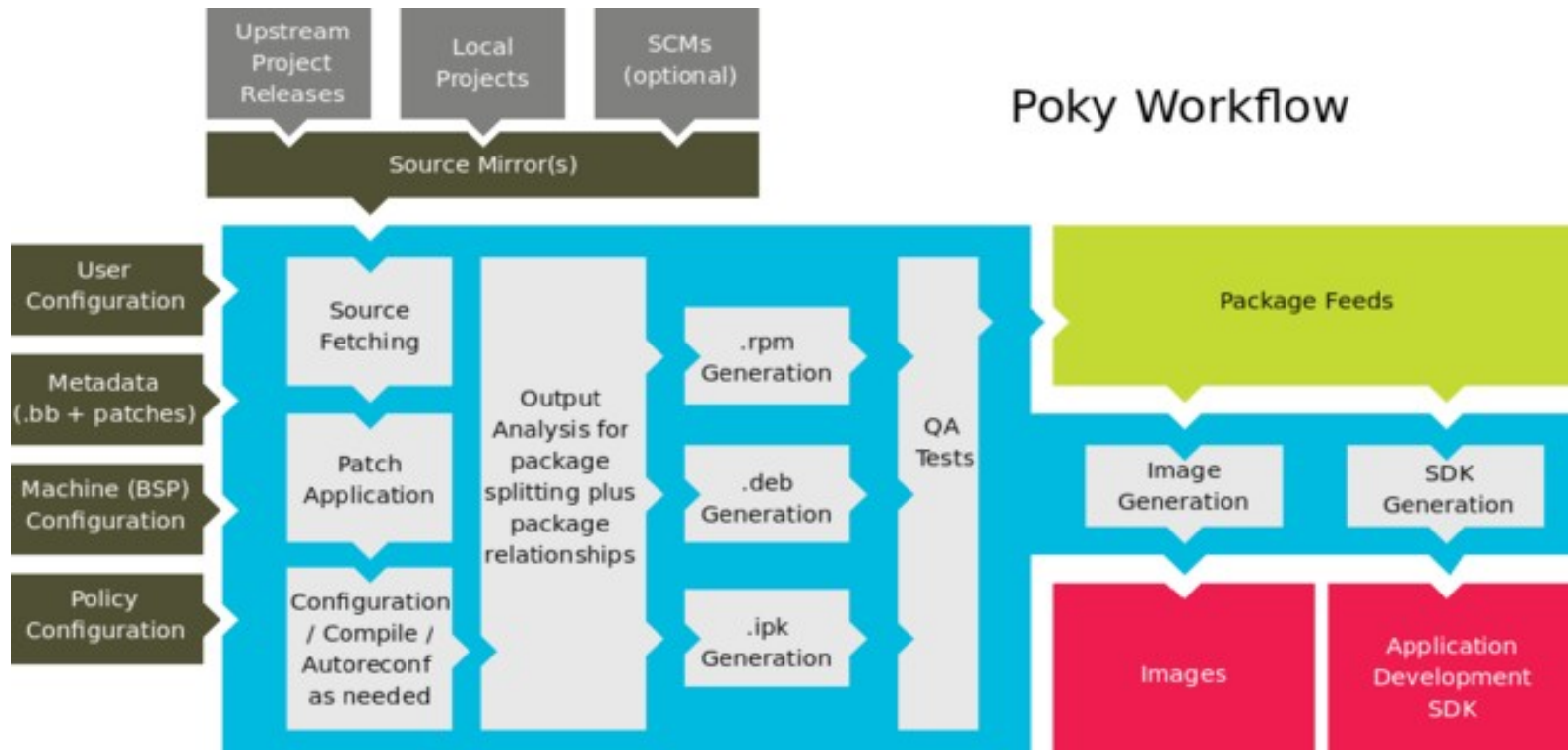
Openembedded (OE) provides the infrastructure to build a complete **Linux distribution** for any architecture that is supported by the Linux kernel. The user must **configure** the **target platform** and **select the collection of packages** for generating a complete **root file system**.





# Embedded Linux distributions

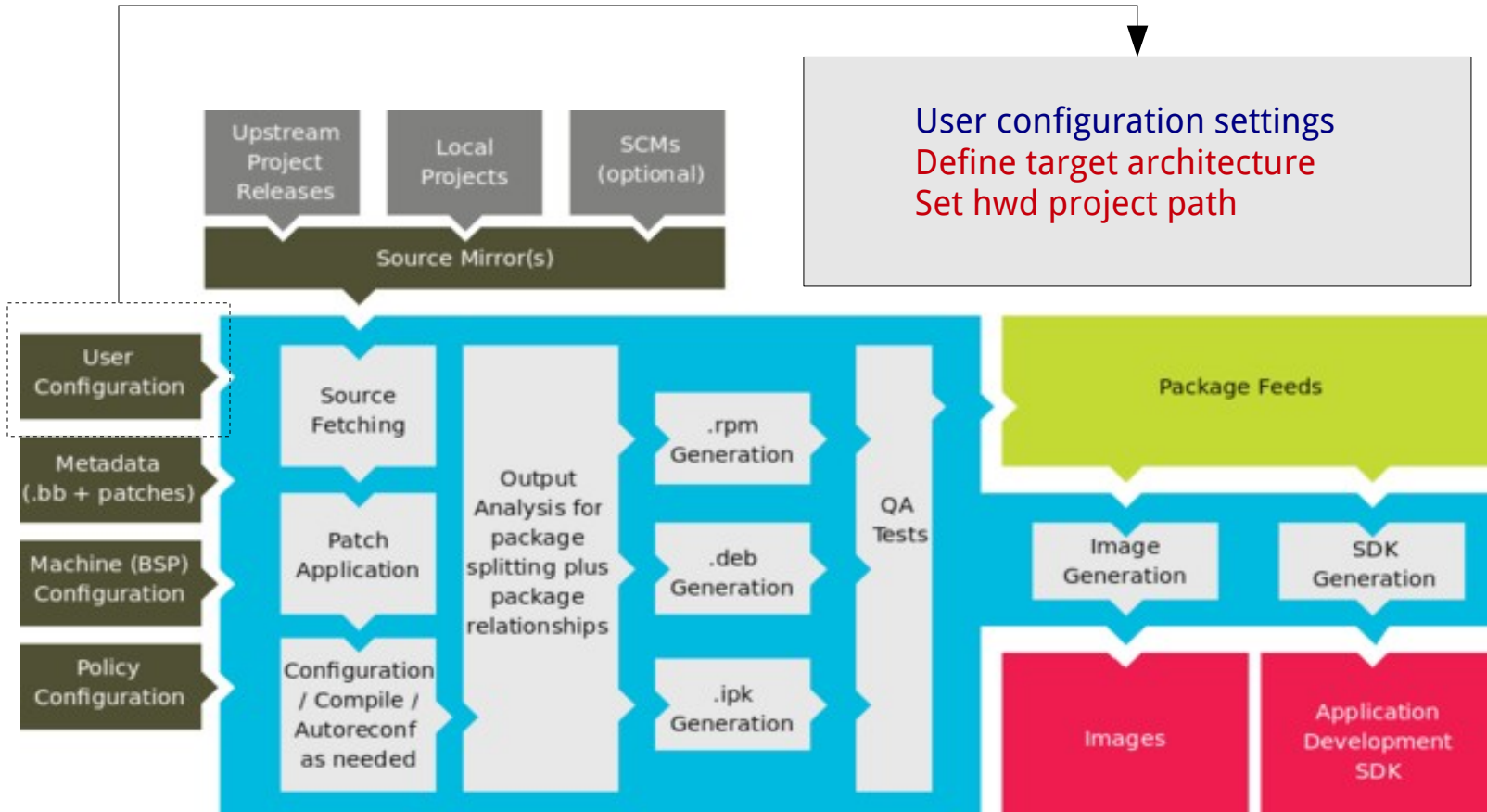
- OE/Poky architecture

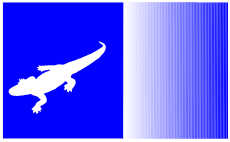


[Yocto project]

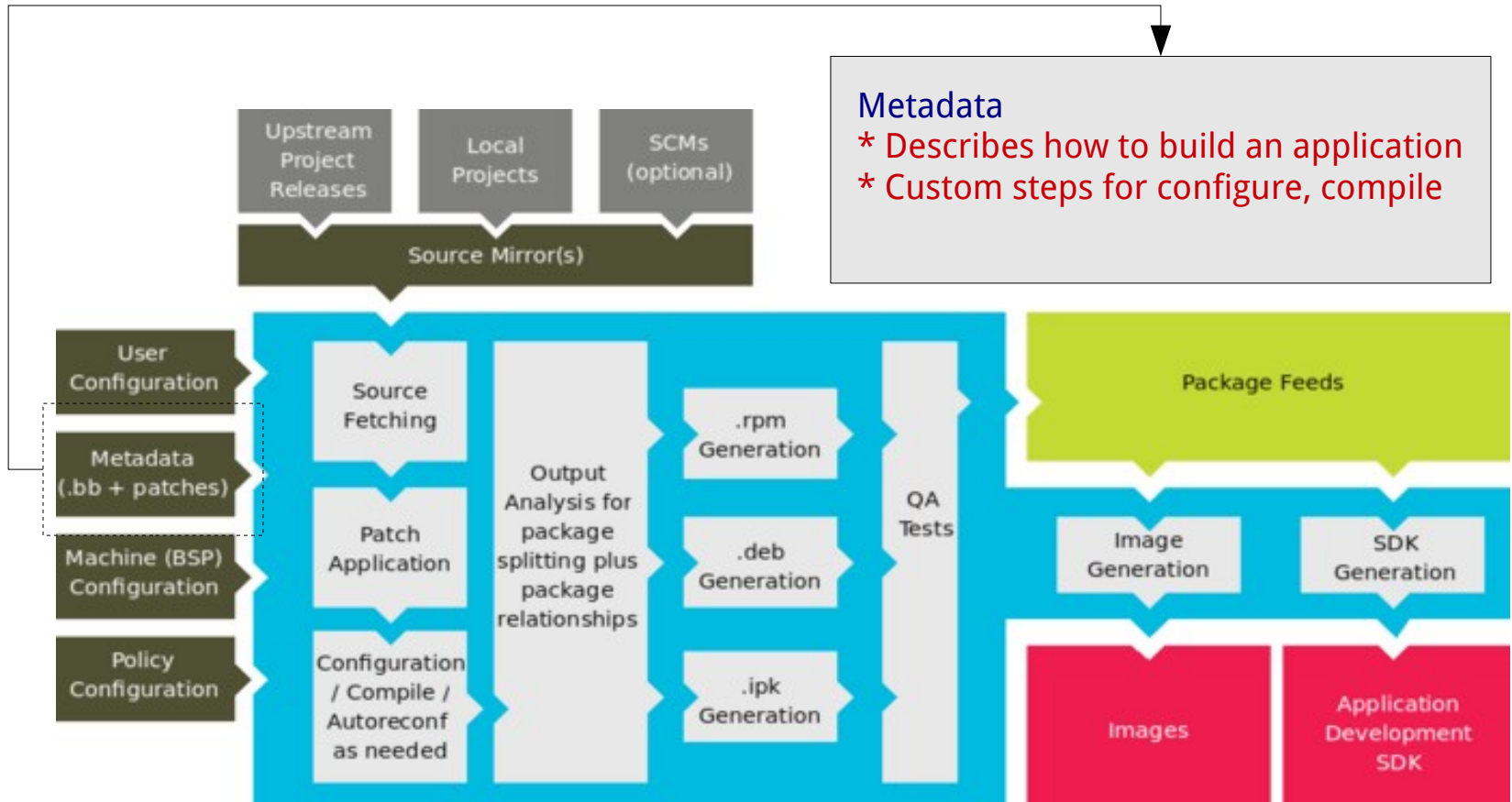


# Embedded Linux distributions



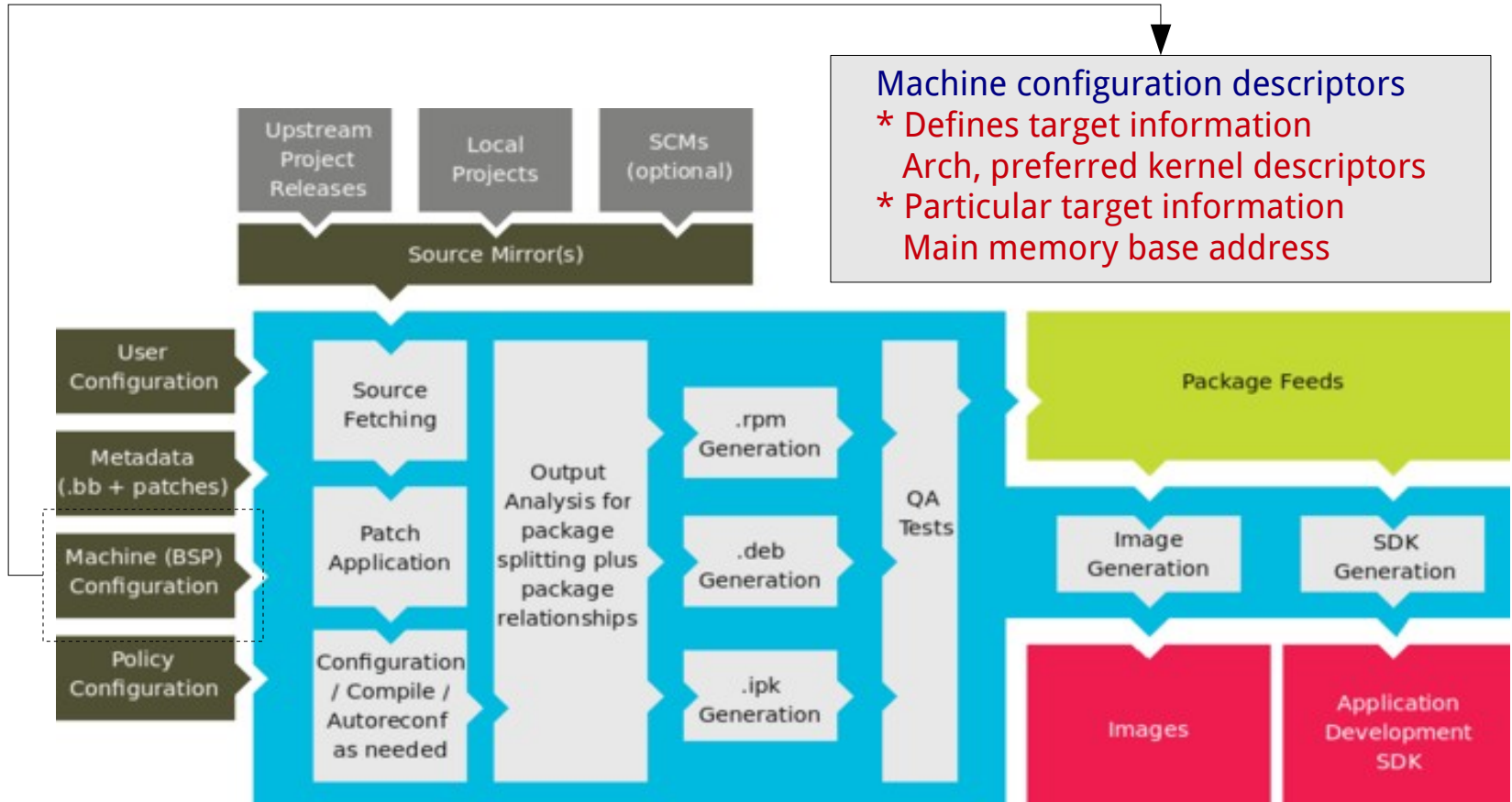


# Embedded Linux distributions



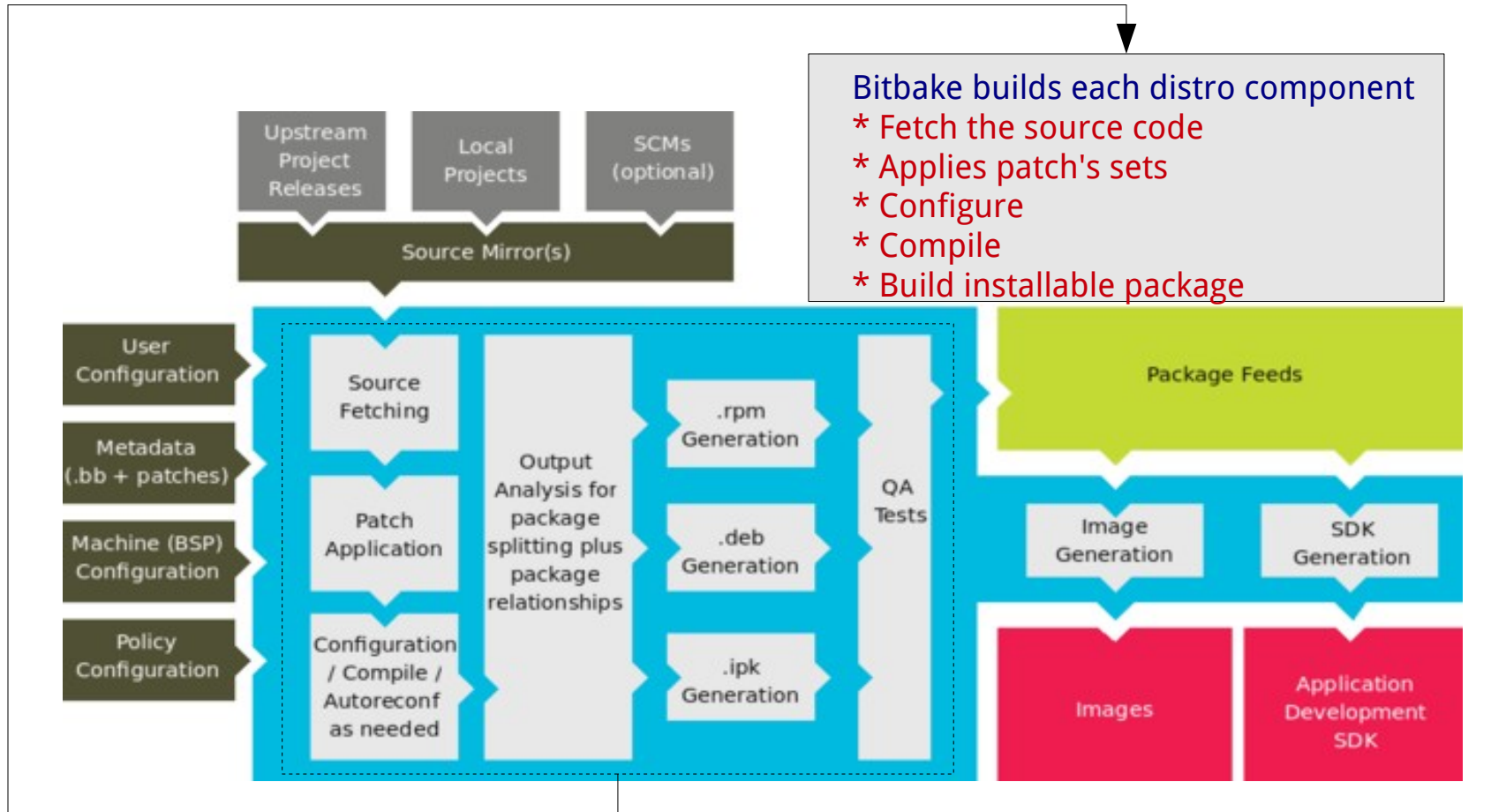


# Embedded Linux distributions



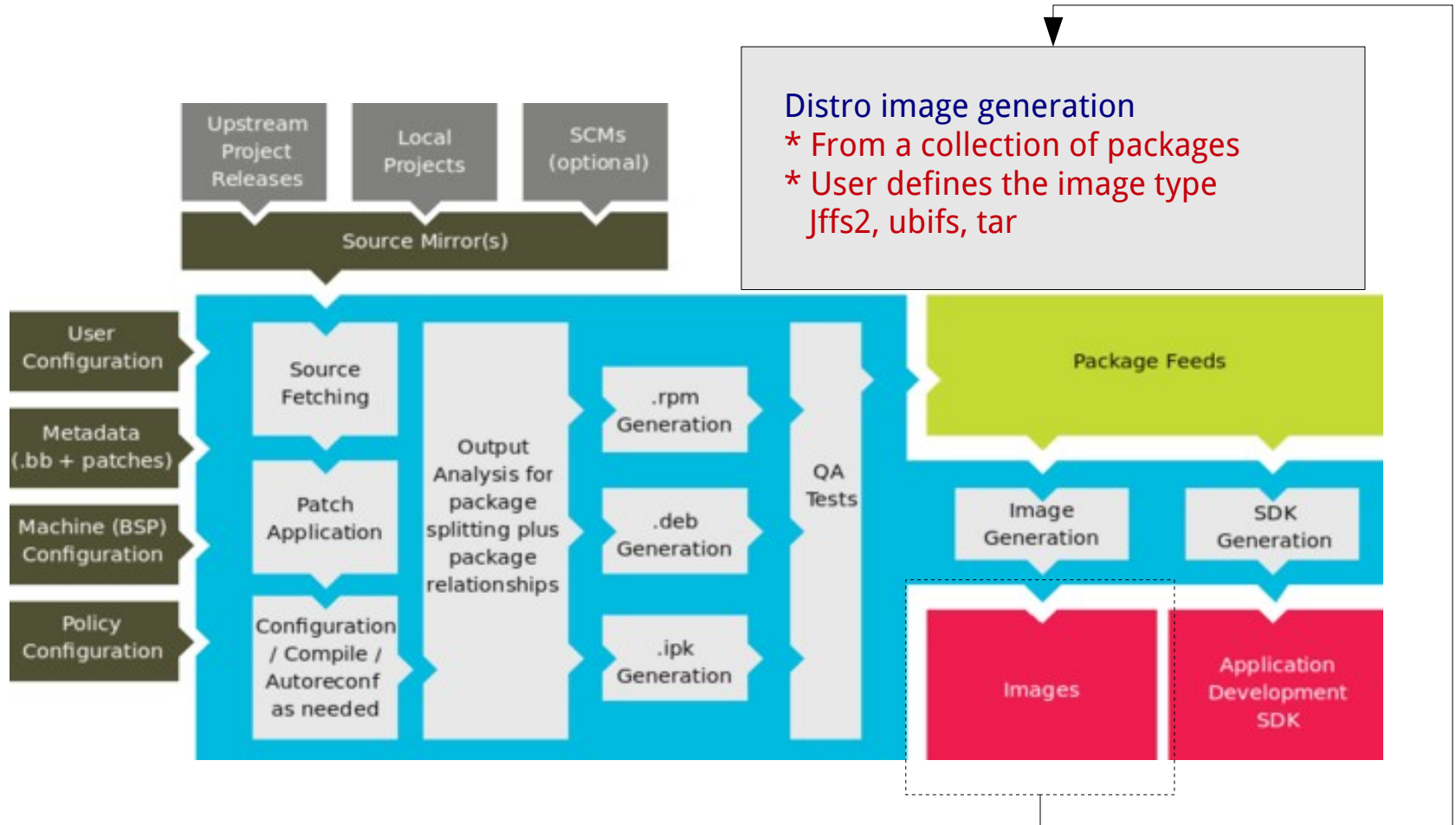


# Embedded Linux distributions





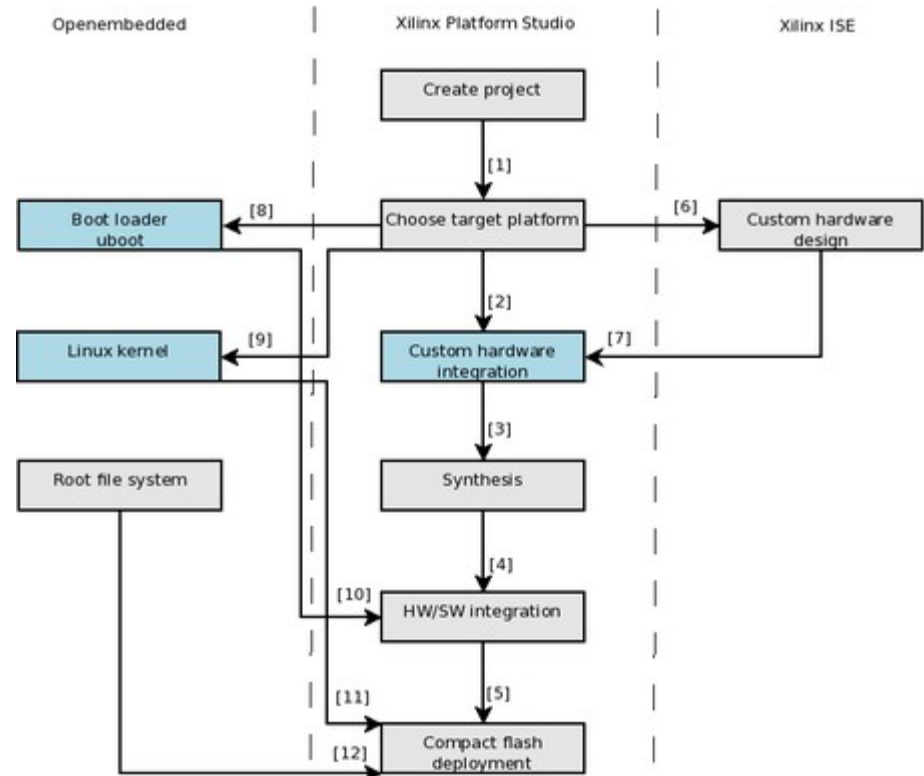
# Embedded Linux distributions





# OE/Poky frameworks

- Dealing with reconfigurable hardware
  - Build frameworks are more suitable to handle more frequent hardware platform changes.
- OE/Poky Hardware/software integration **automated**





# OE/Poky frameworks

## Dealing with reconfigurable hardware: device tree

```

SysACE_CompactFlash: sysace@83600000 {
    compatible = "xlnx,xps-sysace-1.01.a", "xlnx,sysace-1.01.a";
    interrupt-parent = <&xps_intc_0>;
    interrupts = < 6 2 >;
    reg = < 0x83600000 0x10000 >;
    xlnx,family = "virtex5";
    xlnx,mem-width = <0x10>;
};
xps_bram_if_cntlr_1: xps-bram-if-cntlr@ffff0000
    compatible = "xlnx,xps-bram-if-cntlr-1.00.b", "xlnx,xps-bram-if-cntlr-1.00.a";
    reg = < 0xffff0000 0x10000 >;
    xlnx,family = "virtex5";
};
xps_intc_0: interrupt-controller@81800000 {
    #interrupt-cells = <0x2>;
    compatible = "xlnx,xps-intc-2.00.a", "xlnx,xps-intc-2.00.a";
    interrupt-controller;
    reg = < 0x81800000 0x10000 >;
    xlnx,kind-of-intr = <0x15>;
    xlnx,num-intr-inputs = <0xe>;
};

```

ppc440_0	C_IDCR_BA...	0B0000000000	0B0011111111
DDR2_SDRAM	C_MEM_BA...	0x00000000	0x0FFFFFFF
Push_Buttons_5Bit	C_BASEADDR	0x81400000	0x8140FFFF
LEDs_Positions	C_BASEADDR	0x81420000	0x8142FFFF
LEDs_8Bit	C_BASEADDR	0x81440000	0x8144FFFF
DIP_Switches_8Bit	C_BASEADDR	0x81460000	0x8146FFFF
IIC_EEPROM	C_BASEADDR	0x81600000	0x8160FFFF
xps_intc_0	C_BASEADDR	0x81800000	0x8180FFFF
Hard_Ethernet_MAC_fifo	C_BASEADDR	0x81A00000	0x81A0FFFF
SysACE_CompactFlash	C_BASEADDR	0x83600000	0x8360FFFF
xps_timebase_wdt_0	C_BASEADDR	0x83A00000	0x83A0FFFF
xps_timer_0	C_BASEADDR	0x83C00000	0x83C0FFFF
RS232_Uart_1	C_BASEADDR	0x83E00000	0x83E0FFFF
FLASH	C_MEM0_BA...	0xFC000000	0xFDFFFFFF
Hard_Ethernet_MAC	C_BASEADDR	0xFF000000	0xFF07FFFF
xps_bram_if_cntlr_1	C_BASEADDR	0xFFFF0000	0xFFFFFFFF
ppc440_0	C_SPLB0_R...		



# OE/Poky frameworks

- Generating a Linux distro
  - Simple
  - Reproducible
  - Reliable
- All changes are in upstream projects; making this work available to every one.

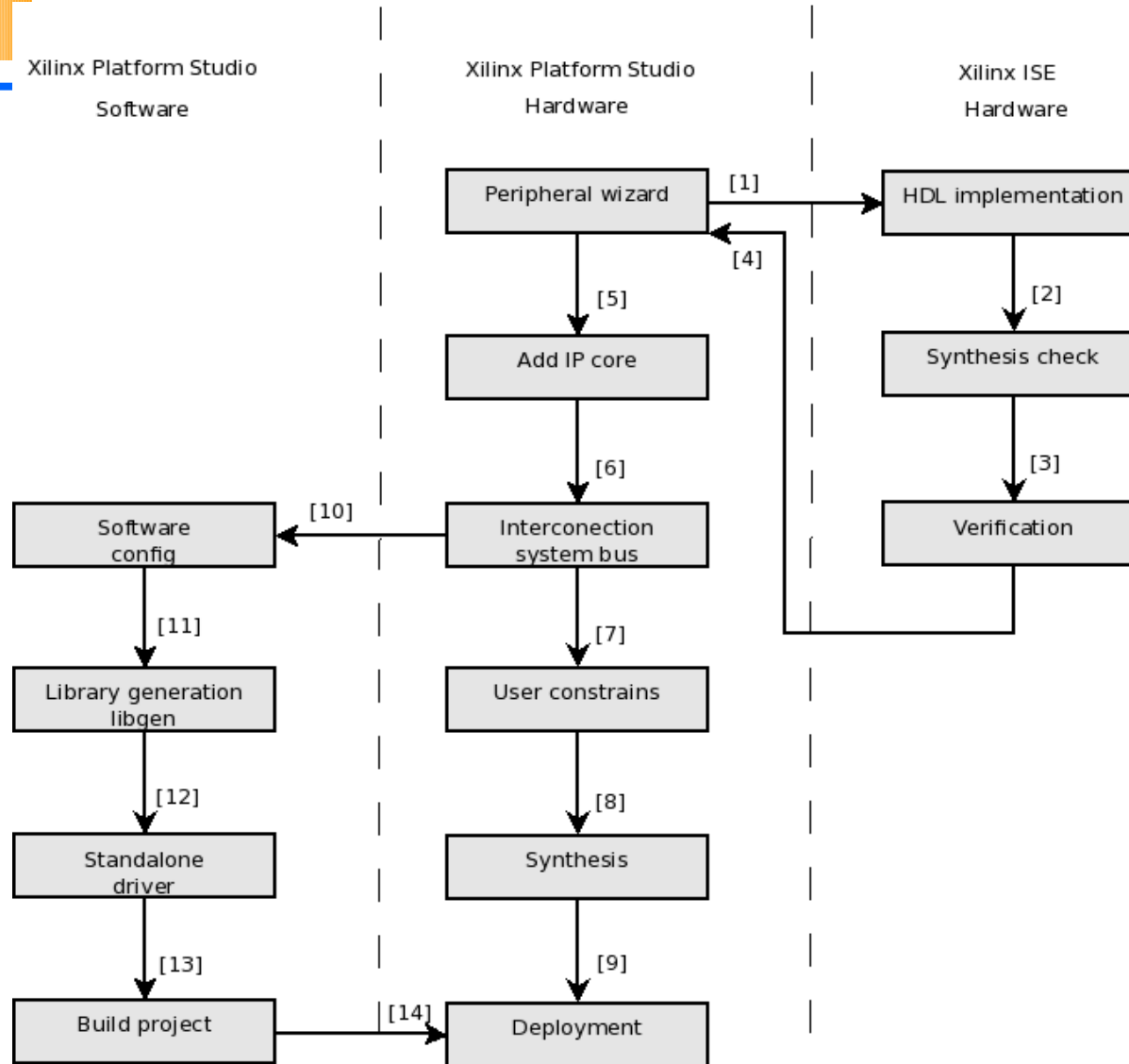
yocto  
PROJECT







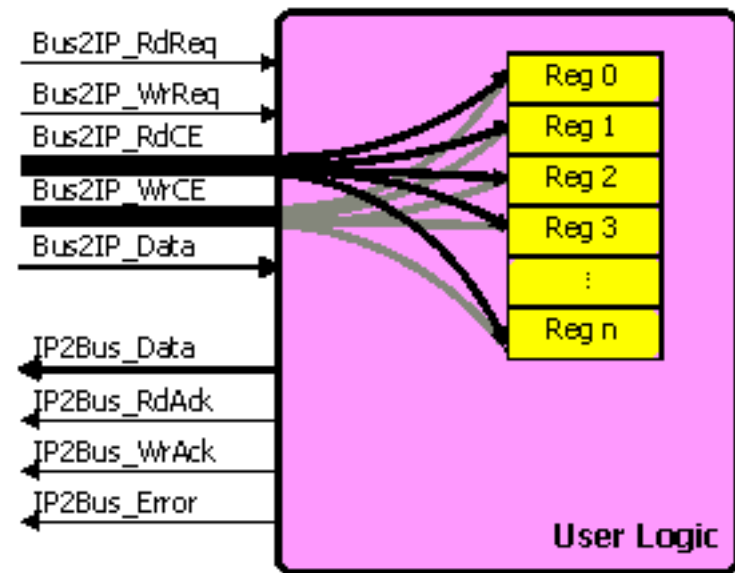
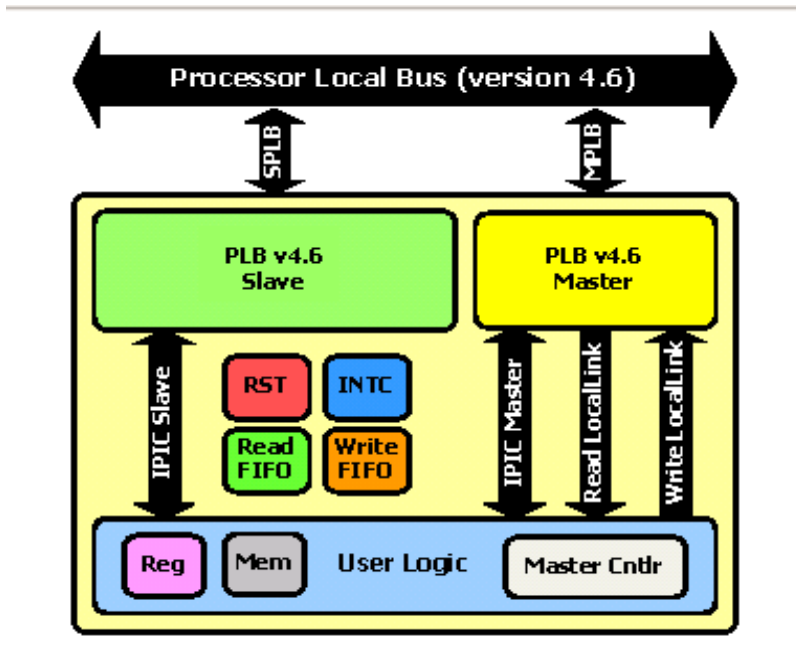
## Part 2: Custom peripheral design





## Custom peripheral design

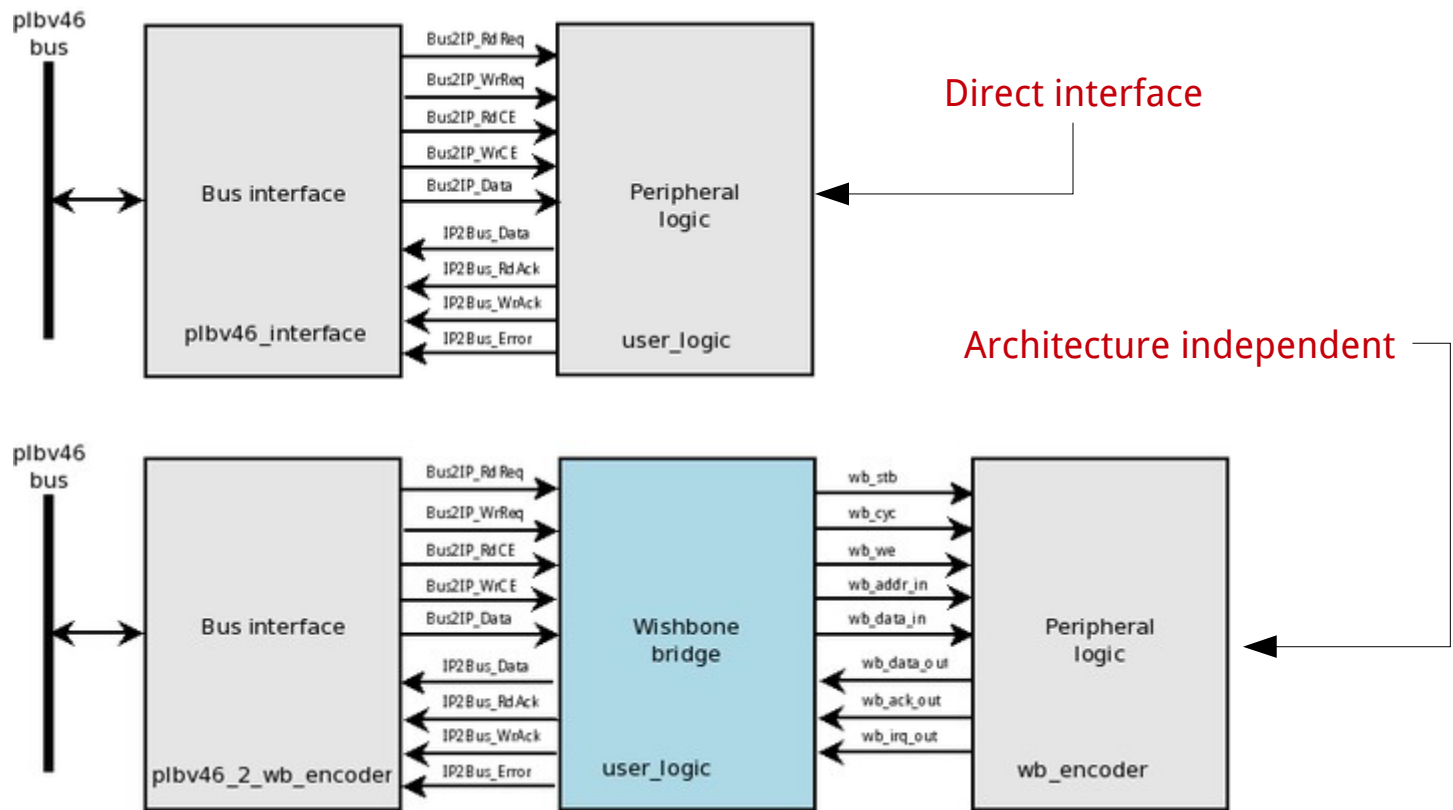
- Peripheral wizard creates two magic files created one tells about the order on how to synthesis the custom hardware module, the other one tell of the module configuration parameters and external ports.





# Custom peripheral design

## Peripheral interconnection

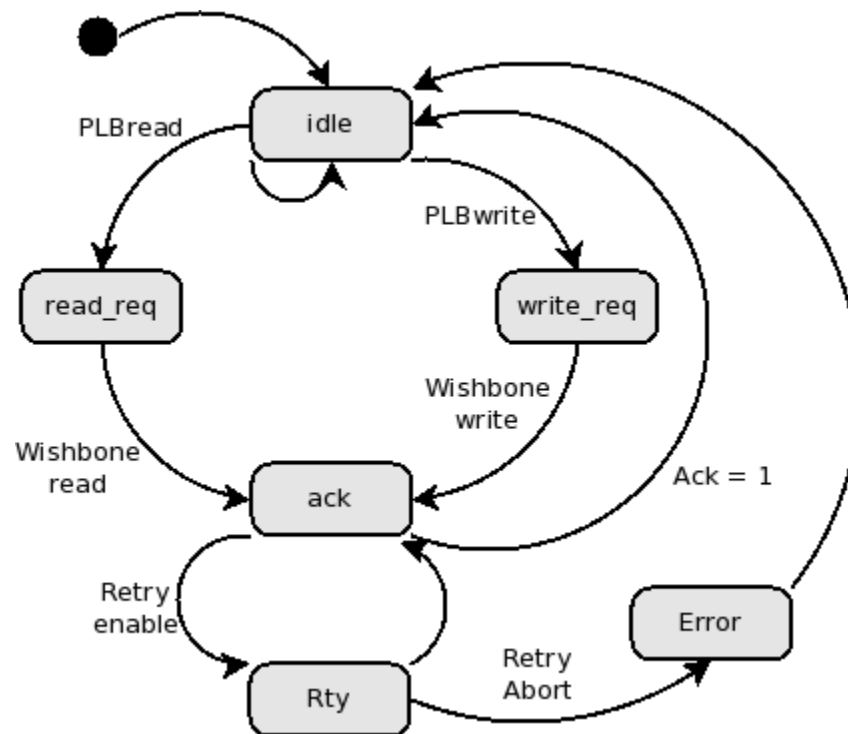




# Portable peripherals, examples

## Wishbone peripherals design

- The **wishbone bridge** is a **state machine** that must respond to **read/write request** from the system bus (PLB), and forward the request to custom peripheral.



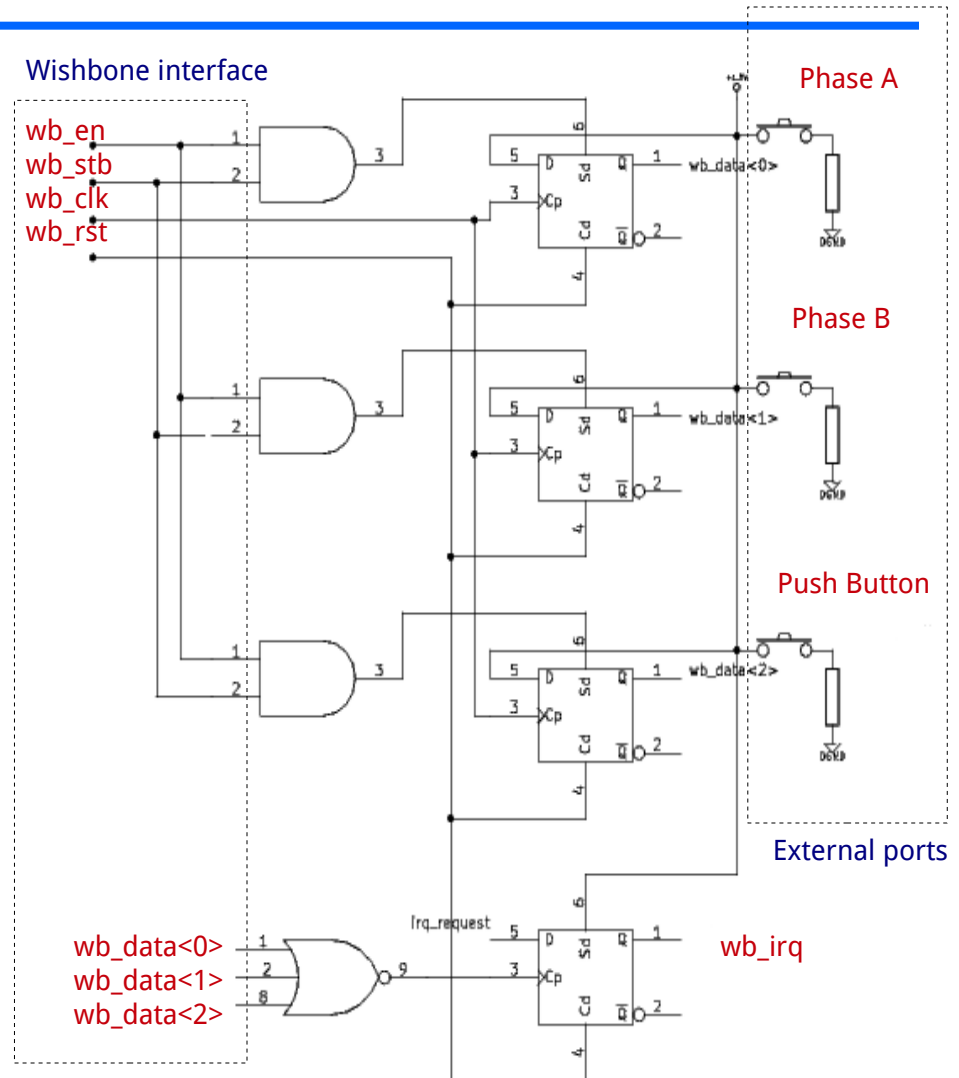


# Portable peripherals, examples

- **Wishbone wheel encoder design**
  - Emulates an input device much as how a mice scroll works.
  - Reports wheel event movements in Y-Axis
  - Emulates Left click when jog is pressed



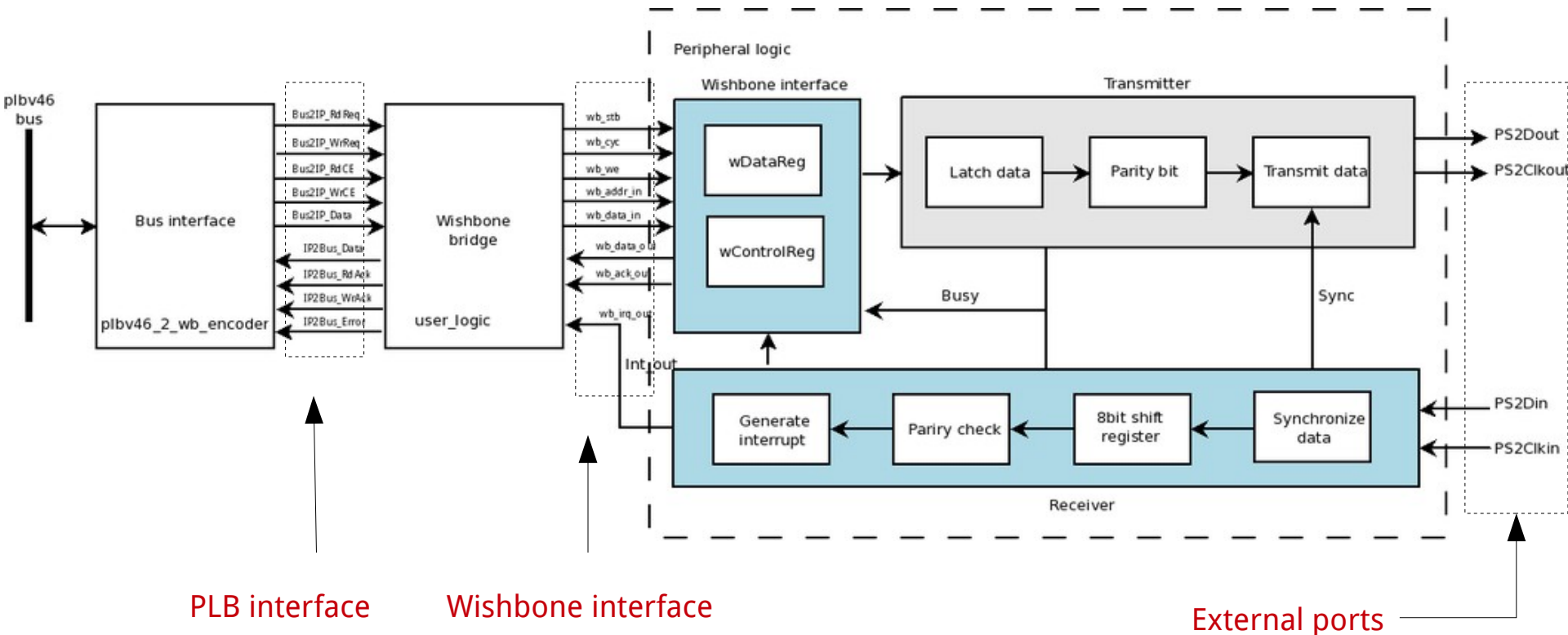
Scroll wheel





# Portable peripherals, examples

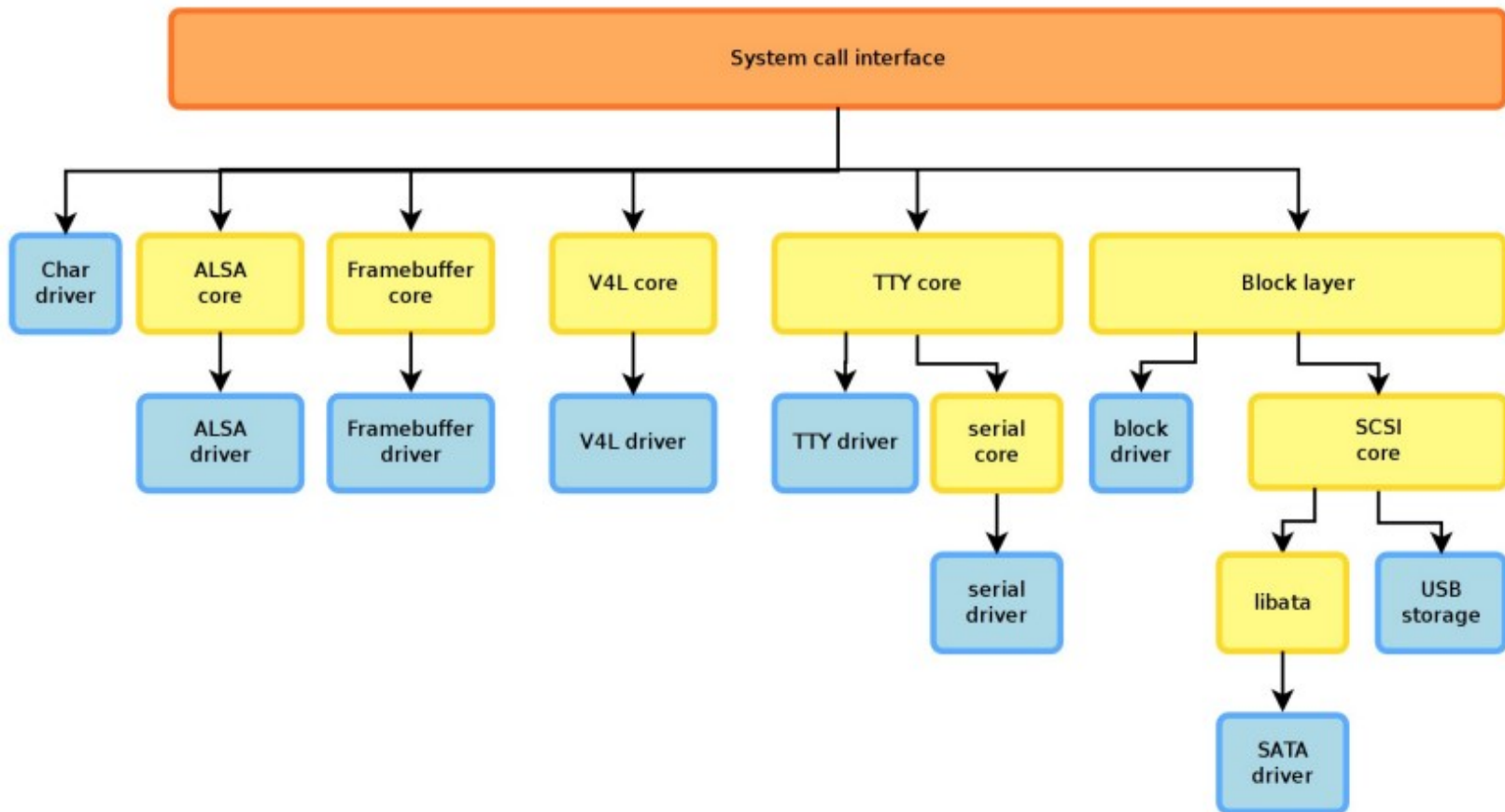
- Wishbone PS/2 keyboard controller peripheral design





# Part 3: Linux device drivers

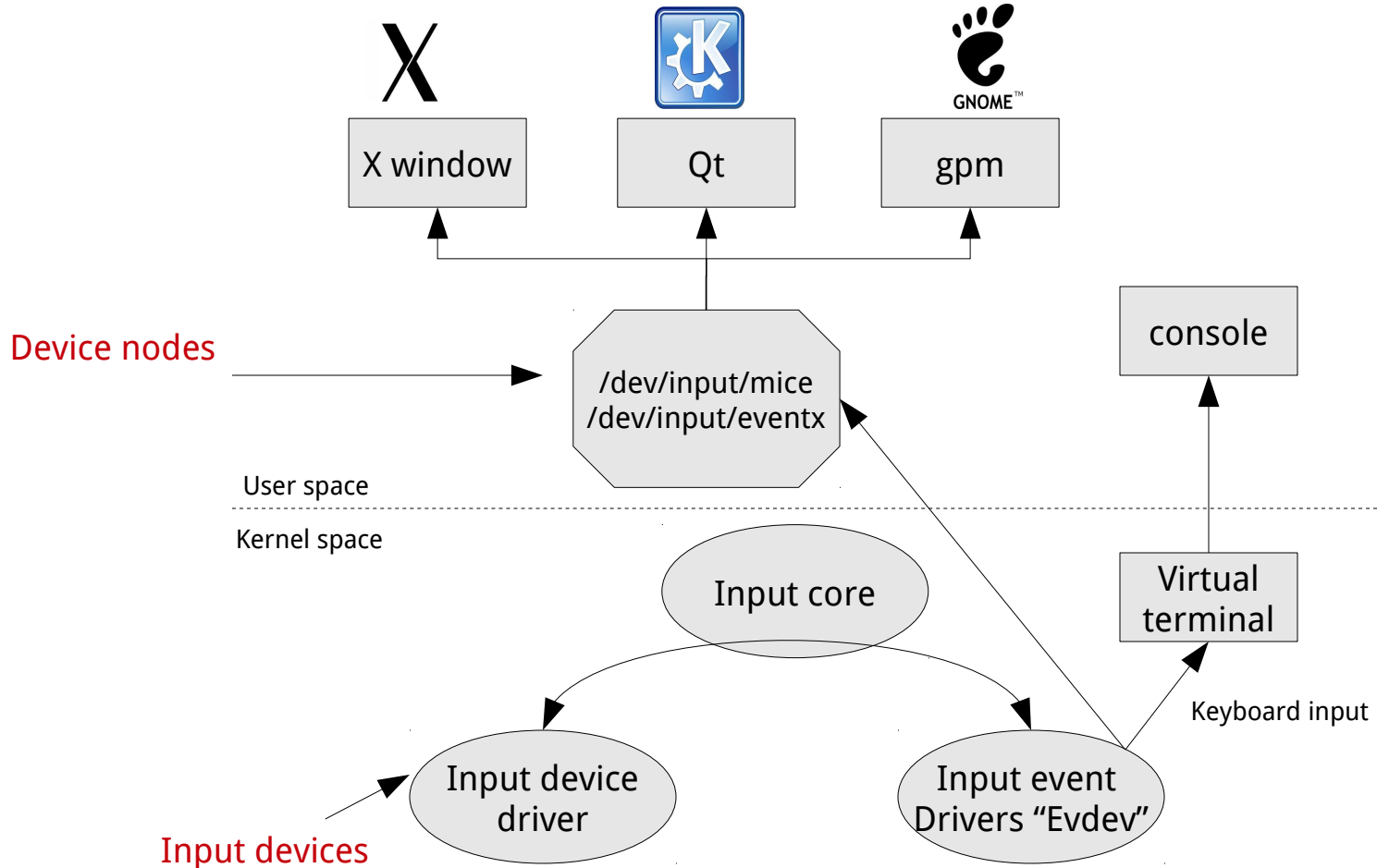
- Kernel device drivers subsystems





# Linux device drivers

## Input subsystem





# Linux device drivers

- Wishbone wheel encoder as input device
  - Emulates a scroll wheel found on standard mice

```
struct wb_encoder_drvdata {
    struct input_dev input; /* Input device subsystem */
    int irq;
    void __iomem base_addr; /* device addr space */
    bool armed;
    unsigned char dir;      /* 0 - clockwise, 1 - CCW */
};
```



# Linux device drivers

- The core functionality is part of the interrupt handler

...

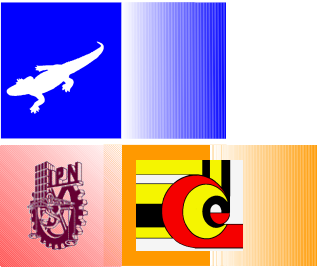
```
switch (state) {  
    case 0x0:  
        if (!encoder->armed) break;  
        input_report_rel(encoder->input, REL_Y,  
                        encoder->dir ? -1 : 1);  
        encoder->armed = false;  
        break;
```

....

```
}
```

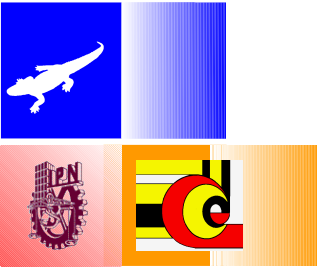
```
if(encoder->key_pressed)      /* Report key pressed */  
    input_report_key(encoder->input, BTN_LEFT, 1);
```

...



# Results

- Part 1: Custom embedded distro for target
  - **Yocto project maintainer for Xilinx target support**  
<http://git.yoctoproject.org/cgit/cgit.cgi/meta-xilinx/>
- Part2: Custom peripheral design
  - **Base hardware reference design**  
Technical report TR-XPS-TFT-Alligator\_OS.pdf
  - **Hardware base project**  
<http://git.gitorious.org/xilinx/xilinx-ml507.git> ←
  - **Wishbone encoder integration**  
Check wb-encoder development branch on



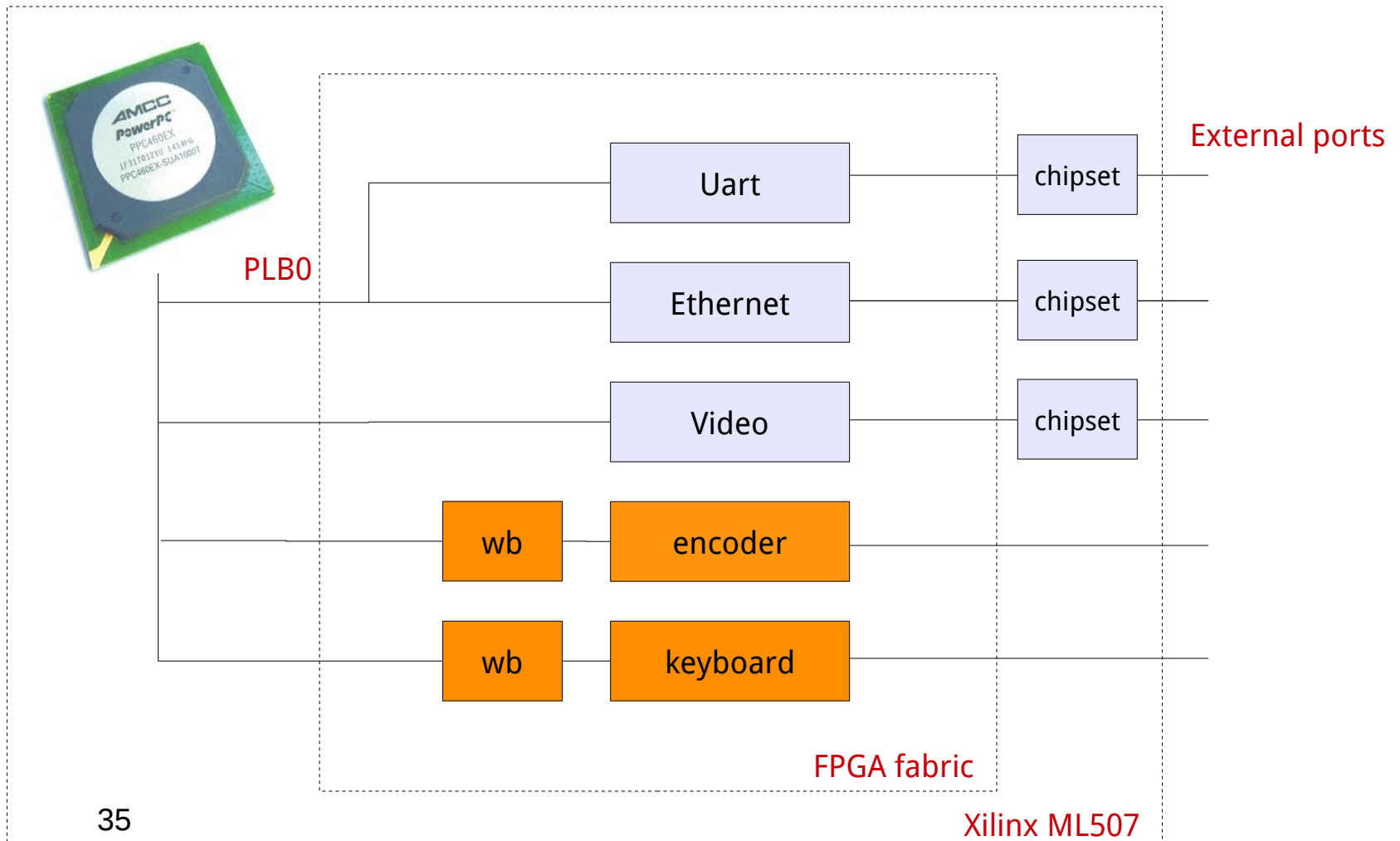
## Results

---

- Part 3: Linux wb\_encoder drivers  
<http://gitorious.org/meta-xilinx/meta-xilinx>  
See wb-encoder branch
- Integrating all  
Kernel error on initializing device  
Stand alone drivers always return `0` on data read  
  
It might be a **hardware bug** talk to the hardware engineer near by  
  
Weaker point **HDL Verification** to be improved by future works.



# Results





## Conclusions

- Automate Linux image generation using OE/Poky build frameworks for Xilinx embedded target platforms.

Openembedded and Poky build frameworks now supports a generic mechanism for Linux images generation for Xilinx target platforms, this generic support have been tested in Xilinx ML507 (PowerPC 440) and Xilinx ML405 (PowerPC 405);



# Conclusions

- Xilinx ML507 hardware reference project

A hardware reference project is available online which integrates most of the peripherals found in typically in any embedded systems, hardware module's configuration options can be inspected to reproduce the same base platform.

- PLB to Wishbone bridge implementation

An initial effort of implementing a Wishbone bridge is presented as an interconnection standard for hardware devices, the interconnection scheme followed is a point to point interconnection only supporting slave devices.



# Conclusions

- Wishbone hardware modules examples

Two hardware peripherals were designed to present as an example on how to implement custom peripherals and how to interconnect them to the processor local bus. The first one is a wheel encoder module that works as an input device much as how a mouse scroll works. The second one is a PS/2 keyboard controller, the implementation is incomplete but it presents how the design is partitioned.

- Linux device driver for Wishbone encoder

Example driver that forwards the events generated by the wheel encoder, it uses the kernel input subsystem so the Xserver and user space programs can work.



# Conclusions

---

- Need to generate knowledge base for HDL verification
  - To many tools and approach to do it
    - System C/Verilog: Test bench
    - Chip scope: Virtual Logic analyzer
    - Xilinx BFM: Bus functional models



## Future work

- HDL verification methodology
  - A lot of info, procedures, that could cover a semester course
- Improve hardware/software co-design
  - More automatization OE/Poky able to generate bitstream
- Wishbone shared bus support
  - A bus arbitration mechanism that can later on be integrated in Alligator\_SP



---

Thanks for your attention



---

## Questions



---

## References

It's alive! Linux on PowerPC porting guide, Grant Likely, SecretLab

Linux Kernel architecture for device drivers, Thomas Petazzoni, Free  
Electrons

On FPGAs with embedded processor cores for application in  
robotics, Roderick Colenbrander, U. Twente