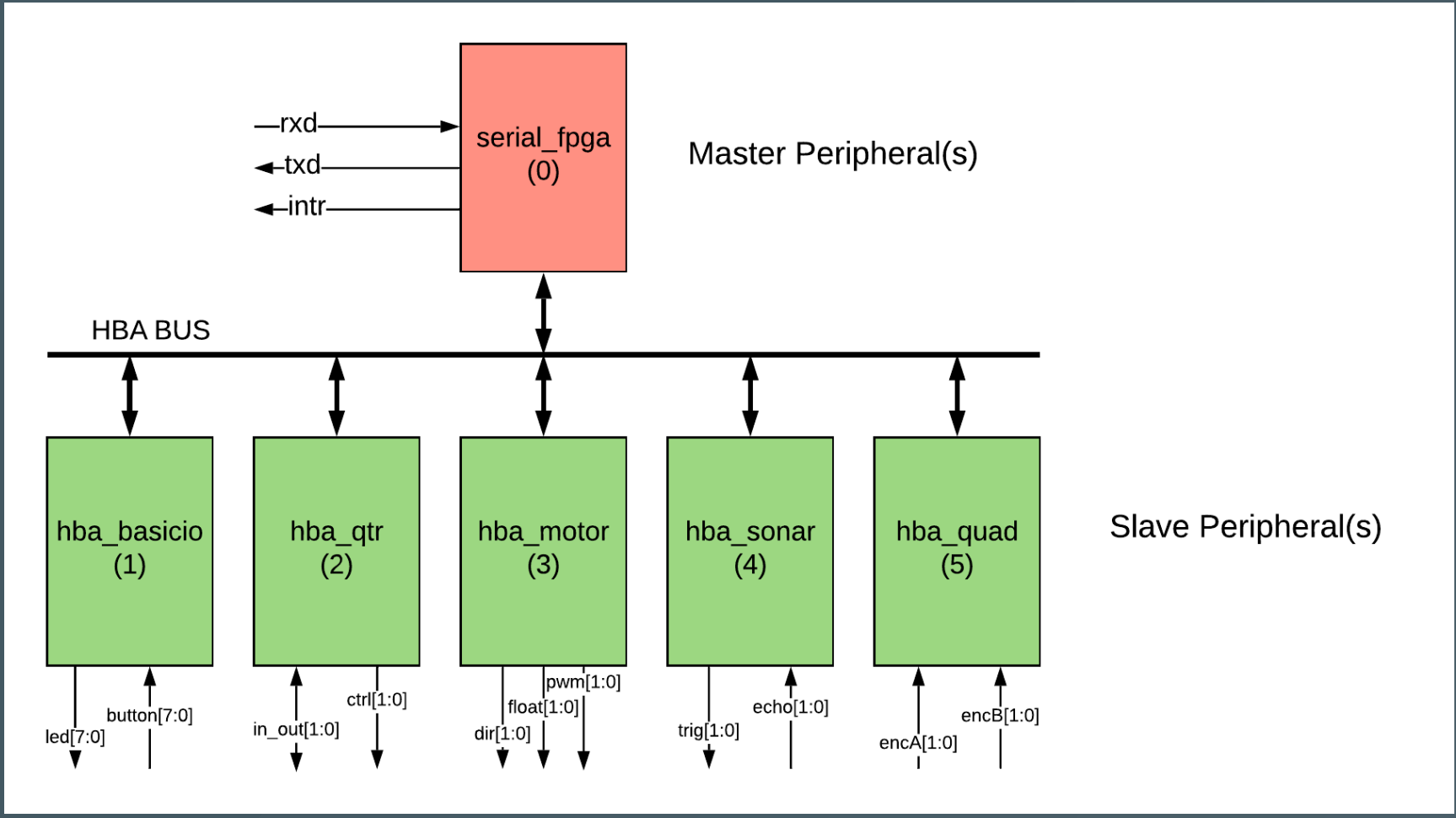
# Today

- FPGA HBA Architecture
- Serial Pi to FPGA Interface
- Serial Demo Using raw read/write
- HBA Peripheral Interface
- HBA BasicIO Peripheral
  - Verilog code walk through
  - HBA plugin code walk through

# FPGA HBA Architecture



# FPGA HBA Arch cont.

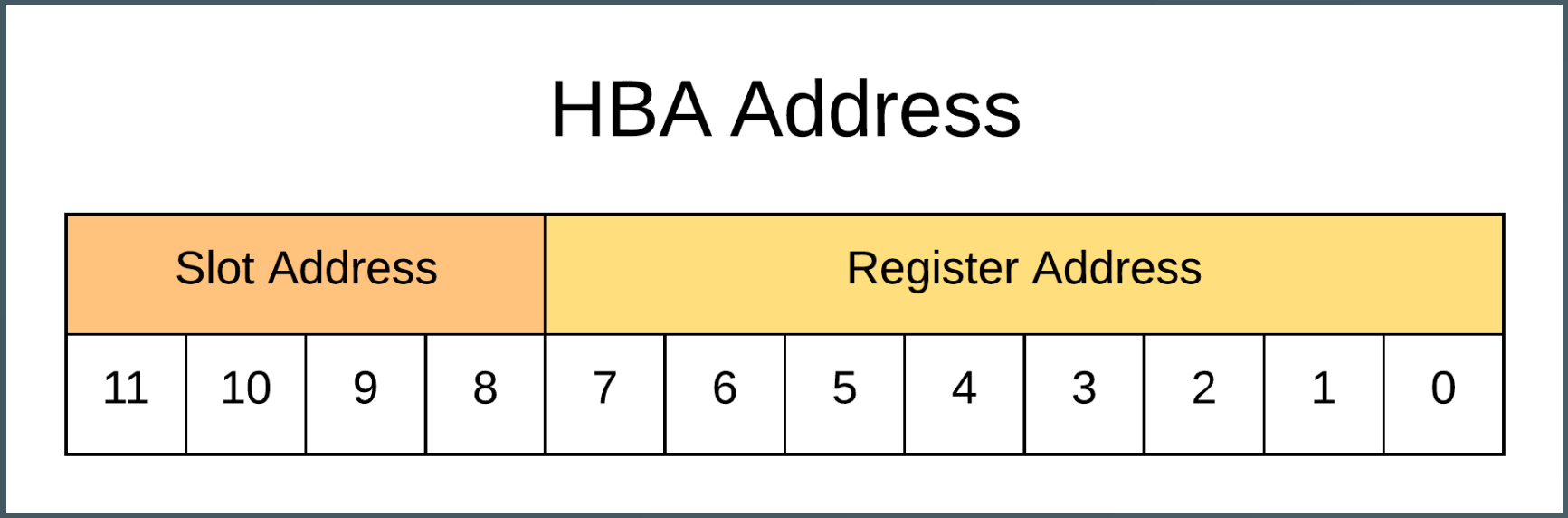
- Serial (UART) interface between Pi and FPGA
- HBA Bus connects Peripherals
- Each Peripheral is assigned a slot
- Peripheral can be Master, Slave or Both
- Slave Peripherals
  - Contain bank of registers (Read, Write or Both)
- Master Peripherals
  - Can read/write to Slave Peripheral registers
  - HBA Bus supports multiple master

# Serial Interface

The Raspberry Pi uses the serial interface to read and write to the
HBA Peripheral registers. So the Raspberry Pi is a Master on the HBA Bus.

- Serial Interface (from perspective of rasp-pi)
  - **rpi_txd** : Transmit data to the FPGA.
  - **rpi_rxd** : Receive data from the FPGA
  - **rpi_intr** : Interrupt from FPGA. Indicates FPGA has data to be read.

# HBA Addressing

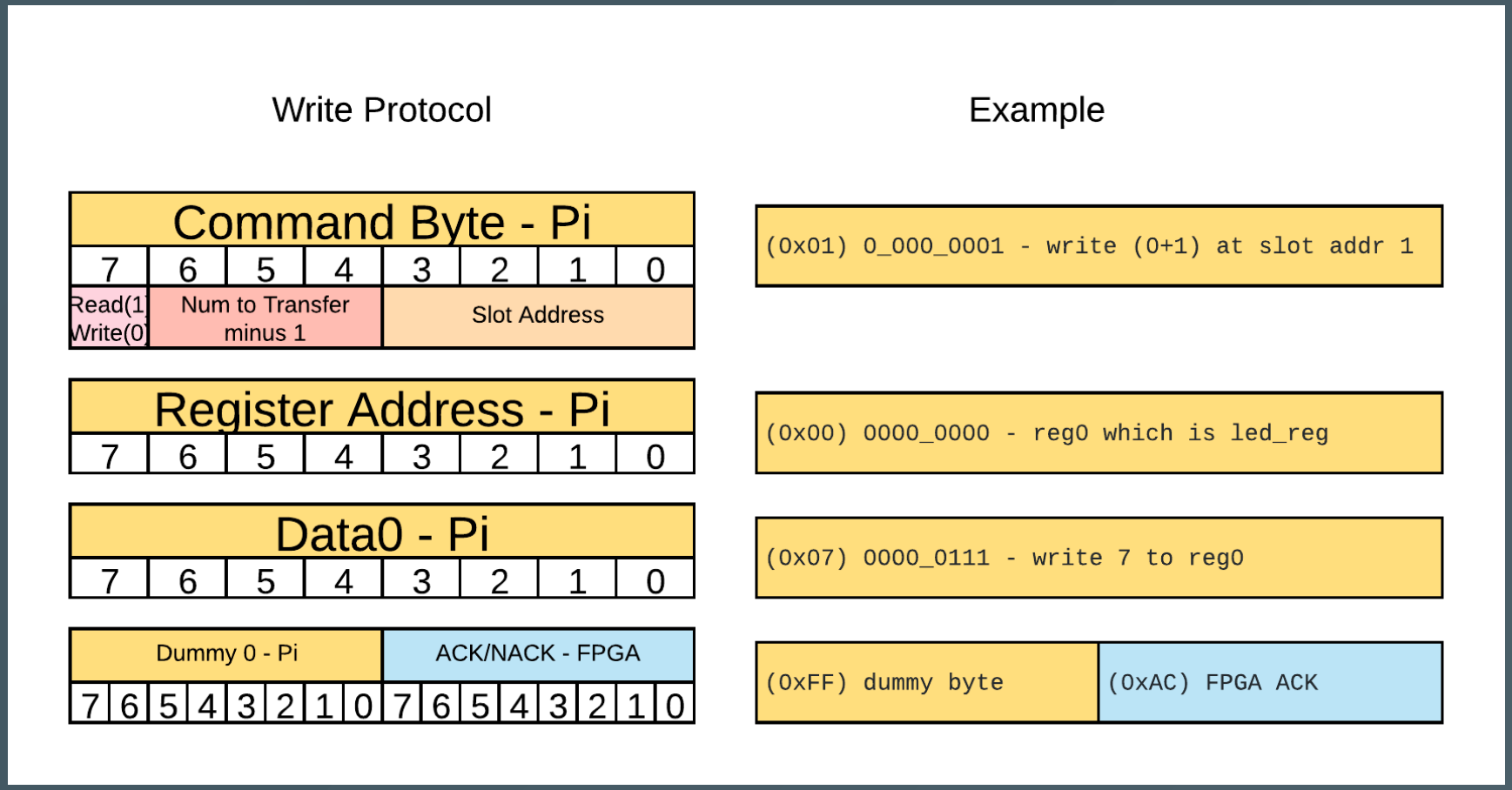



- The upper 4 bits select the desired peripheral slot. There are 16 possible slots.
- The lower 8 bits select the desired peripheral register. There are 256 possible registers.

# Serial Protocol

- To receive a byte from the FPGA the Pi must send a dummy byte.
- The serial interface is full duplex.
- Serial Interface Documentation
  More information about the serial interface.

# Write Protocol



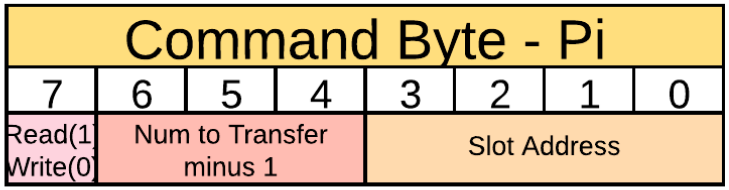
# Write Protcol cont.

- Command Byte:
  - 7 - Write(0) operation
  - 6:4 - Number of Registers to write minus 1. So (1-8) possible.
  - 3:0 - Peripheral Slot Address
- Starting Peripheral Register Address. Auto increments if multiple data.
- Data0 .. DataN : The data to write.
- ACK/NACK : Pi sends dummy byte. FPGA sends ACK, indicates successful write of data.

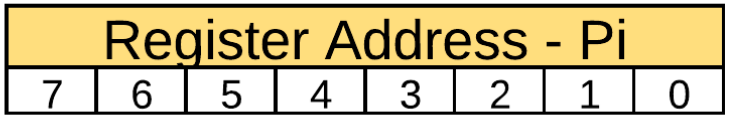
# Read Protocol

Read Protocol

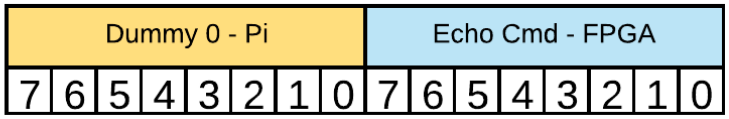
Example



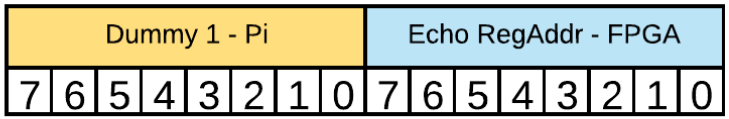

```
(0x81) 1_000_0001 - read (0+1) at slot addr 1
```



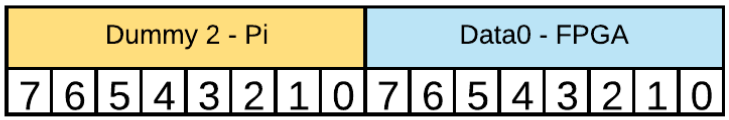

```
(0x01) 0000_0001 - reg1 which is led_button_in
```




```
(0xFF) dummy byte | (0x81) Echo'd Cmd
```




```
(0xFF) dummy byte | (0x01) Echo'd Reg Addr
```




```
(0xFF) dummy byte | (0x03) Buttons value
```

# Read Protocol cont.

- Command Byte:
  - 7 - Read(1) operation
  - 6:4 - Number of Registers to read minus 1. So (1-8) possible.
  - 3:0 - Peripheral Slot Address
- Starting Peripheral Register Address. Auto increments if multiple data.
- Echo Cmd : The Pi sends a dummy byte. The FPGA echos the cmd byte.

# Read Protocol cont(2)

- Echo RegAddr : The Pi sends a dummy byte. The FPGA echos the RegAddr byte.
- Data0 .. DataN : The Pi sends dummy byte for each reg read. The FPGA sends reg value.

# Serial_fpga read raw bytes

- Usually use HBA resources for accessing peripherals
- However it is possible to echo all the byte received from the FPGA
- Useful for development and debug
- In first robot terminal type:

```
hbacat serial_fpga rawin
```

# Serial_fpga write raw bytes

- It is also possible to send raw bytes to the FPGA
- Example 1 write 7 to leds
  - Slot 1 peripheral, Reg0 to the value 07
  - This should be the hba_basicio led register
- In second robot terminal type:

```
hbaset serial_fpga rawout 01 00 07 ff
```

# Serial_fpga write raw bytes cont.

- Example 2 read button value
  - Slot 1 peripheral, Reg1 (read)
  - This is the hba_basicio button_in register
- In second robot terminal type:

```
hbaset serial_fpga rawout 81 01 ff ff ff
```

# HBA Bus Slave Interface (1)

- **hba_clk (input)** : This is the bus clock. The HBA Bus signals are valid on the rising edge of this clock.
- **hba_reset (input)** : This signal indicates the peripheral should be reset.
- **hba_rnw (input)** : 1=Read from register. 0=Write to register.
- **hba_select (input)** : Indicates a transfer in progress.

# HBA Bus Slave Interface (2)

- **hba_abus[11:0] (input)** : The address bus.
  - **bits[11:8]** : These 4 bits are the peripheral address. Max number of peripherals 16.
  - **bits[7:0]** : These 8 bits are the register address. Max 256 reg per peripheral.
- **hba_dbus[7:0] (input)** : Data sent to the slave peripherals.

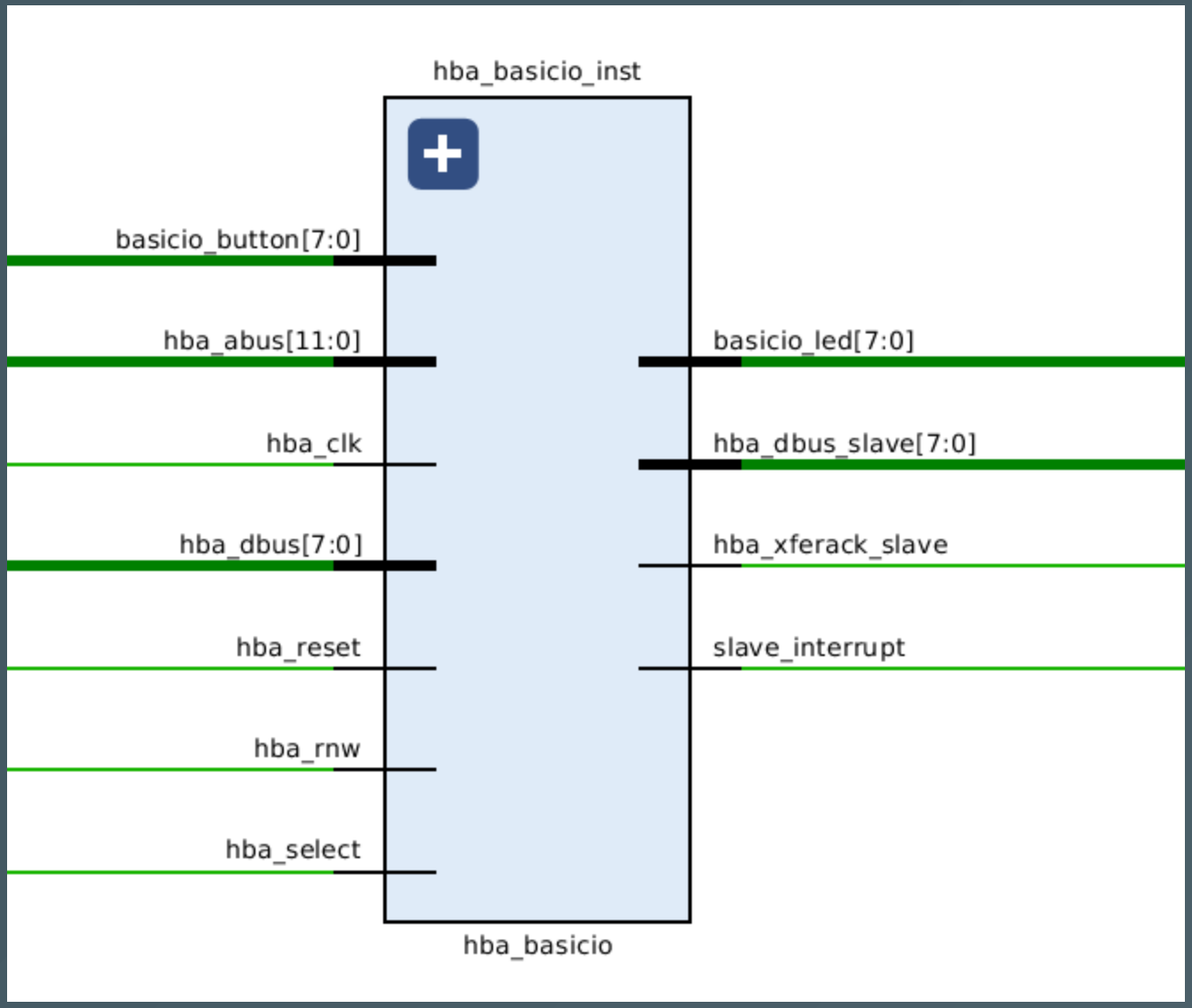
# HBA Bus Slave Interface (3)

- **hba_xferack_slave (output)** : Acknowledge transfer requested. Asserted when request has been
completed. Must be zero when inactive.
- **hba_dbus_slave[7:0] (output)** : Data from the slave. Must be zero when inactive.
- **hba_interrupt_slave (output)** : Each slave has a dedicated signal back to
a interrupt controller. If not used tie to zero.

# HBA Bus Slave Interface (4)

- HBA Bus Documentation
  More information about the HBA Bus.

# hba_basicio Peripheral



# hba_basicio Peripheral Links

- hba_basicio documentation
- hba_basicio driver readme
- hba_basicio verilog source

# hba_basicio HBA Pluggin in C

- In each peripherals directory there is a directory called **sw** that
  holds the HBA pluggin that is written in C.
- HBA Daemon Design Doc
- EEDD documentation
- hba_basicio HBA C pluggin