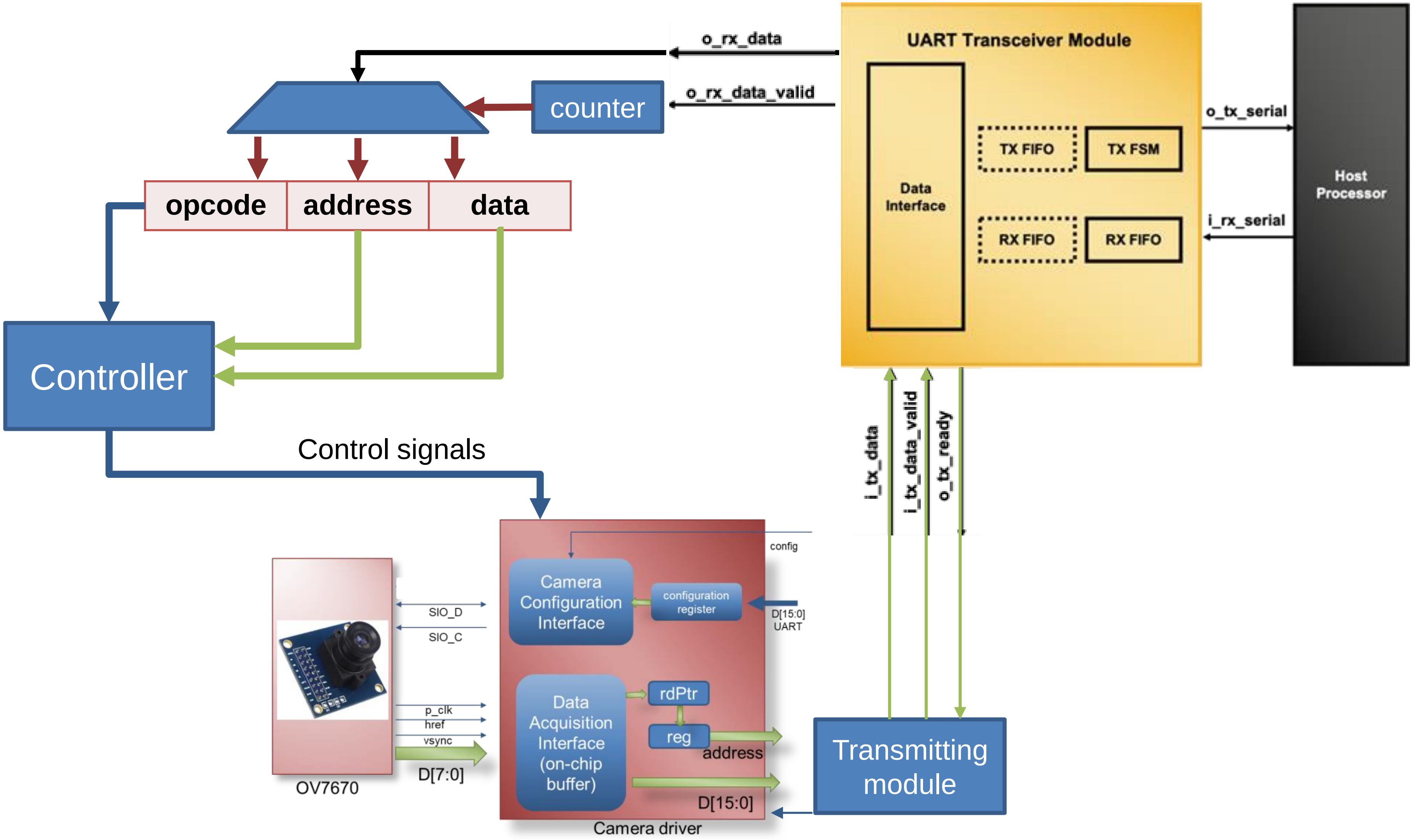
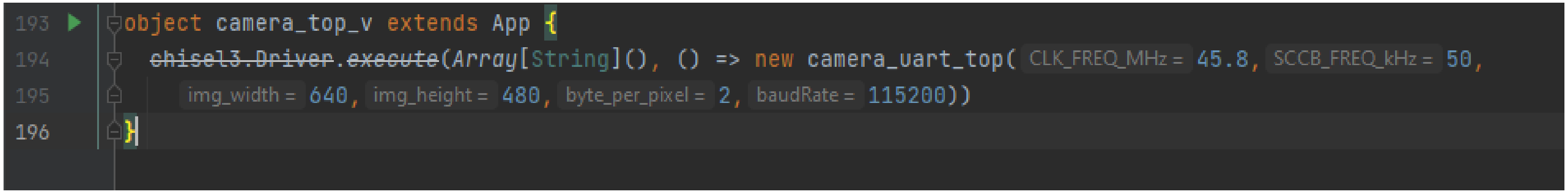
Communication architecture



How to configure the HDL module

- The camera_uart_top has 3 child files
  - Camera_module(img_width: Int, img_height:Int, byte_per_pixel: Int)
  - SCCB_interface(CLK_FREQ_MHz: Double, SCCB_FREQ: kHz)
    - Where CLK_FREQ_MHz is the working frequency of capture module. This should be high, eg: CLK_FREQ_MHz=45.7. You need to specify this value before generating the corresponding Verilog file for synthesis.
    - SCCB_FREQ: SIOC frequency. Typically this values is smaller than 400KHz, but I think you should specify it to be 50 kHz to play around :P
  - UART(fre: Int, baudRate: Int)
    - Fre=CLK_FREQ_MHz



```
object camera_top_v extends App {
  ~~chisel3.Driver~~.~~execute~~(Array[String](), () => new camera_uart_top( CLK_FREQ_MHz = 45.8, SCCB_FREQ_kHz = 50,
    img_width = 640, img_height = 480, byte_per_pixel = 2, baudRate = 115200))
}
```

Instruction register

opcode(18,16)	address	data

24-bit IR

- Data sent to instruction register are in package of 3 bytes
- Opcode(18,16), bit(19,23) are reserved
  - op(18,16): 000 (check sccb's status)
    - address, data: Don't care
  - op(18,16): 001 (generate a configure signal for the sccb)
    - address, data: must be specified (these two will be used to configure working modes for the camera)
  - op(18,16): 010 (capture image)
    - Address, data: don't care
  - op(18,16): 011 (check camera's status)
    - Binary returned value indicate working status of the camera
    - Address, data: don't care
  - op(18,16): 100 (check buffer status)
  - op(18,16): 101 (read buffer and transmit back to CPU)
    - Address, data: don't care

Tutorial ☺ ?

opcode(18,16)	address	data

- To configure a mode for the sccb interface, you need to transmit data from CPU to UART in the following manner
  - Opcode: 0x00
  - Address: 0xAA (some value)
  - Data: 0xAA (some value)
- For other modes, you only need to specify the correct opcode, address and data could be any values. They must be transmitted in a package of 3 bytes
- The UART configurations:
  - Two stop bits, one start bit