

ECE 448

Lecture 9

Introduction to the FPro System

Required Reading

P. Chu, FPGA Prototyping by VHDL Examples

- 8.1 Embedded SoC
- 8.3 FPro SoC platform
- 10.3 MMIO I/O core
- 10.4 Timer core development
- 10.5.3 Vanilla MMIO subsystem
- 10.7 Vanilla FPro system construction
- 9 Bare Metal System Software Development

First Lab Task

Companion Website of

FPGA Prototyping by VHDL Examples 2nd edition

https://academic.csuohio.edu/chu_p/rtl/fpga_mcs_vhdl.html

- **General info**

- Updated FPro System Development Tutorial

- A.5 SHORT TUTORIAL ON FPRO SYSTEM DEVELOPMENT

- A.4.3 Generate a MicroBlaze MCS

Source Code

Companion Website of

FPGA Prototyping by VHDL Examples 2nd edition

https://academic.csuohio.edu/chu_p/rtl/fpga_mcs_vhdl.html

- **Source codes**

- read_me file: readme_source_code.pdf

- source file: fpga_mcs_vhdl_src.zip

- (last updated 11/10/2017)

Embedded Systems vs. General-Purpose Computing

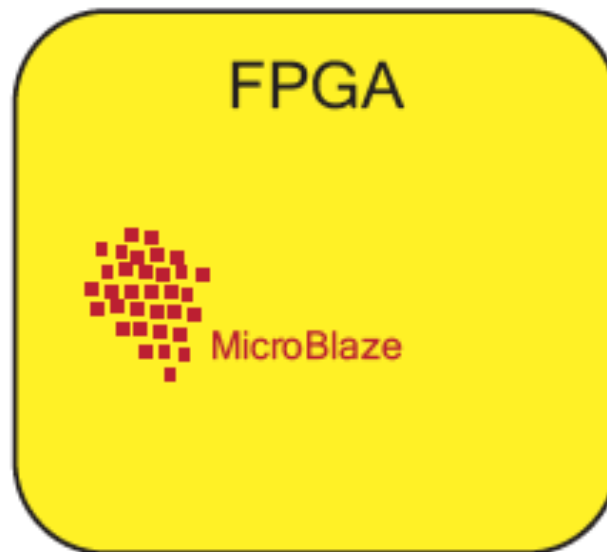
General-Purpose Computing

- Broad range of applications
- Programmable by end user
- Faster is better
- **Criteria:**
 - Average speed
 - Cost

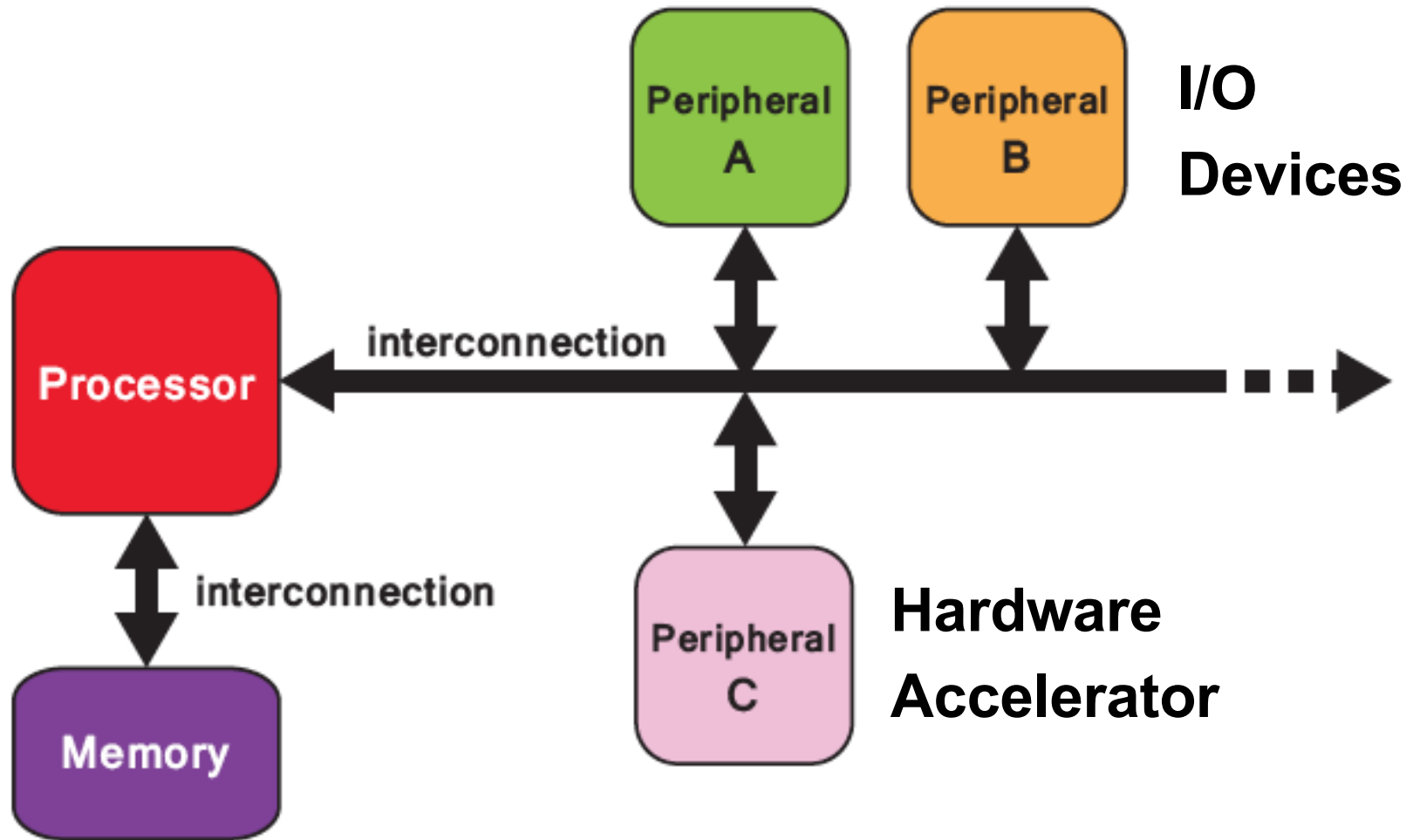
Embedded Systems

- Few applications known at design time
- Not programmable by end user
- Fixed run-time requirements (additional computing power not useful)
- **Criteria:**
 - Cost
 - Power
 - Energy

FPGA with Soft Processor Core = FPGA-Based System on Chip (SoC)



Simplified Hardware Architecture of an Embedded SoC



IP Cores

(IP = Intellectual Property)

- *Blocks of logic used in developing digital systems.*
- *Support repeated use of previously designed components.*
- *Ideally, portable to*
 - *devices, e.g., FPGAs, of various vendors*
 - *various boards, e.g., Basys 3, Nexys 4 DDR, Arty A7*
 - *various design methodologies,*
e.g., hardware only, software/hardware
- *Can be developed by the device manufacturers, third-party vendors, or the users themselves*

What is Software/Hardware Codesign?

Integrated design of systems that consist of hardware and software components

- Analysis of HW/SW boundaries and interfaces
- Evaluation of design alternatives

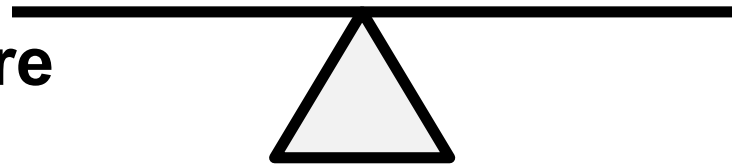
Software vs. Hardware Trade-offs

Improve Performance
Improve Energy Efficiency
Reduce Power Density

Manage Design Complexity
Reduce Design Cost
Stick to Design Schedule

**Implement
more in Hardware**

**Implement
more in Software**

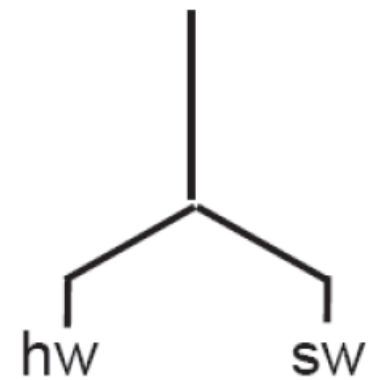
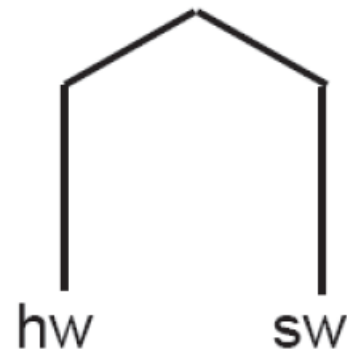


Why Co-design?

classic design



co-design



FPro System

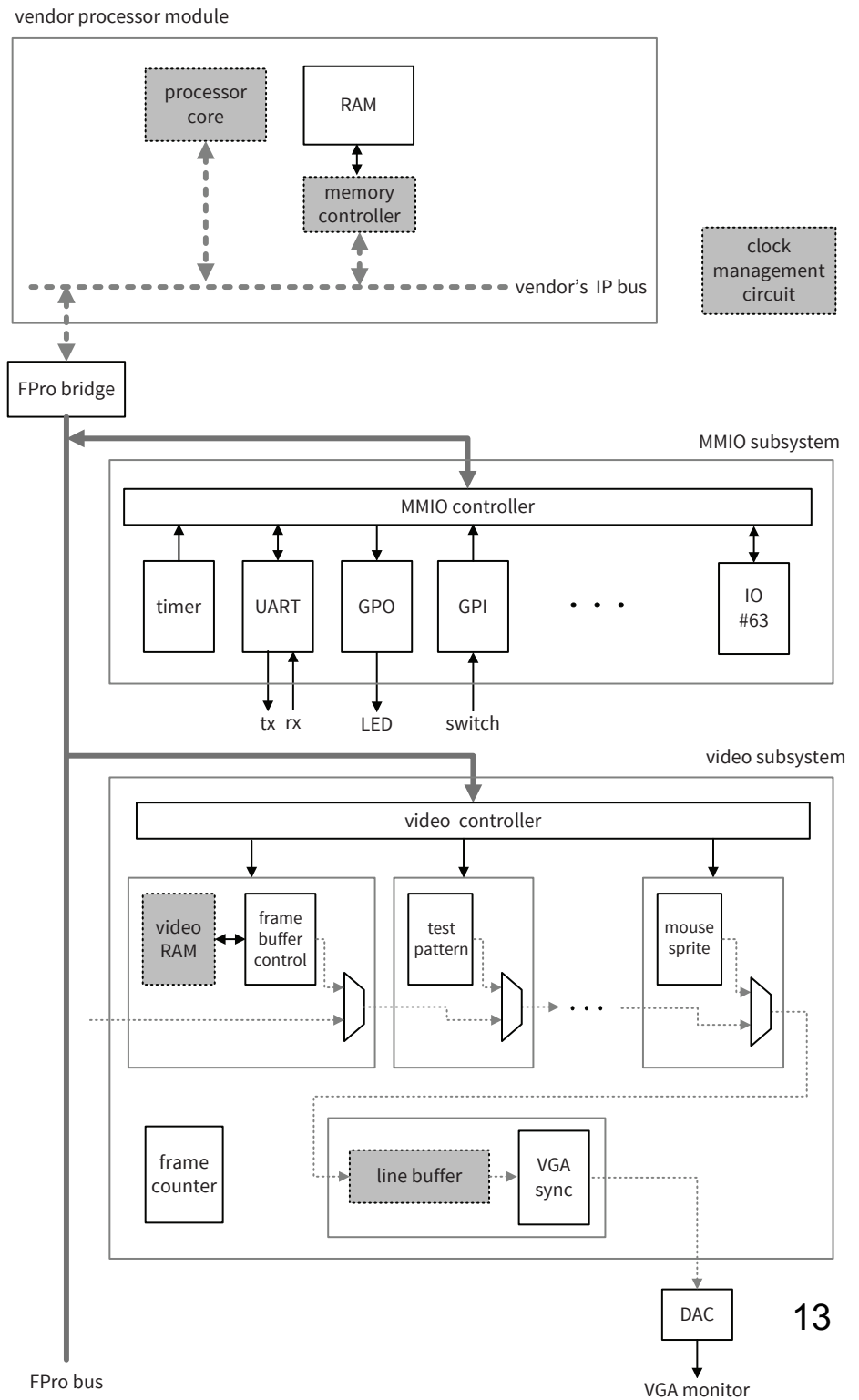
FPGA Prototyping” or “Fun and Professional

- System for software/hardware co-design using popular FPGA boards
- Developed by the author of the primary textbook, Dr. Pong P. Chu, Cleveland State University
- Used in ECE 448 since Spring 2018

Major Features:

- Simple
- Functional
- Portable
- “Upward compatible”
- Fun

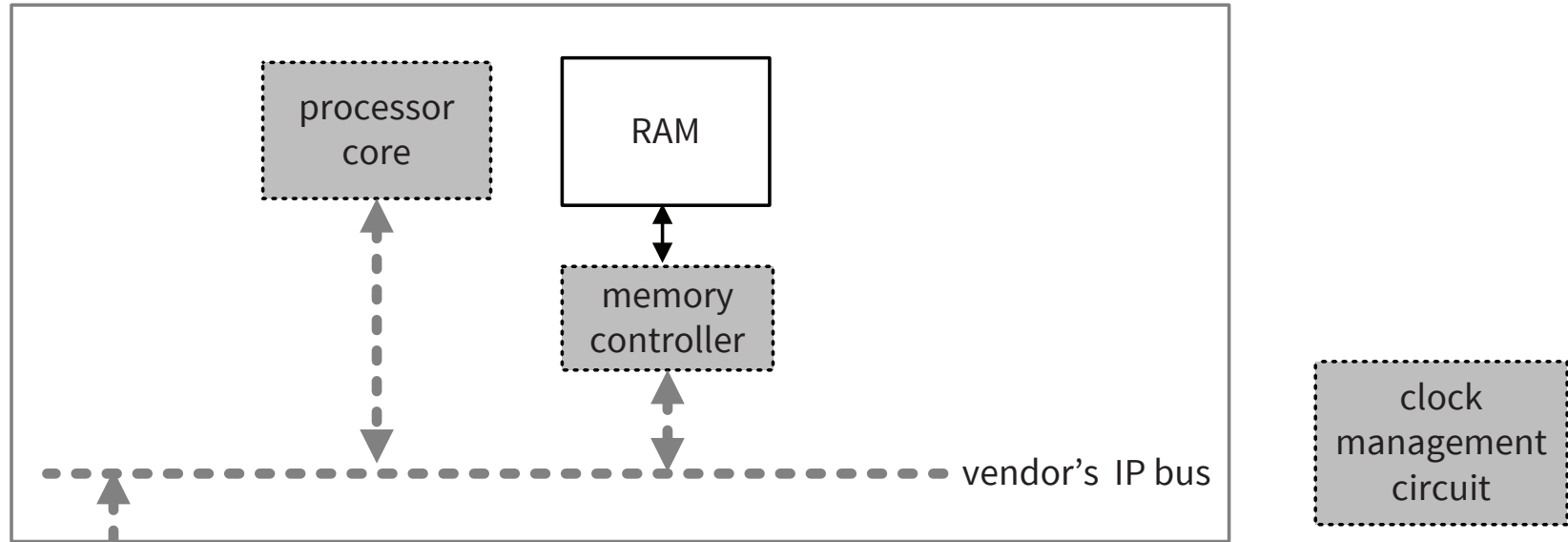
Top-level diagram of an FPro system



Platform Hardware Organization

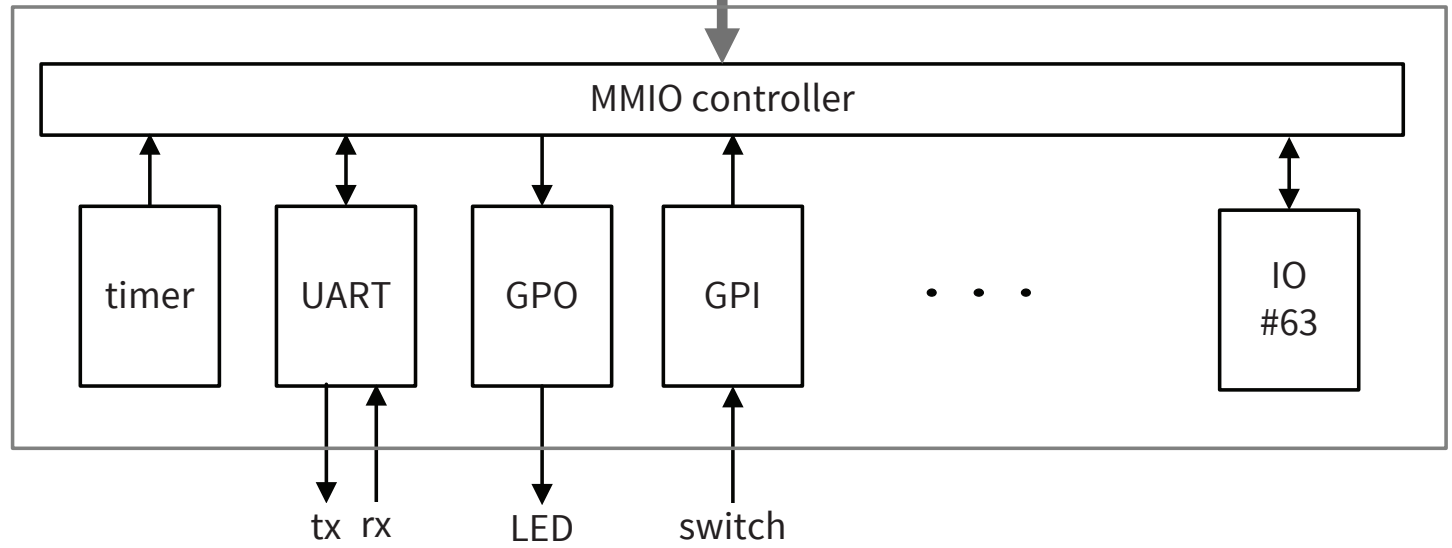
- **Processor module**
 - 32-bit-wide data path
 - 32-bit memory address space
 - Memory-Mapped-I/O scheme for I/O access
- **FPro bridge and FPro bus**
 - simple synchronous bus protocol for the two subsystems
- **MMIO (memory-mapped I/O) subsystem**
 - the memory and registers of the I/O peripherals are mapped to the same address space
 - accessing memory and I/O peripherals involves the same instructions
- **Video subsystem**
 - coordinates the operation of video cores

vendor processor module

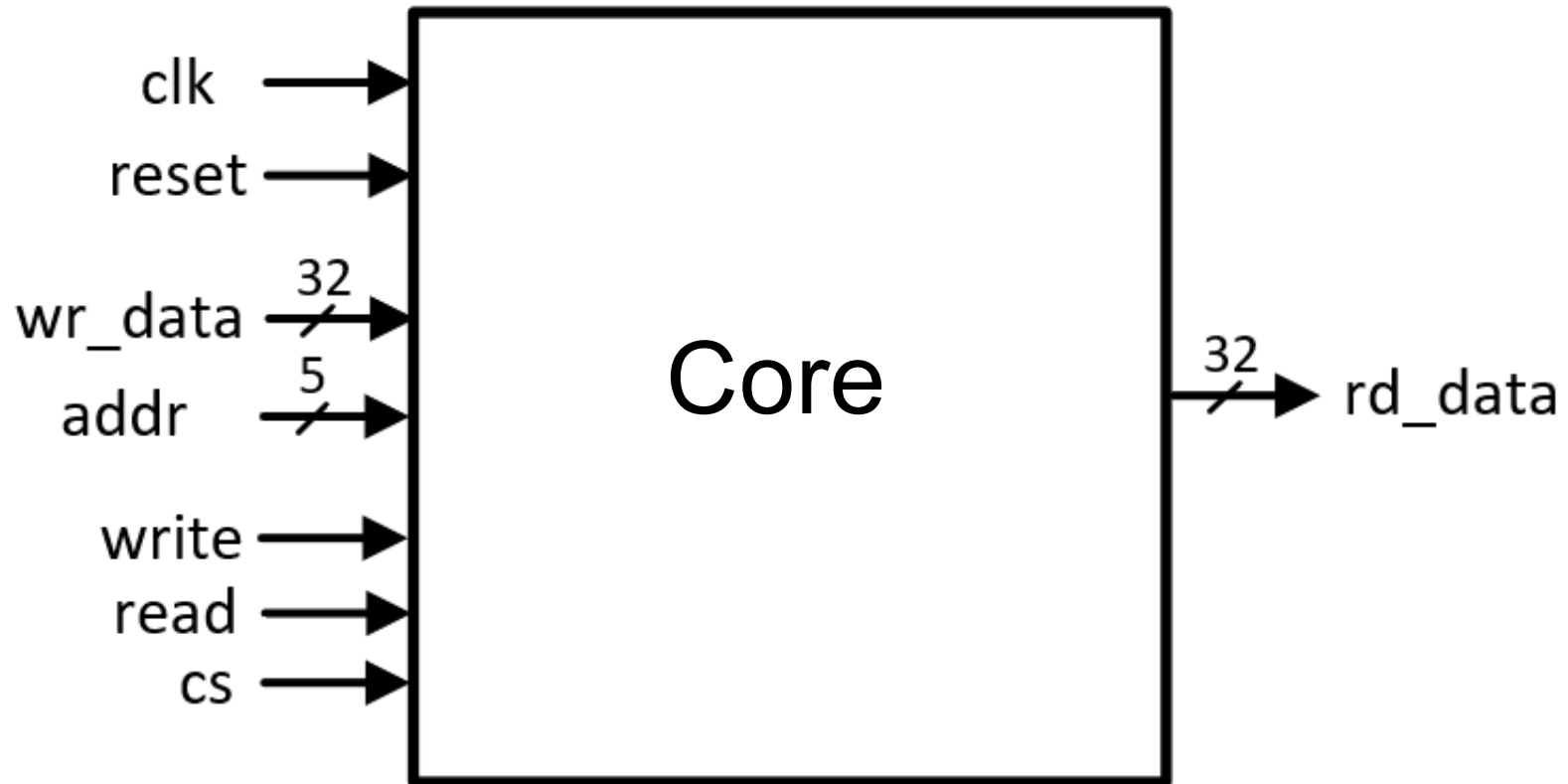


FPro bridge

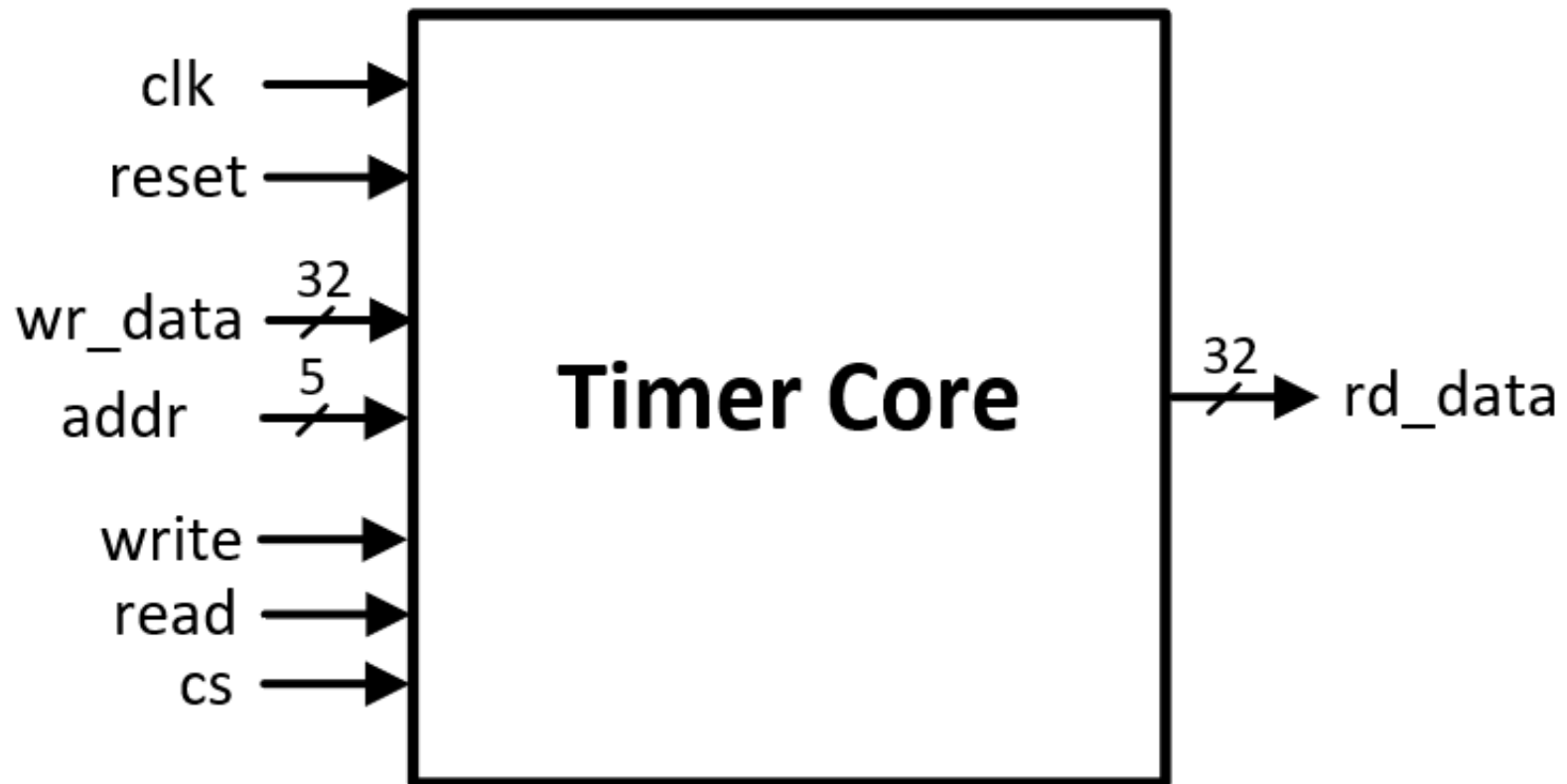
MMIO subsystem



Interface of every MMIO Core



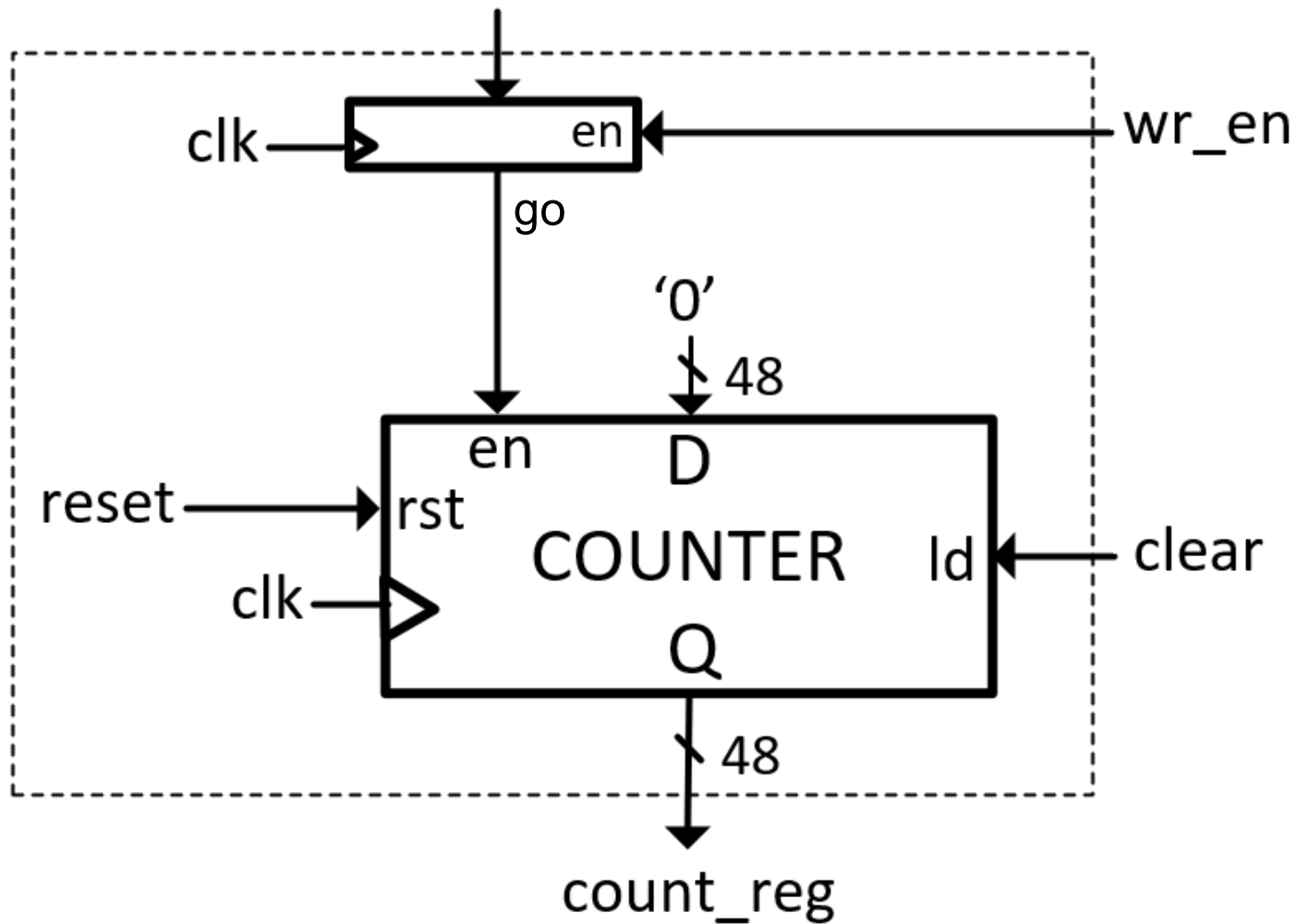
Interface of the Timer Core



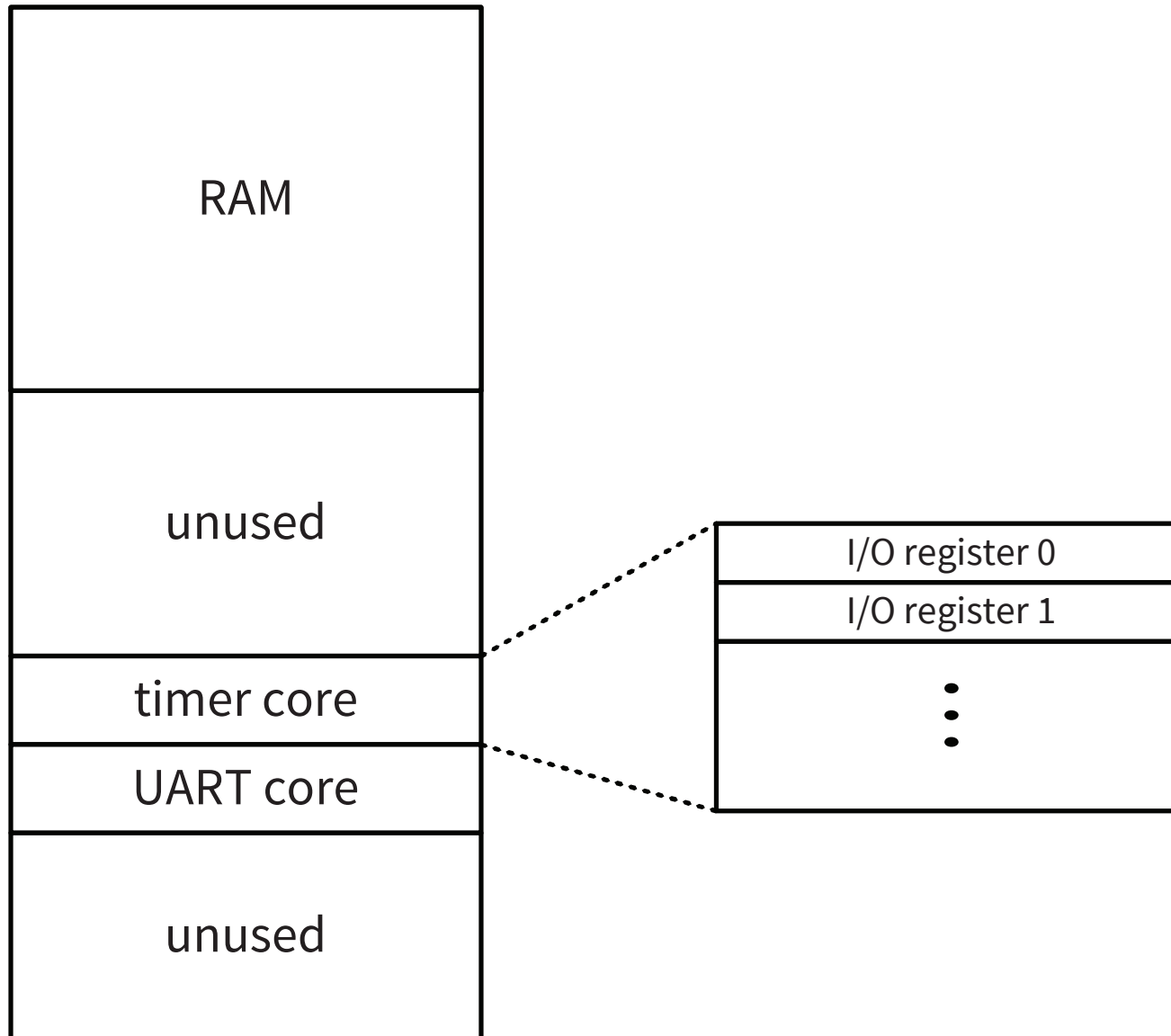
Timer Core

- 48-bit counter value
- The processor interacts with the counter as follows:
 - retrieve (i.e., read) the counter value.
 - set or reset (i.e., write) a “go” signal to resume or pause the counting.
 - generate (i.e., write) a “clear” pulse to clear the counter to 0.

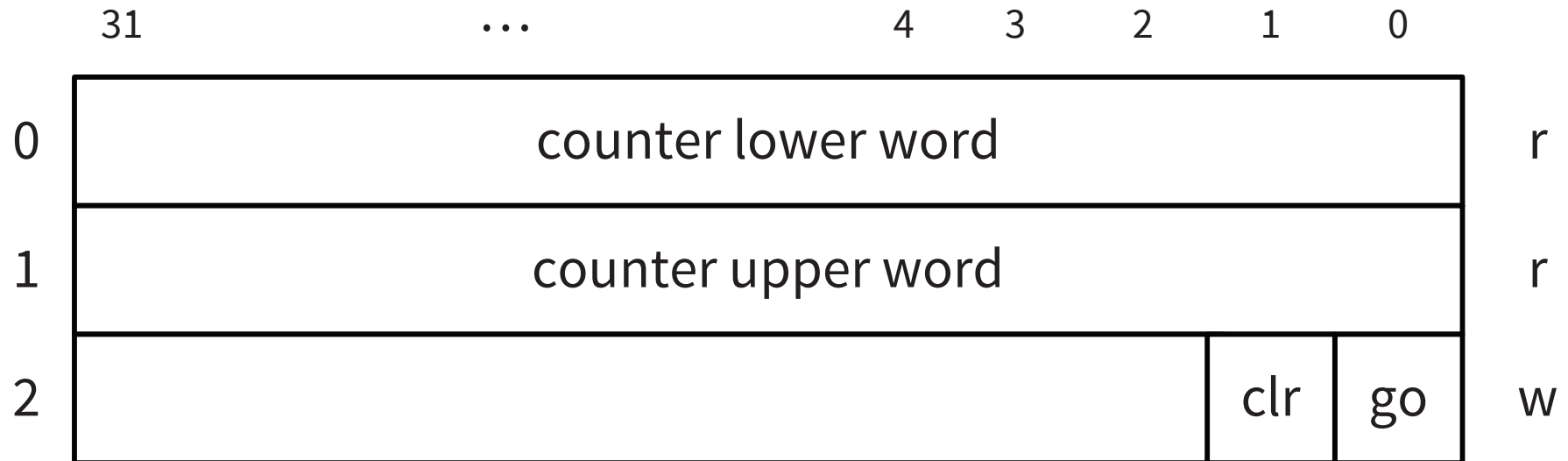
Main Functionality of the Timer Core



Address Map of the FPro System

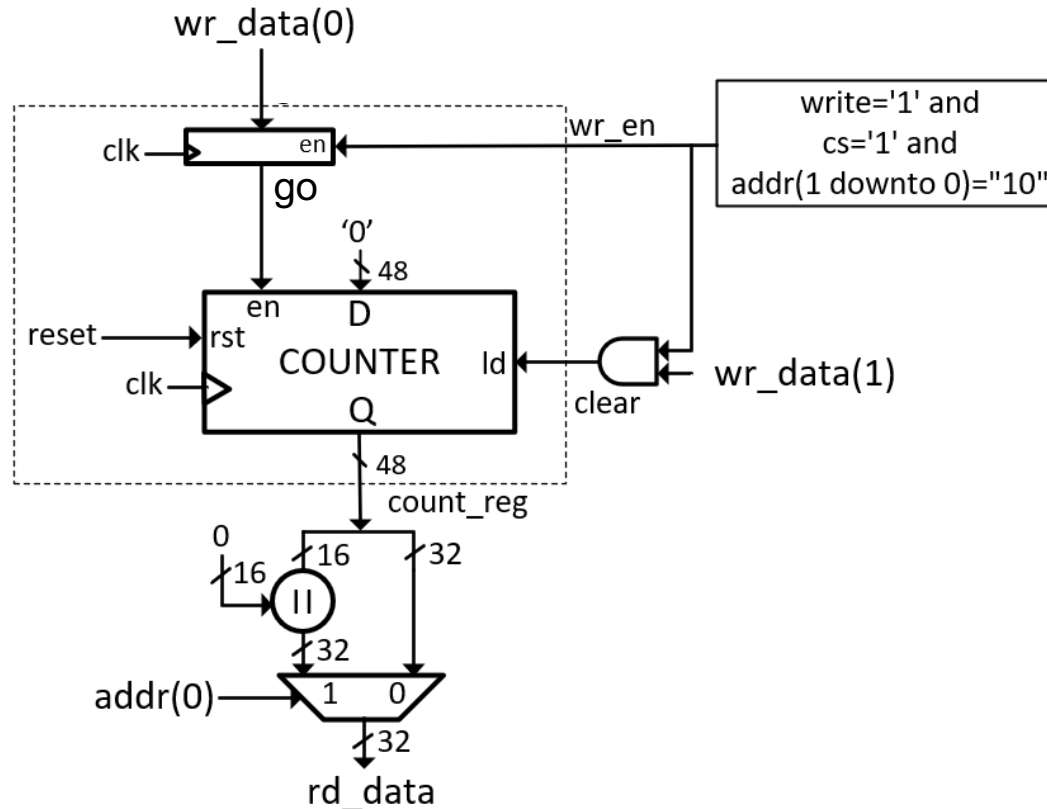


IO Register Map of Timer Core



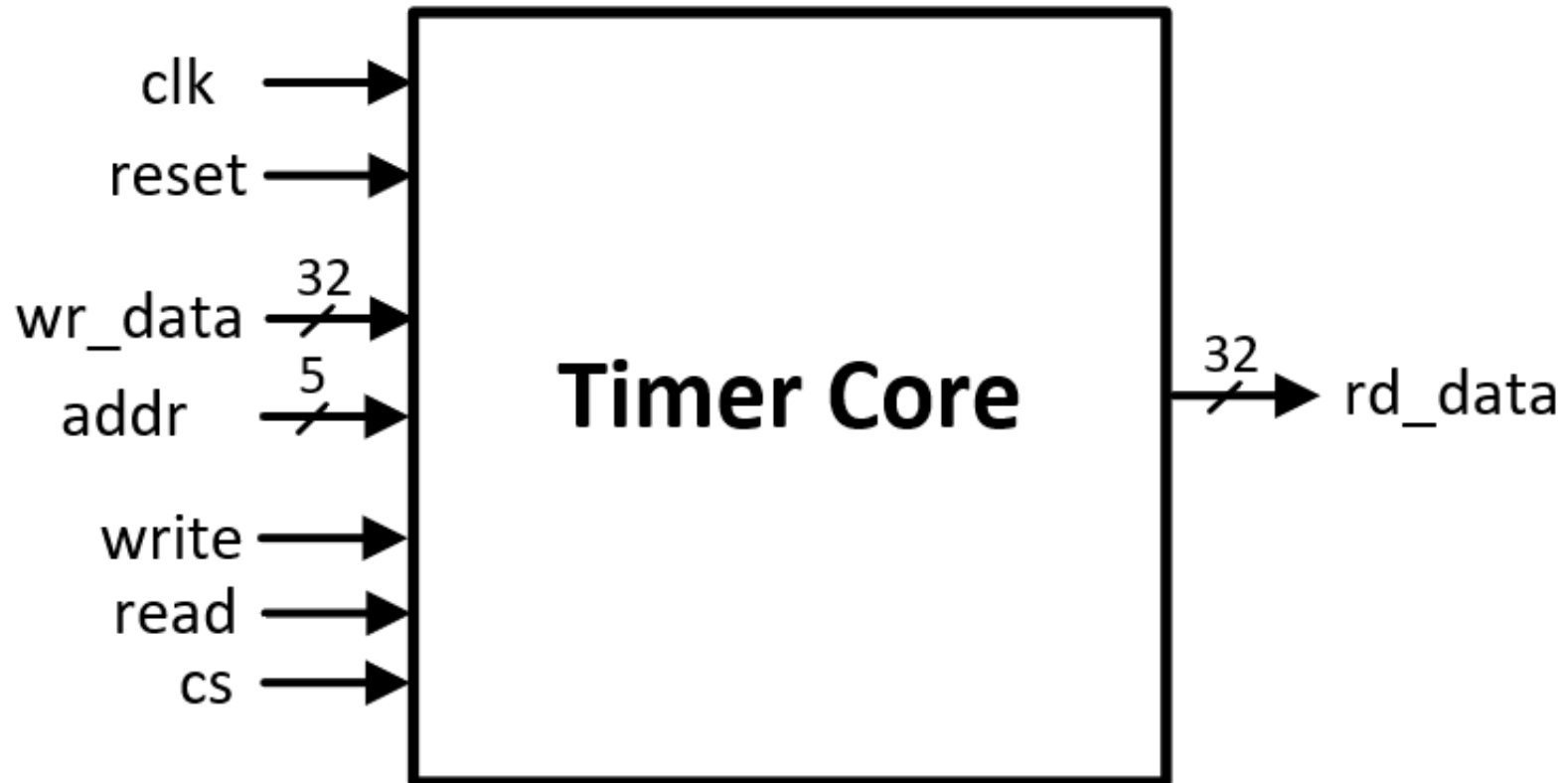
cs	write	read	addr(1:0)	wr_data(1:0)	Operation
1	0	1	00	--	Read lower word
1	0	1	01	--	Read upper word
1	1	0	10	01	Set "go" signal
1	1	0	10	00	Reset "go" signal
1	1	0	10	1-	Set "clear" signal

Wrapping Circuit



cs	write	read	addr(1:0)	wr_data(1:0)	Operation
1	0	1	00	--	Read lower word
1	0	1	01	--	Read upper word
1	1	0	10	01	Set "go" signal
1	1	0	10	00	Reset "go" signal
1	1	0	10	1-	Set "clear" signal

Interface of the Timer Core



Timer Core in VHDL (1)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity chu_timer is
    port(
        clk      : in  std_logic;
        reset    : in  std_logic;
        -- slot interface
        cs       : in  std_logic;
        write    : in  std_logic;
        read     : in  std_logic;
        addr     : in  std_logic_vector(4 downto 0);
        rd_data  : out std_logic_vector(31 downto 0);
        wr_data  : in  std_logic_vector(31 downto 0)
    );
end chu_timer;
```

Timer Core in VHDL (2)

```
architecture arch of chu_timer is
    signal count_reg  : unsigned(47 downto 0);
    signal count_next : unsigned(47 downto 0);
    signal ctrl_reg   : std_logic;
    signal wr_en      : std_logic;
    signal clear, go  : std_logic;
begin
    -- *****
    -- counter
    -- *****
    -- register
    process(clk, reset)
    begin
        if reset = '1' then
            count_reg <= (others => '0');
        elsif (clk'event and clk = '1') then
            count_reg <= count_next;
        end if;
    end process;
end arch;
```

Timer Core in VHDL (3)

```
-- next-state logic
count_next <= (others => '0') when clear = '1' else
               count_reg + 1   when go = '1' else
               count_reg;

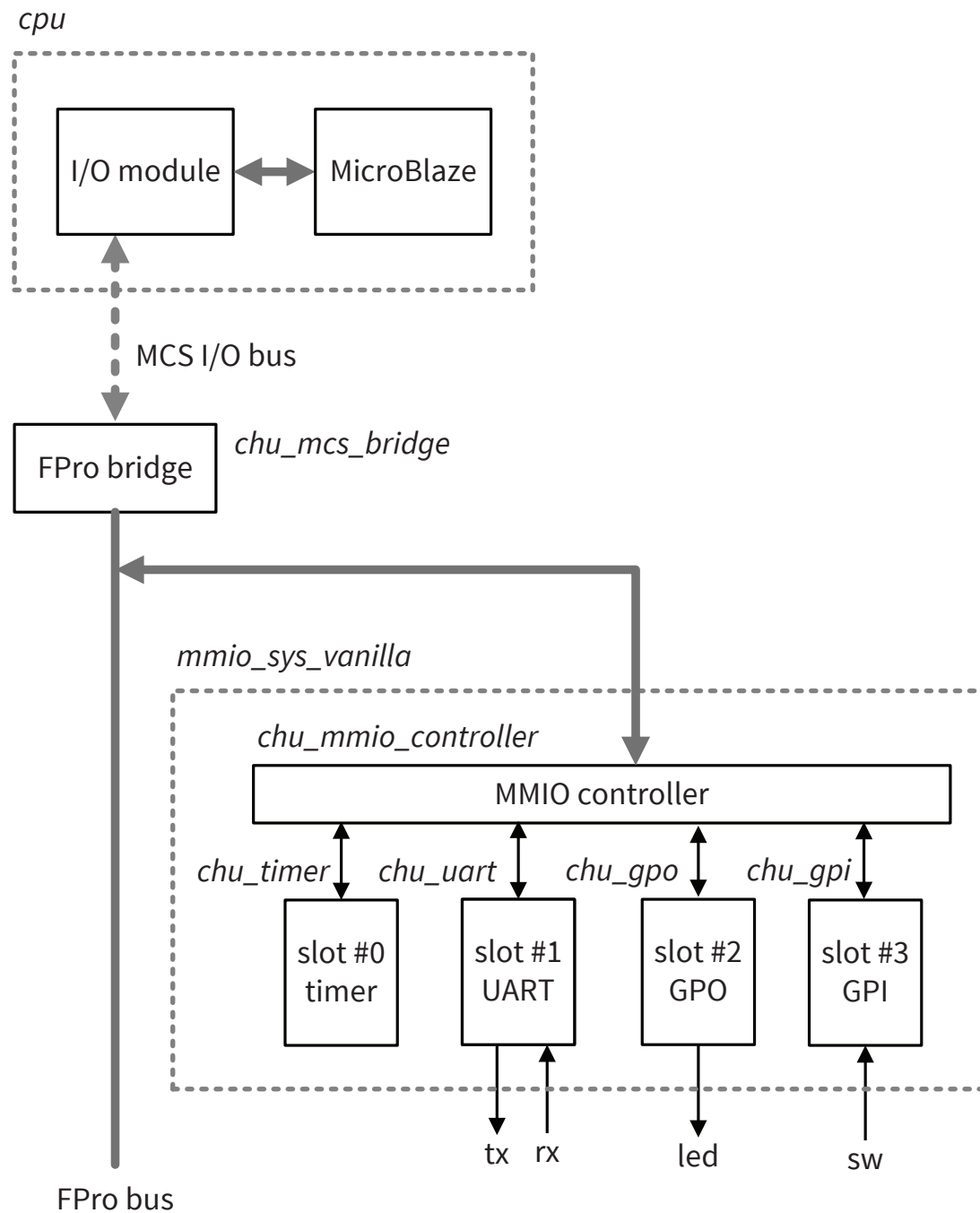
-- ctrl register
process(clk, reset)
begin
    if reset = '1' then
        ctrl_reg <= '0';
    elsif (clk'event and clk = '1') then
        if wr_en = '1' then
            ctrl_reg <= wr_data(0);
        end if;
    end if;
end process;
```

Timer Core in VHDL (4)

```
-- decoding logic
wr_en <=
    '1' when write='1' and cs='1' and addr(1 downto 0)="10" else '0';
clear <= '1' when wr_en='1' and wr_data(1)='1' else '0';
go      <= ctrl_reg;

-- slot read multiplexing
rd_data <=
    std_logic_vector(count_reg(31 downto 0)) when addr(0)='0' else
    x"0000" & std_logic_vector(count_reg(47 downto 32));
end arch;
```

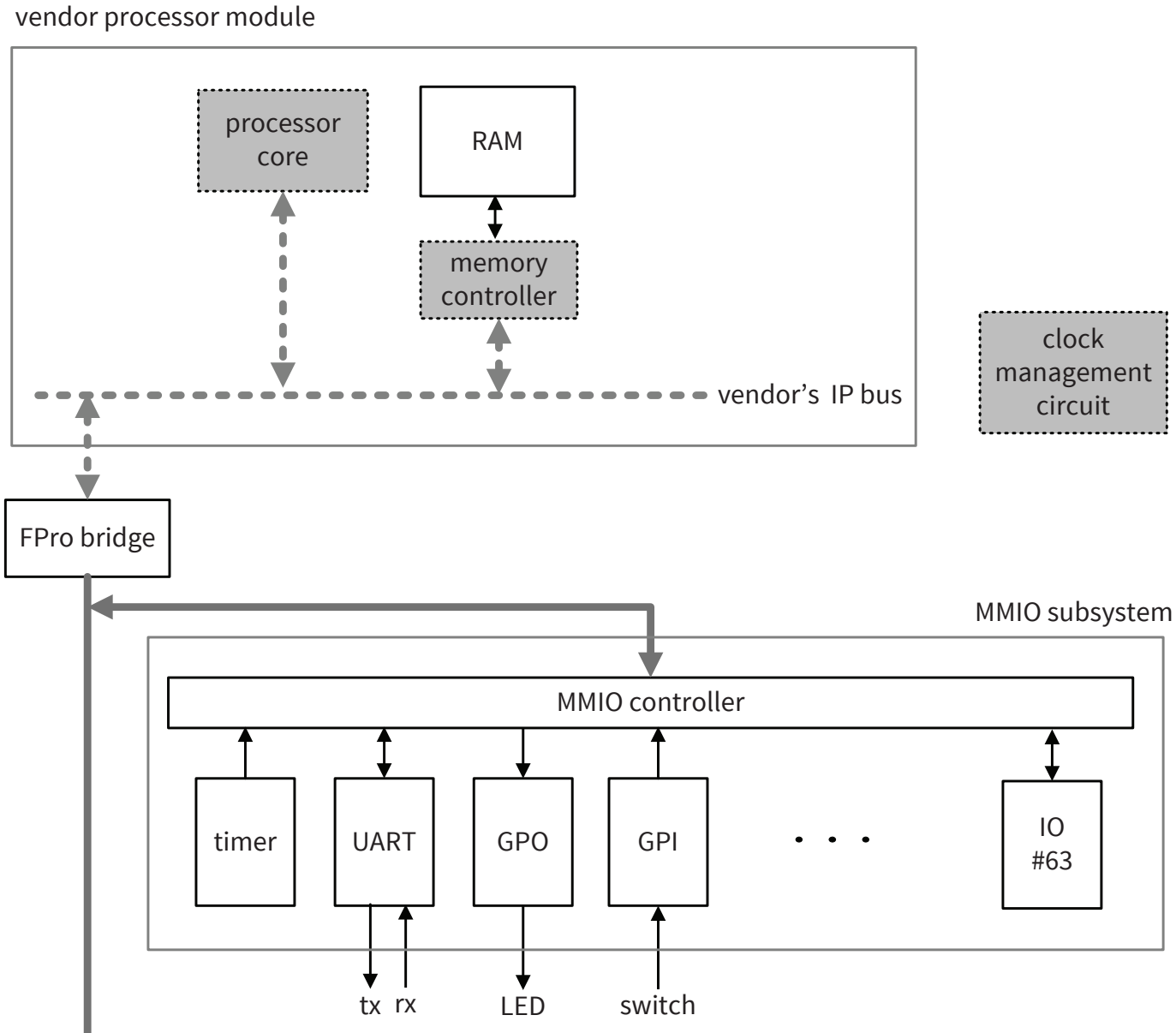
Vanilla FPro System



MMIO Cores of the Vanilla FPro System

- **Timer**
- **UART: Universal Asynchronous Receiver and Transmitter**
(for communication with a terminal emulator program on a PC, including displaying debug messages)
- **GPO: General-Purpose Output** (for driving LEDs)
- **GPI: General-Purpose Input** (for reading values of switches)

Sampler FPro System



MMIO Cores of the Sampler FPro System

- **Timer**
- **UART: Universal Asynchronous Receiver and Transmitter**
- **GPO: General-Purpose Output (for driving LEDs)**
- **GPI: General-Purpose Input (for reading values of switches)**
- **User (custom user IP)**
- XADC: Xilinx Analog-to-Digital Converter
- PWM: Pulse-Width Modulation
- **Debouncing (for sensing buttons)**
- **LED-MUX (for driving seven-segment displays)**
- SPI: Serial Peripheral Interface
- I²C (inter-IC)
- PS2 (for communication with a keyboard or a mouse)
- DDFS: Direct Digital Frequency Synthesis (for communication systems)
- ADSR: attack-decay-sustain-release (for music synthesizers)³¹