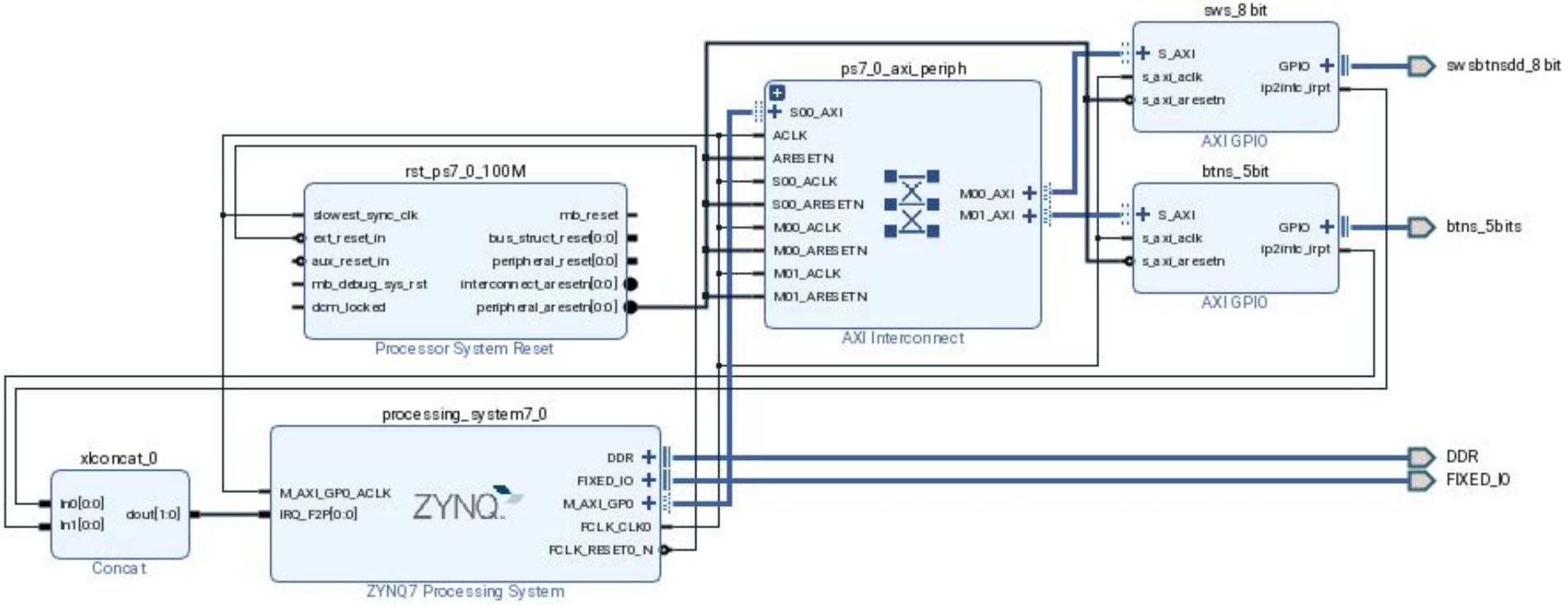
Embedded Systems (ECE340) Project 2

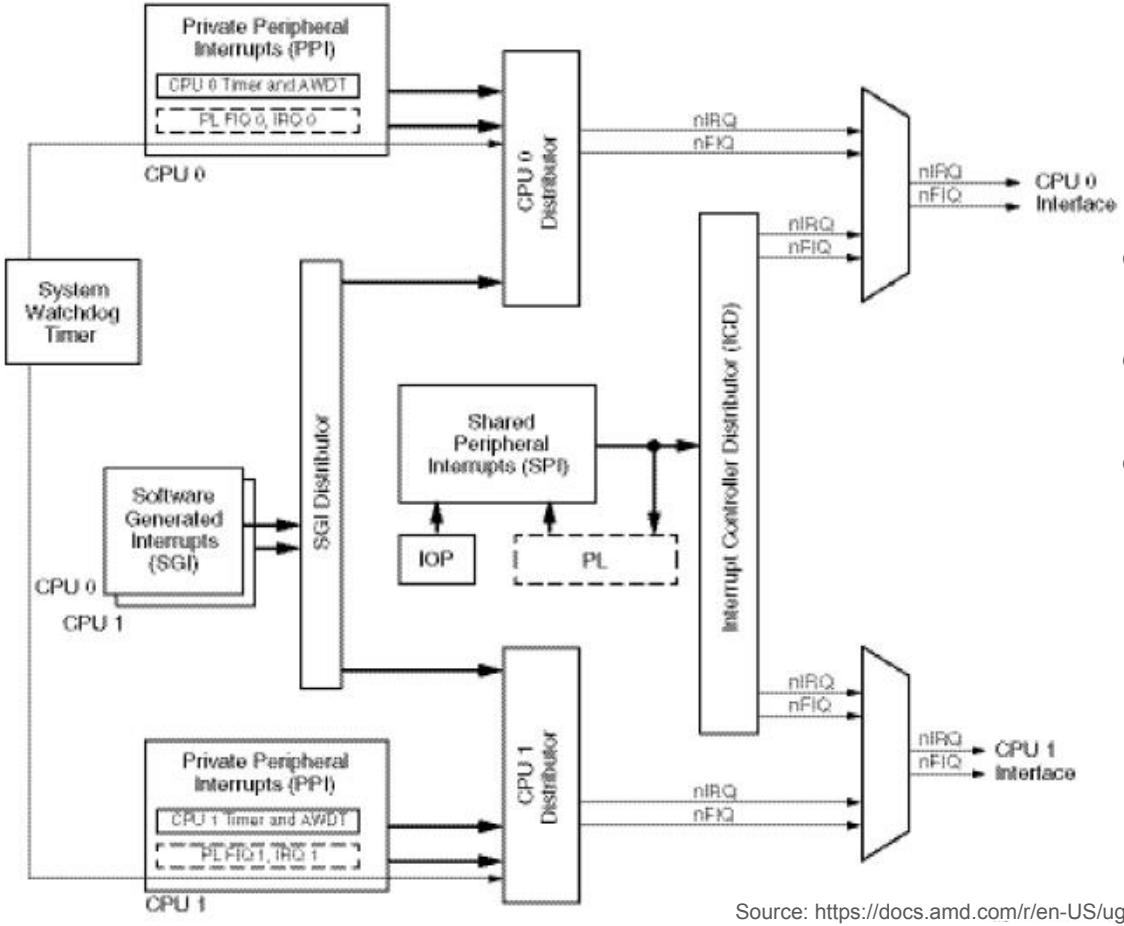
Χριστόδουλος Ζερδαλής & Τσιαντός Δημήτριος

(3531 & 3796)

Step 1: Interrupts


sws_8 bit
S_AXI
s_axi_aclk
s_axi_aresetn
GPIO
ip2intc_irpt
AXI GPIO
swsbtnsdd_8 bit
ps7_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M01_ACLK
M01_ARESETN
M00_AXI
M01_AXI
AXI Interconnect
btns_5bit
S_AXI
s_axi_aclk
s_axi_aresetn
GPIO
ip2intc_irpt
AXI GPIO
btns_5bits
rst_ps7_0_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset
processing_system7_0
M_AXI_GP0_ACLK
IRQ_F2P[0:0]
ZYNQ
DDR
FIXED_IO
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System
DDR
FIXED_IO
xlconcat_0
In0[0:0]
In1[0:0]
dout[1:0]
Concat

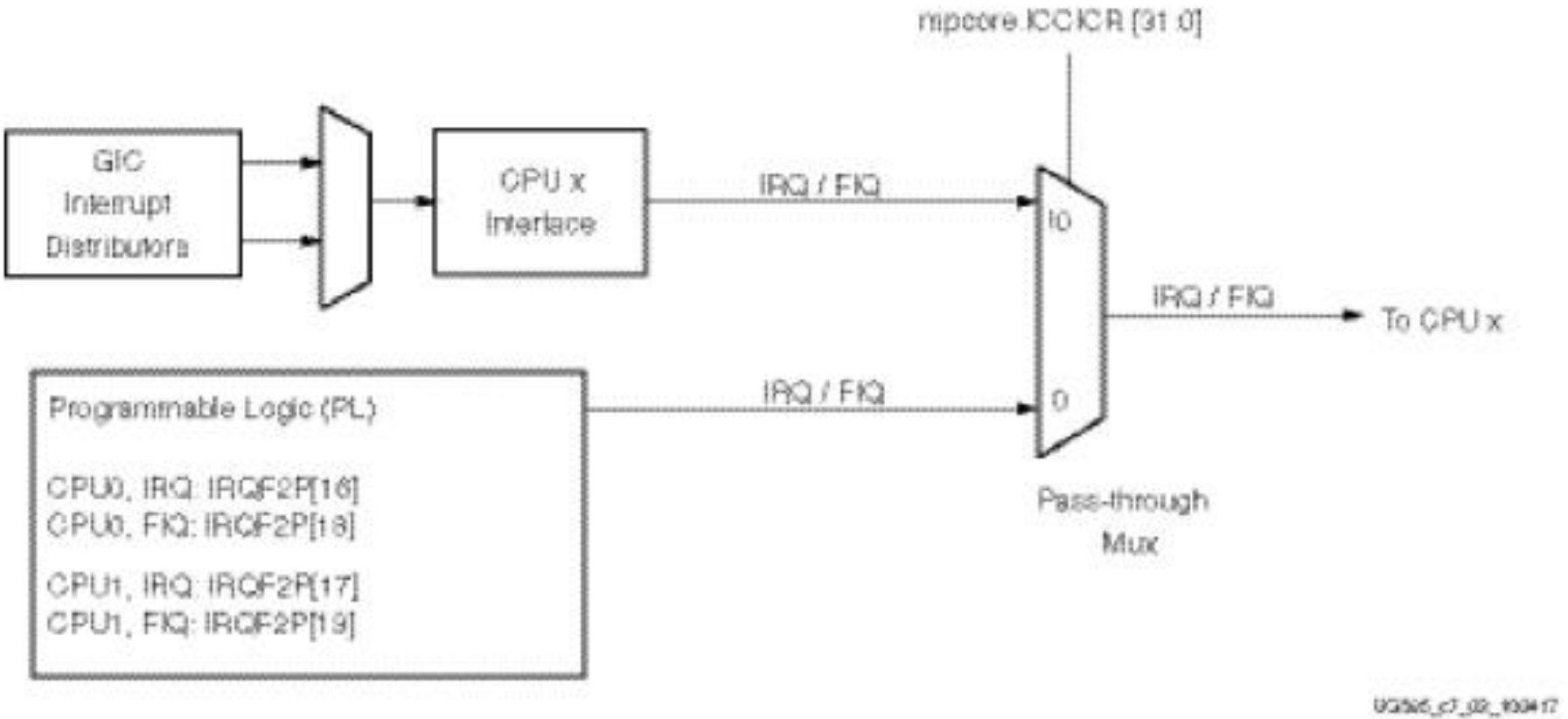
Generic Interrupt Controller



- Private Peripheral Interrupts (PPI)
- Software Generated Interrupts (SGI)
- Shared Peripheral Interrupts (SPI)
    - PL interrupts

Source: https://docs.amd.com/r/en-US/ug585-zynq-7000-SoC-TRM/Block-Diagram?tocId=gQMnmOTBqj9aw89RpKx3Ow

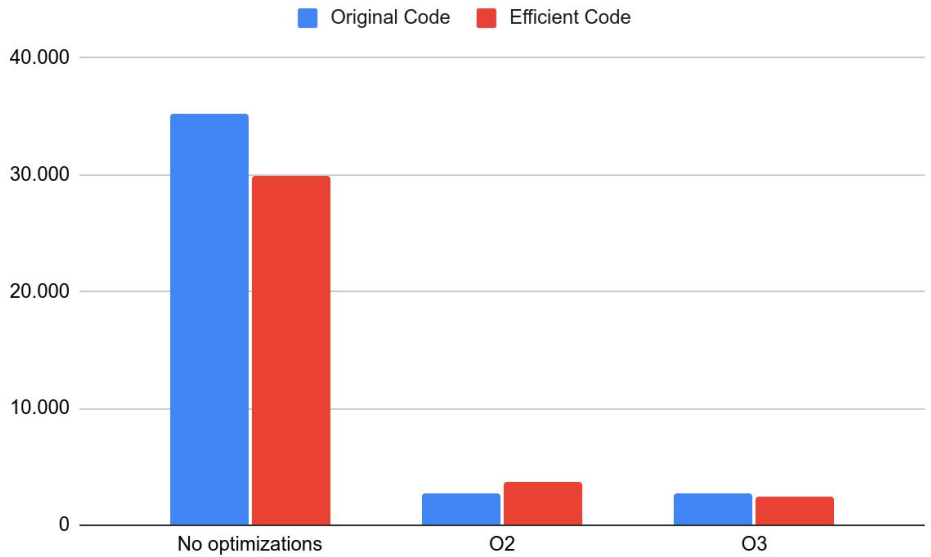
Interrupt Signal Pass-through



Step 2: Code Profiling

36x36 Matrix Multiplication

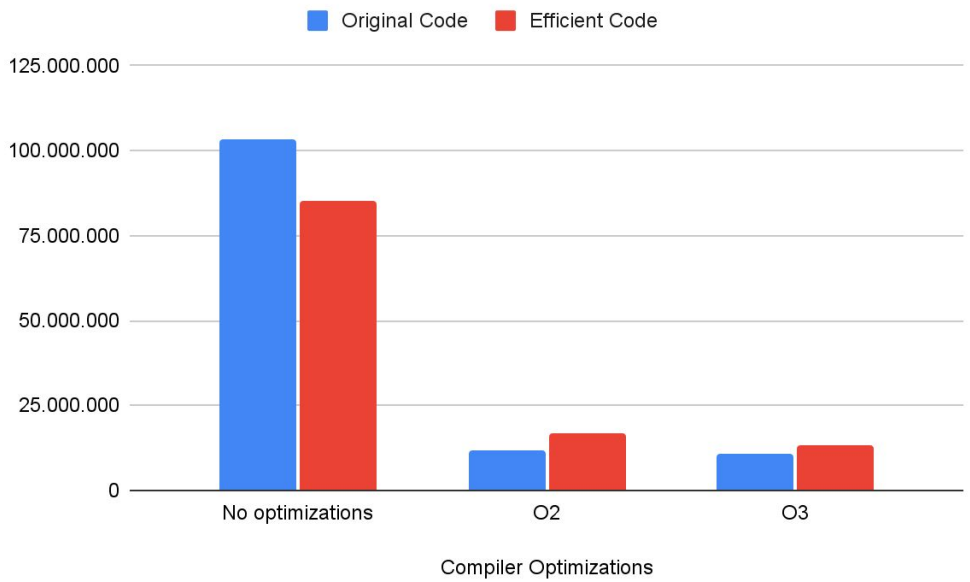
Compiler Optimizations	Original Code	Efficient Code
No optimizations	35.265	29.841
O2	2.703	3.650
O3	2.658	2.443



Total size = 36×36×4×3 = 5.184×3 = 15.552 bytes

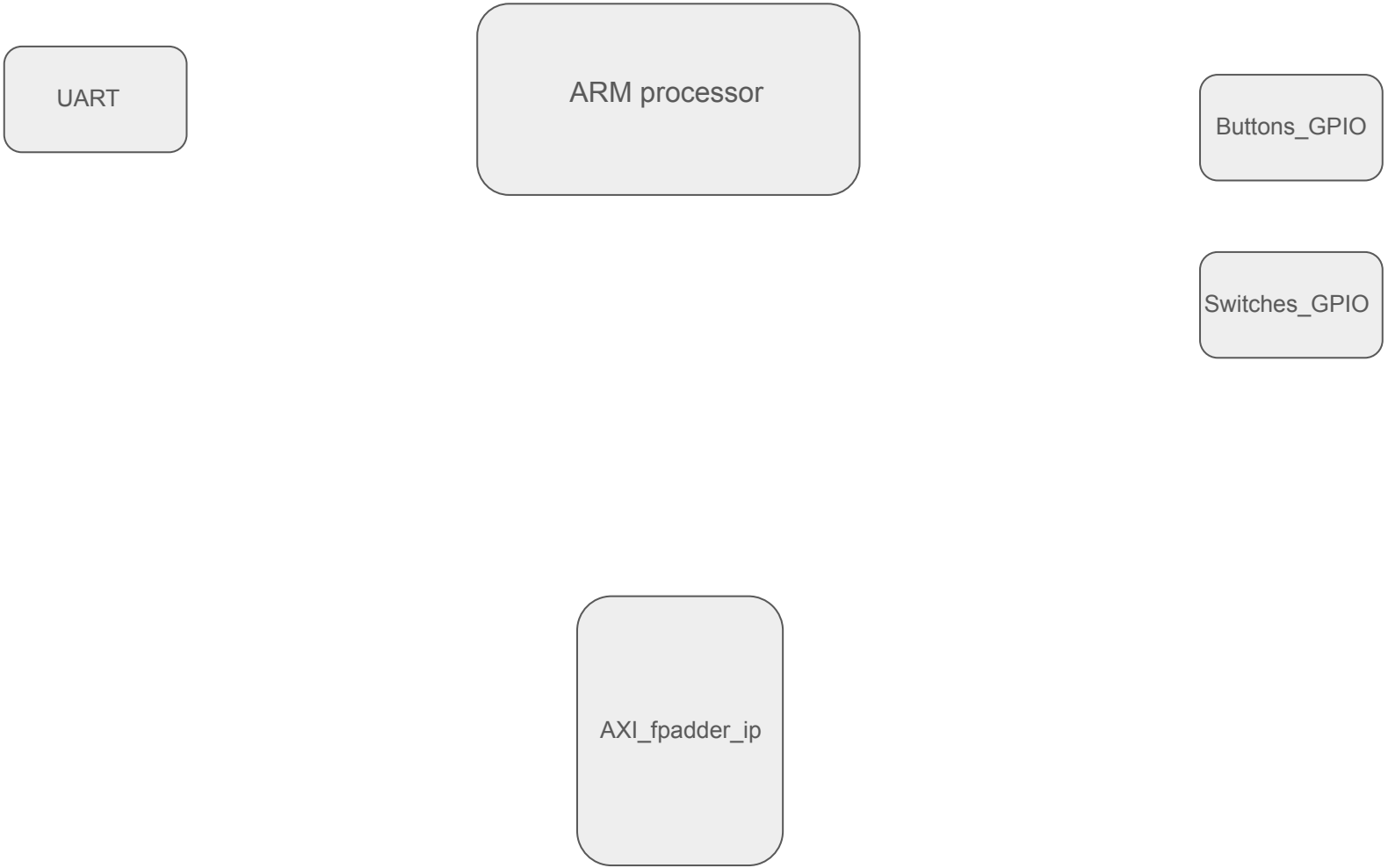
500x500 Matrix Multiplication

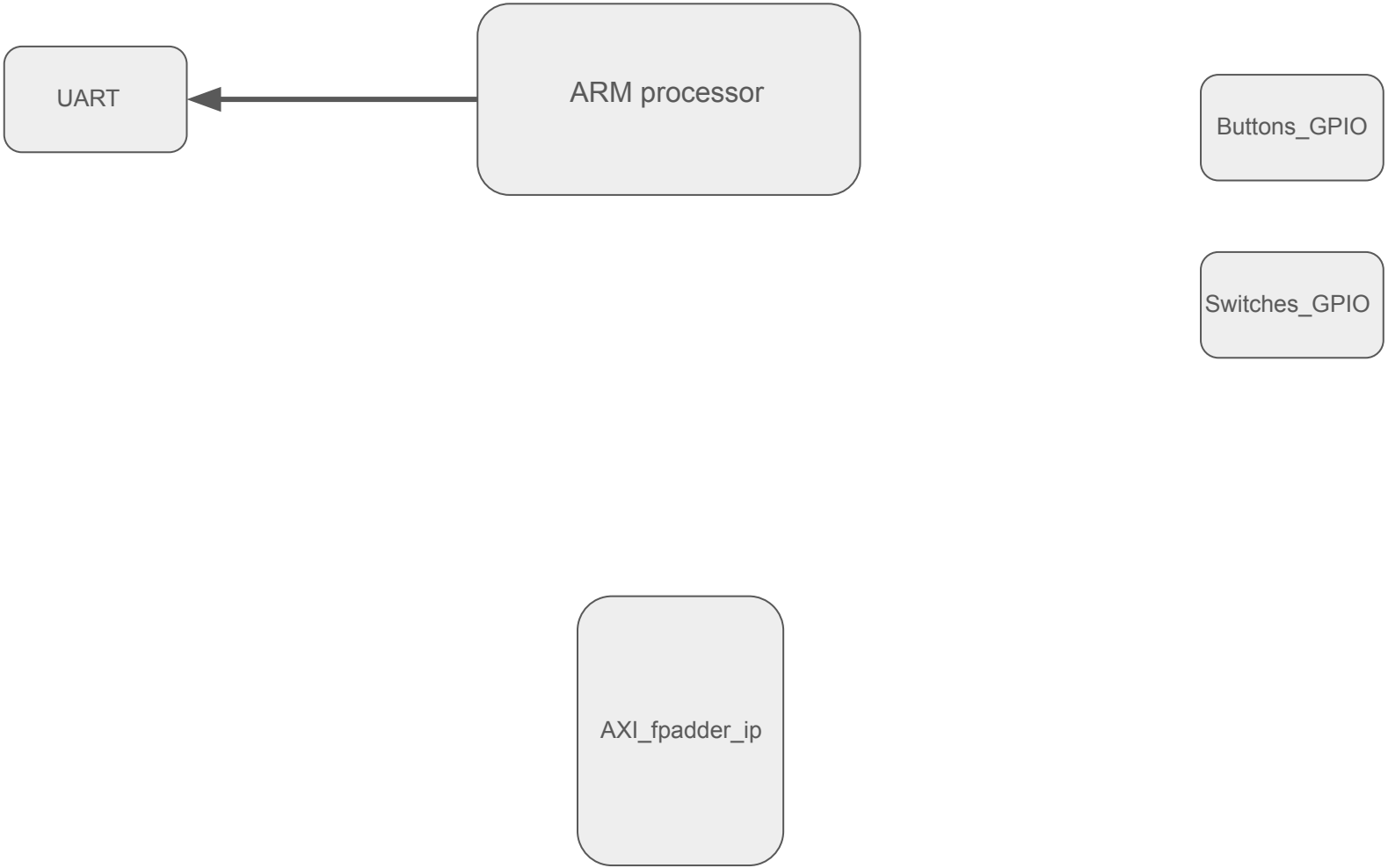
Compiler Optimizations	Original Code	Efficient Code
No optimizations	103.201.279	85.048.008
O2	11.967.299	16.808.782
O3	10.869.735	13.270.360

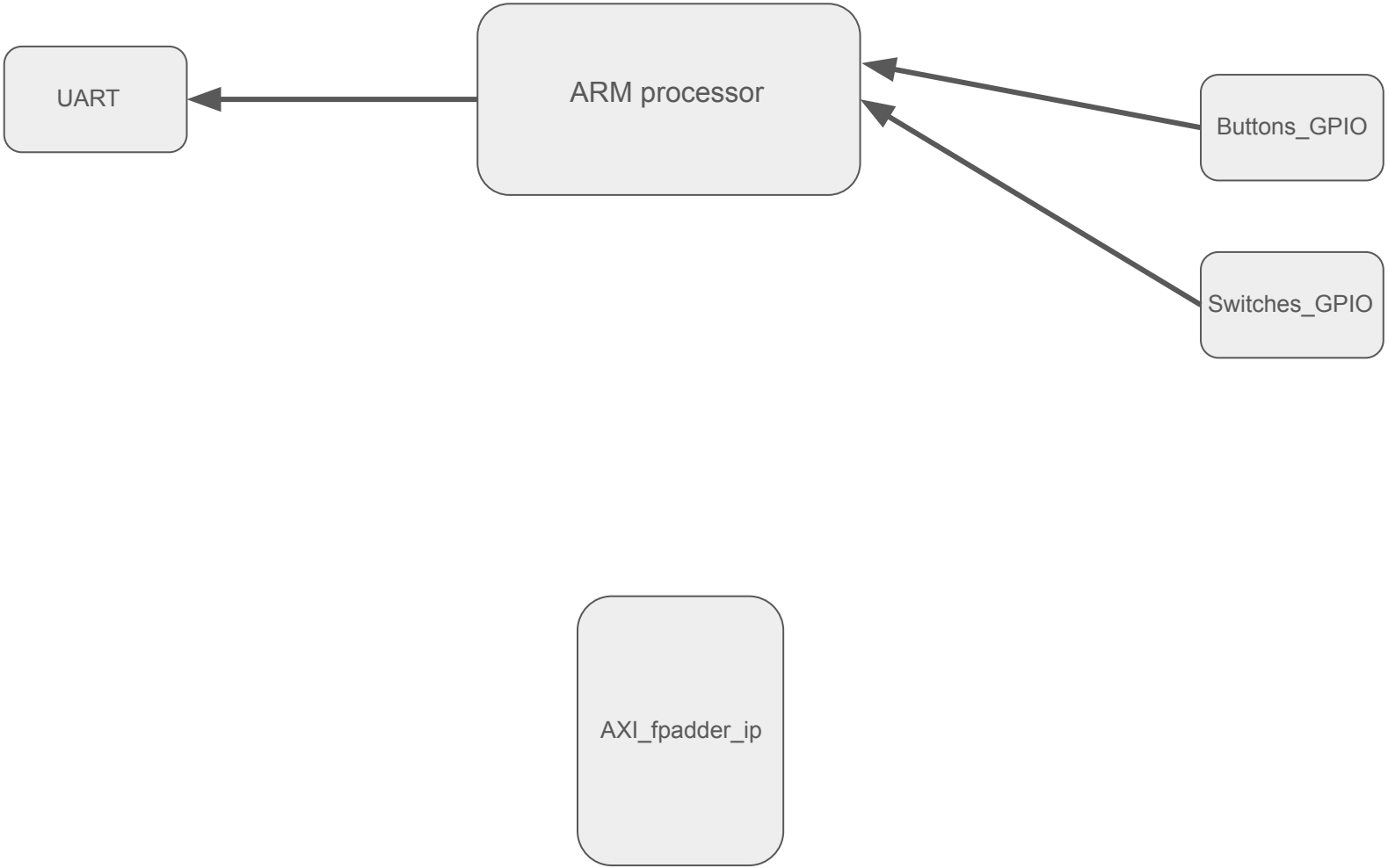


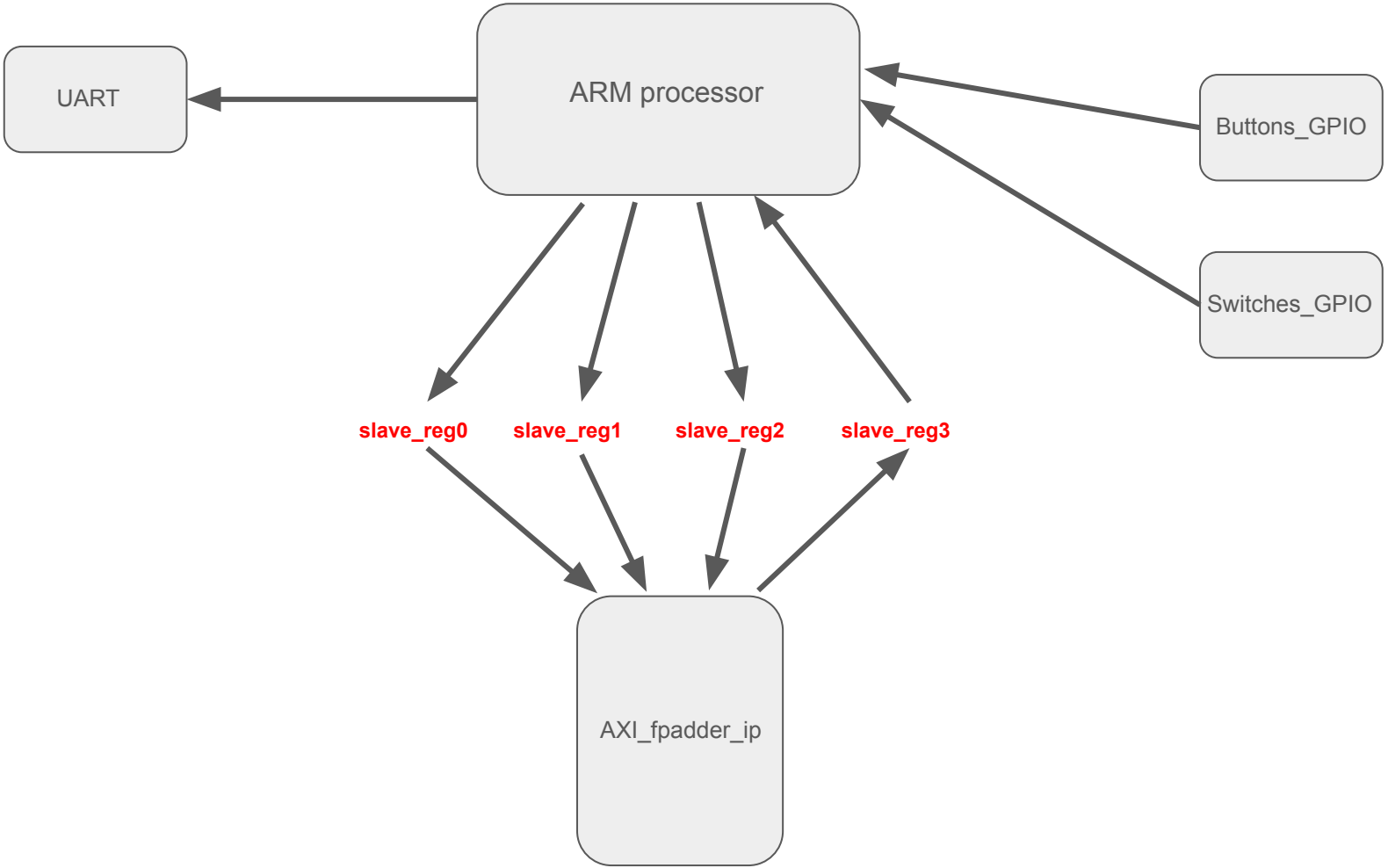
Total size = 500×500×4×3 = 1.000.000×3 = 3.000.000 bytes

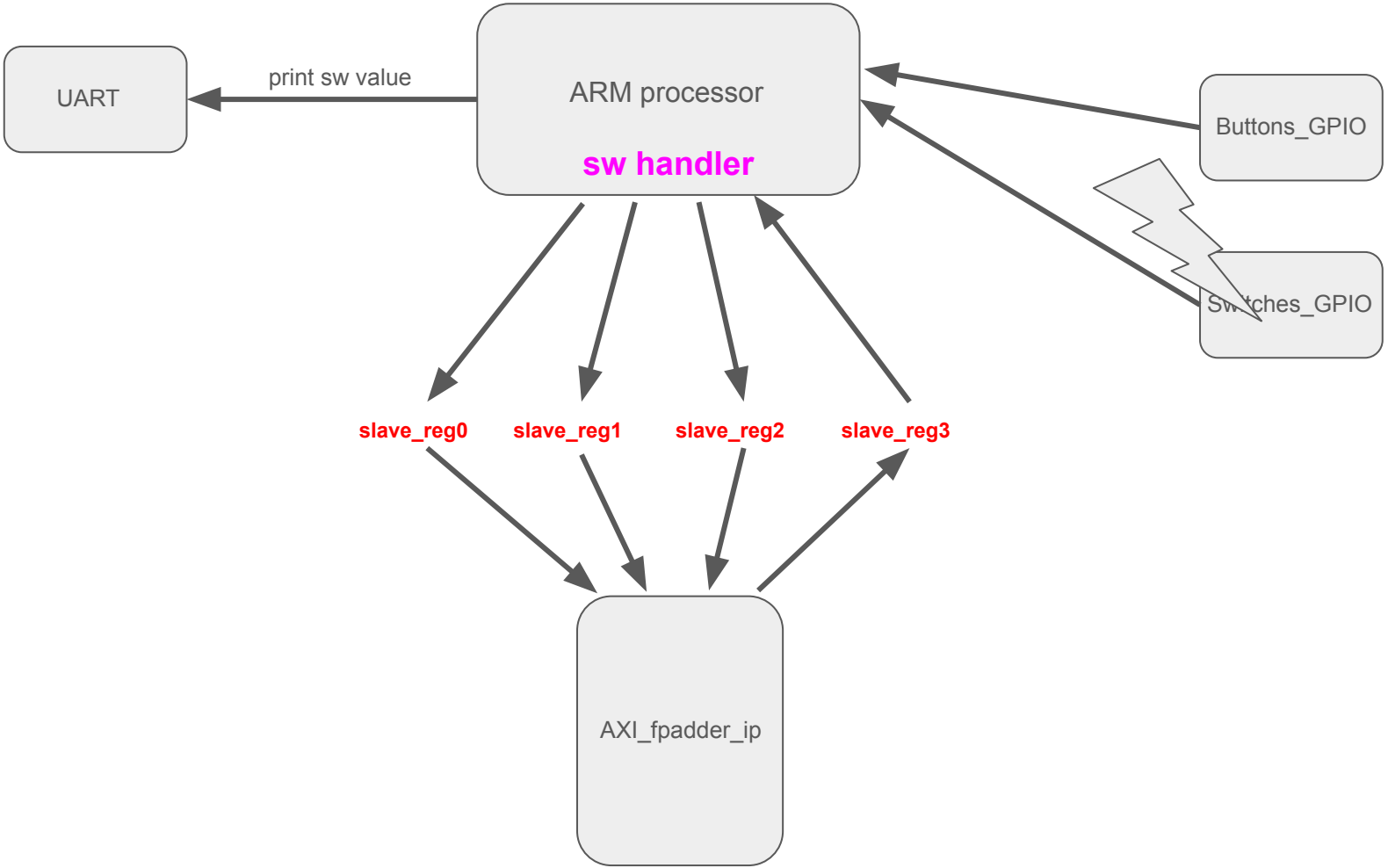
Step 3: Custom IP

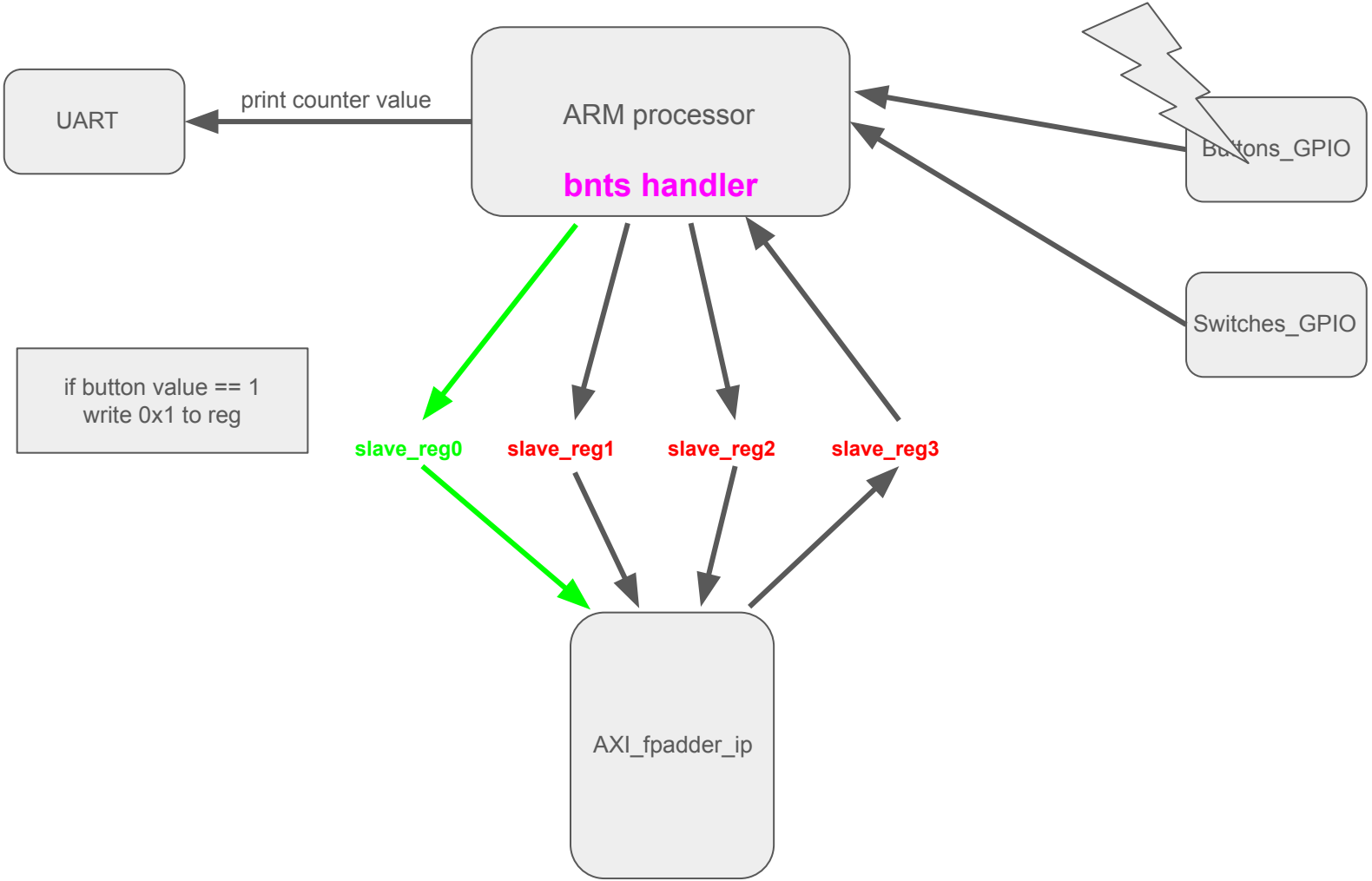

UART
ARM processor
Buttons_GPIO
Switches_GPIO
AXI_fpadder_ip

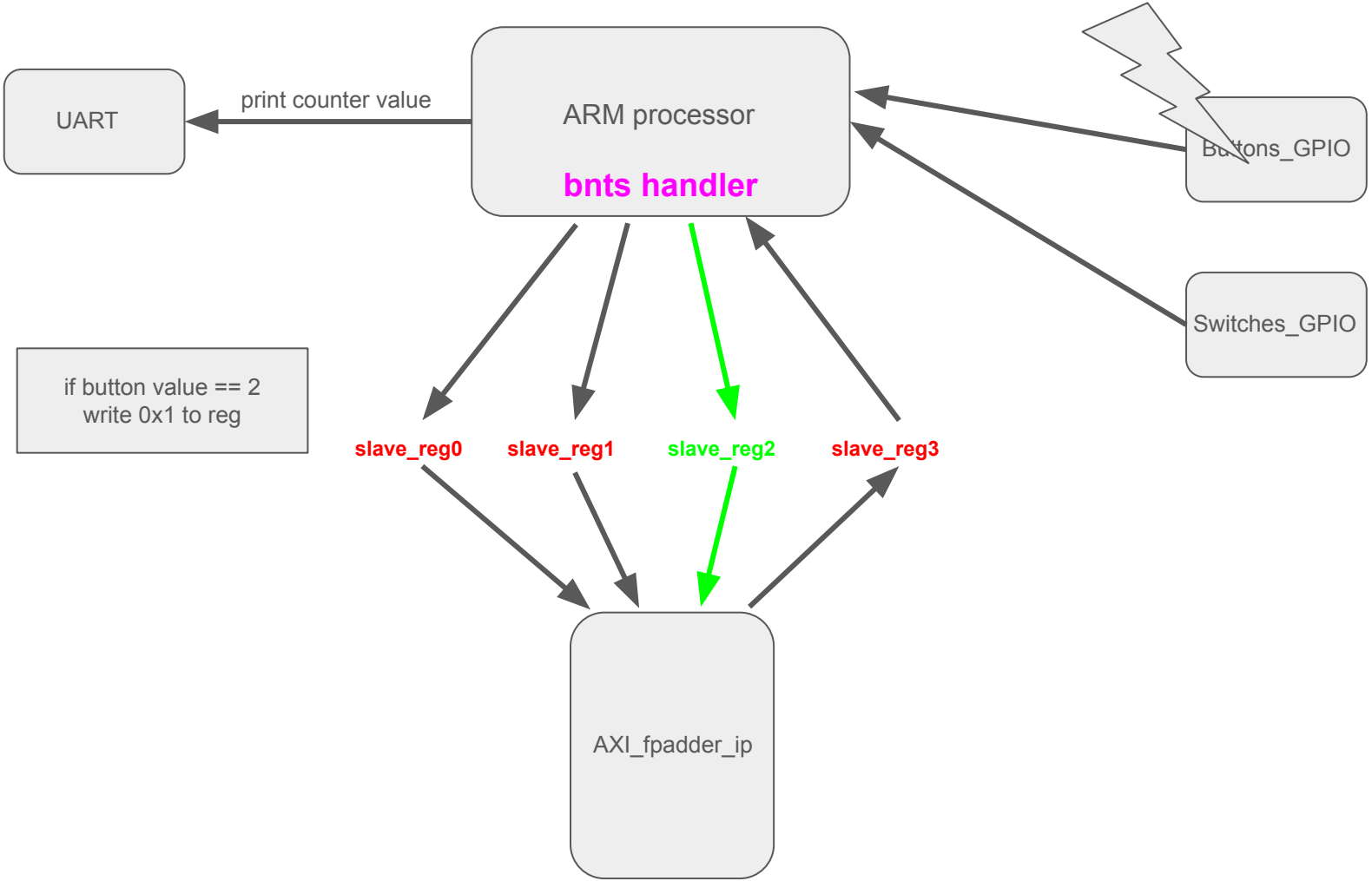

UART
ARM processor
Buttons_GPIO
Switches_GPIO
AXI_fpadder_ip

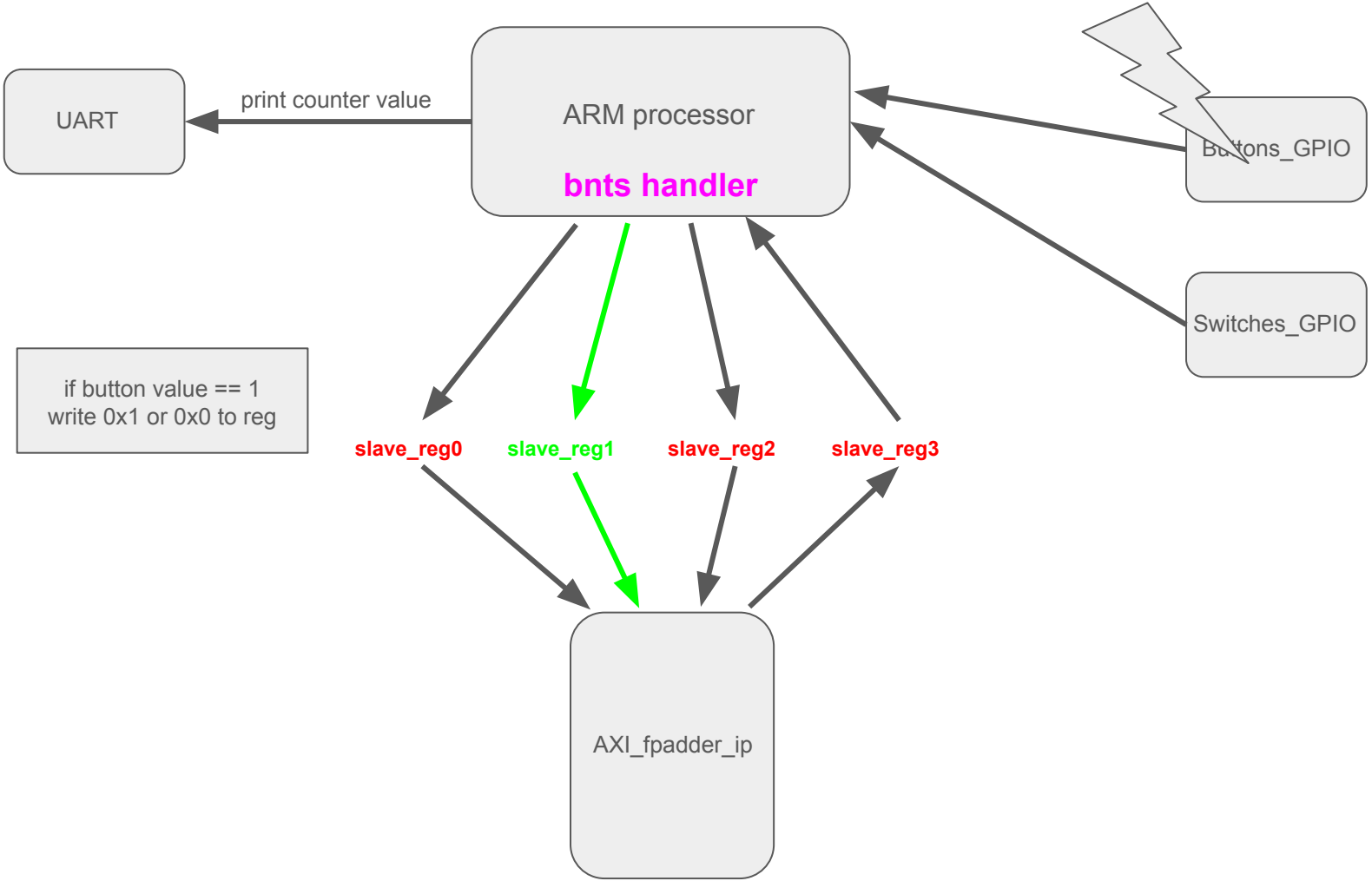

UART
ARM processor
Buttons_GPIO
Switches_GPIO
AXI_fpadder_ip

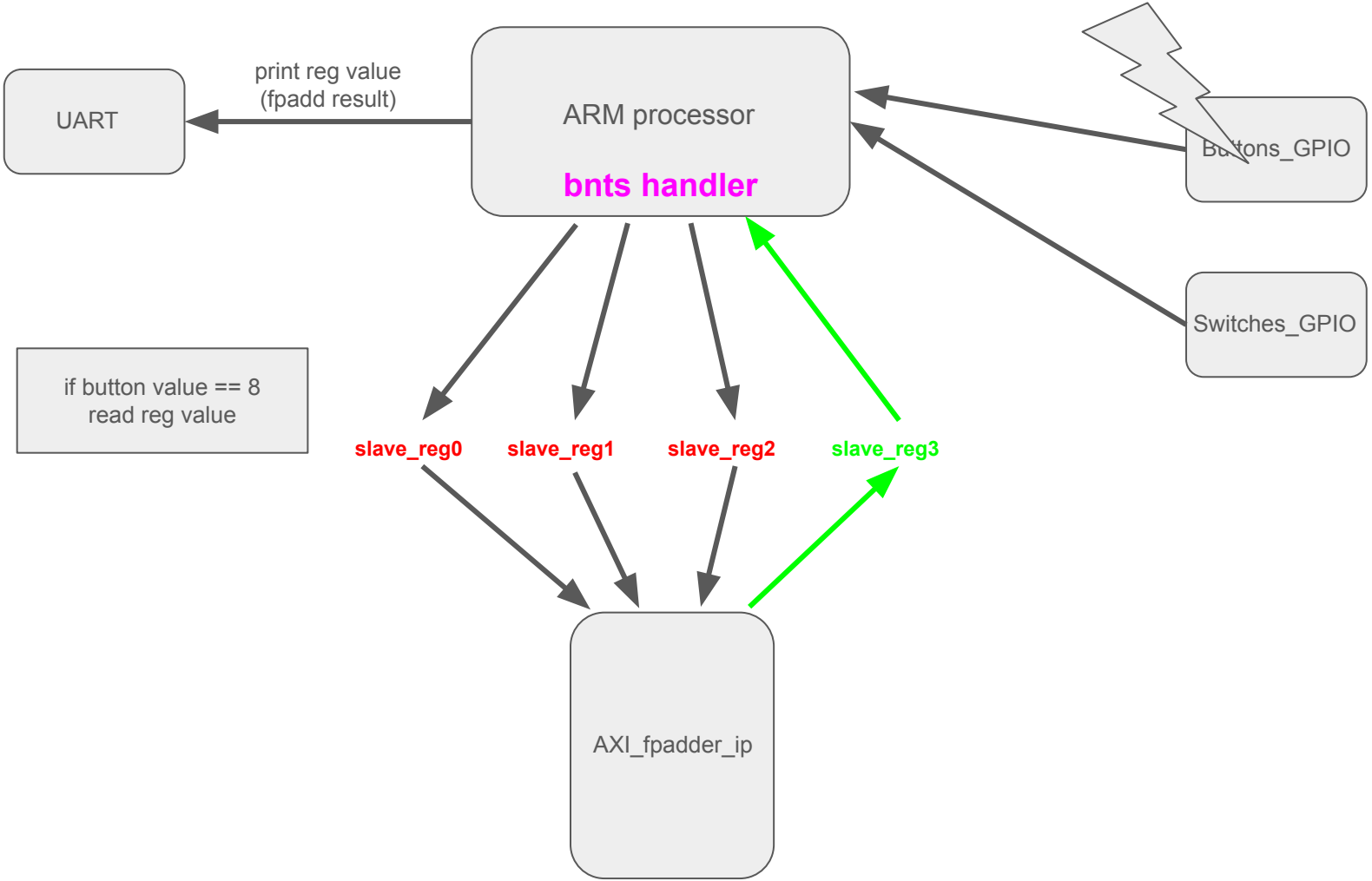

UART
ARM processor
Buttons_GPIO
Switches_GPIO
slave_reg0
slave_reg1
slave_reg2
slave_reg3
AXI_fpadder_ip


UART
print sw value
ARM processor
sw handler
Buttons_GPIO
Switches_GPIO
slave_reg0
slave_reg1
slave_reg2
slave_reg3
AXI_fpadder_ip


UART
print counter value
ARM processor
bnts handler
Buttons_GPIO
Switches_GPIO
if button value == 1
write 0x1 to reg
slave_reg0
slave_reg1
slave_reg2
slave_reg3
AXI_fpadder_ip


UART
print counter value
ARM processor
bnts handler
Buttons_GPIO
Switches_GPIO
if button value == 2
write 0x1 to reg
slave_reg0
slave_reg1
slave_reg2
slave_reg3
AXI_fpadder_ip


UART
print counter value
ARM processor
bnts handler
Buttons_GPIO
Switches_GPIO
if button value == 1
write 0x1 or 0x0 to reg
slave_reg0
slave_reg1
slave_reg2
slave_reg3
AXI_fpadder_ip


UART
print reg value
(fpadd result)
ARM processor
bnts handler
Buttons_GPIO
Switches_GPIO
if button value == 8
read reg value
slave_reg0
slave_reg1
slave_reg2
slave_reg3
AXI_fpadder_ip

FP Addition Testing

Hex A	Hex B	Hex Result	A (Decimal)	B (Decimal)	A + B (Decimal)
3f800000	40000000	**40400000**	1.0	2.0	3.0
bf800000	3f800000	**0**	-1.0	1.0	0.0
c2de8000	45155	**450e6a00**	-111.25	2.389.875	2.278.625
6b64b235	6ac49214	**6ba37d9f**	27.647.700.000.000.000.000.000.000.000.000.000	1.188.200.000.000.000.000.000.000.000.000.000	39.529.600.000.000.000.000.000.000.000.000.000
2ac49214	6ac49214	**6ac49214**	0	1.188.200.000.000.000.000.000.000.000.000.000	1.188.200.000.000.000.000.000.000.000.000.000
bfc66666	3fc7ae14	**3c23d700**	-1.55	1.56	9.999.990.463.256.830
c565ee8b	4565ee8a	**b9800000**	-3.678.908.935.546.870	367.890.869.140.625	-24.414.061.999.778
447a4efa	c47a1ccd	**3f48b400**	10.012.340.087.890.600	-100.045.001.220.703.000	78.399.658.203.125

Step 1a

- create vivado project **lab2_simple_arm**
- create **block_design**
- add **zynq7 PS IP**
- disable all IO peripherals except **UART1, GPIO, APU** timer
- create and configure the **GPIOs** for btns and sws
- connect GPIOs with Processing System
- **assign addresses** to the inputs (btns, sws)
- generate the HDL: the **wrapper** (verilog file)
- synthesize & generate bitstream

Step 1b

- export the **hardware** (.xsa) from Vivado
- create new vitis **platform & project**
- check **heap & stack** sizes
- run lab2.c

Step 1b. a

- **polling** vs **interrupts**
- **non-maskable** vs **maskable** interrupts
- **interrupt controller** and priorities (GIC)
- interrupt types

Step 1b. b

- Enable interrupts in buttons GPIO
- Connect interrupts from button to the Zynq PS
- Implement adder functionality in C