End-to-End On-Chip Encryption to Prevent Physical Attacks

Jonas Zöschg
Advisor: Harshavardhan Unnibhavi
Chair of Distributed Systems and Operating Systems
https://dse.in.tum.de/

15.11.2022 – 15.03.2023

Introduction

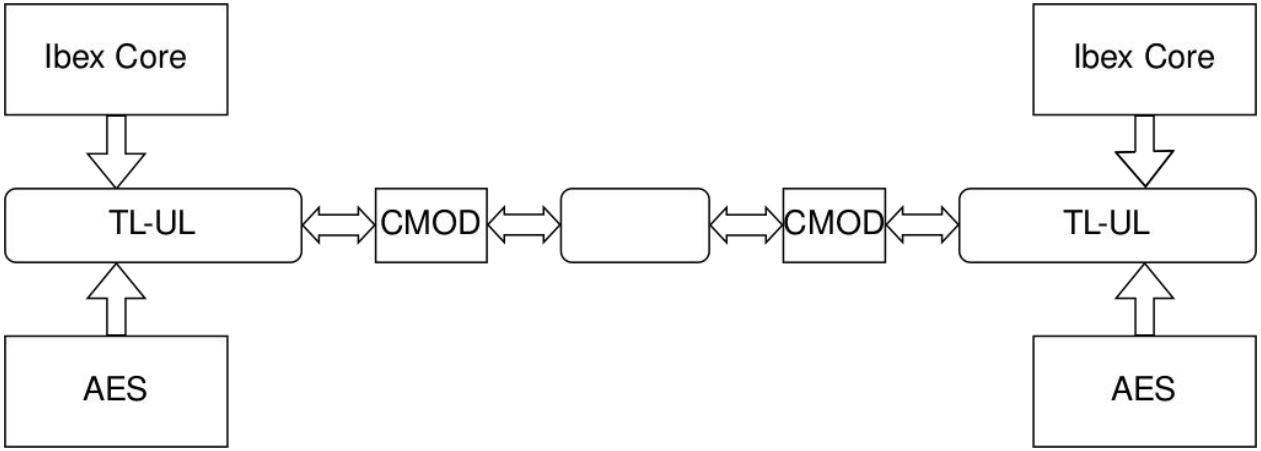
- **OpenTitan:** Project that builds a transparent reference design and integration guidelines for **silicon Root of Trust (RoT)** chips.
- Many measurements against physical attacks
- Designed for single core chips

Implementation, Integration, and Evaluation of a Hardware Design that enables End-to-End On-Chip Encryption

Outline

- ~~Introduction~~
- Design
- Evaluation
- Conclusion
- Future Work

Design



- Flexibility
- Performance
- Comportability

Design: Transmission Process

1) Initialize
2) Write first data block to the input registers
3) For the remaining data blocks do:
   a) Write data to the input registers
   b) Read encrypted data
   c) Send encrypted data
4) Read last encrypted data block
5) Send last data block
6) Deinitialize

Design: Reception Process

1) Initialize
2) Until the last received data block was read:
   a) Read received data
   b) Write data to AES IP core
   c) If not the first iteration: Read the decrypted data
3) Read last decrypted data block
4) Deinitialize

Design: CMOD IP Core Registers Overview

Name	Description
cmod.INTR_STATE	Interrupt State Register
cmod.INTR_ENABLE	Interrupt Enable Register
cmod.INTR_TEST	Interrupt Test Register

cmod.ALERT_TEST	Alert Test Register

Design: CMOD IP Core Registers Overview

Name	Description
cmod.CTRL	CMOD Control Register
cmod.STATUS	CMOD Status Register

cmod.WDATA_{0-3}	Data to be transmitted.
cmod.RDATA_{0-3}	If RXVALID is 1, data that was received.

Outline

- ~~Introduction~~
- ~~Design~~
- Evaluation
- Conclusion
- Future Work

Evaluation: Key Questions

- Is the **implementation** of the CMOD IP core **correct**?
- What are the **overheads** of using the CMOD IP core?
- What is the **performance** of the end-to-end encrypted communication using our core?
- What is the **impact** of using **different cipher modes**?

Evaluation: Software

- **Functionality Test:**
  - Tests whether the CMOD IP core works as expected.
  - Tests edge cases
  - Checks if the registers are set correctly
- **Performance Test:**
  - Measures cycles taken by different parts of the communication process

Evaluation: Background Information

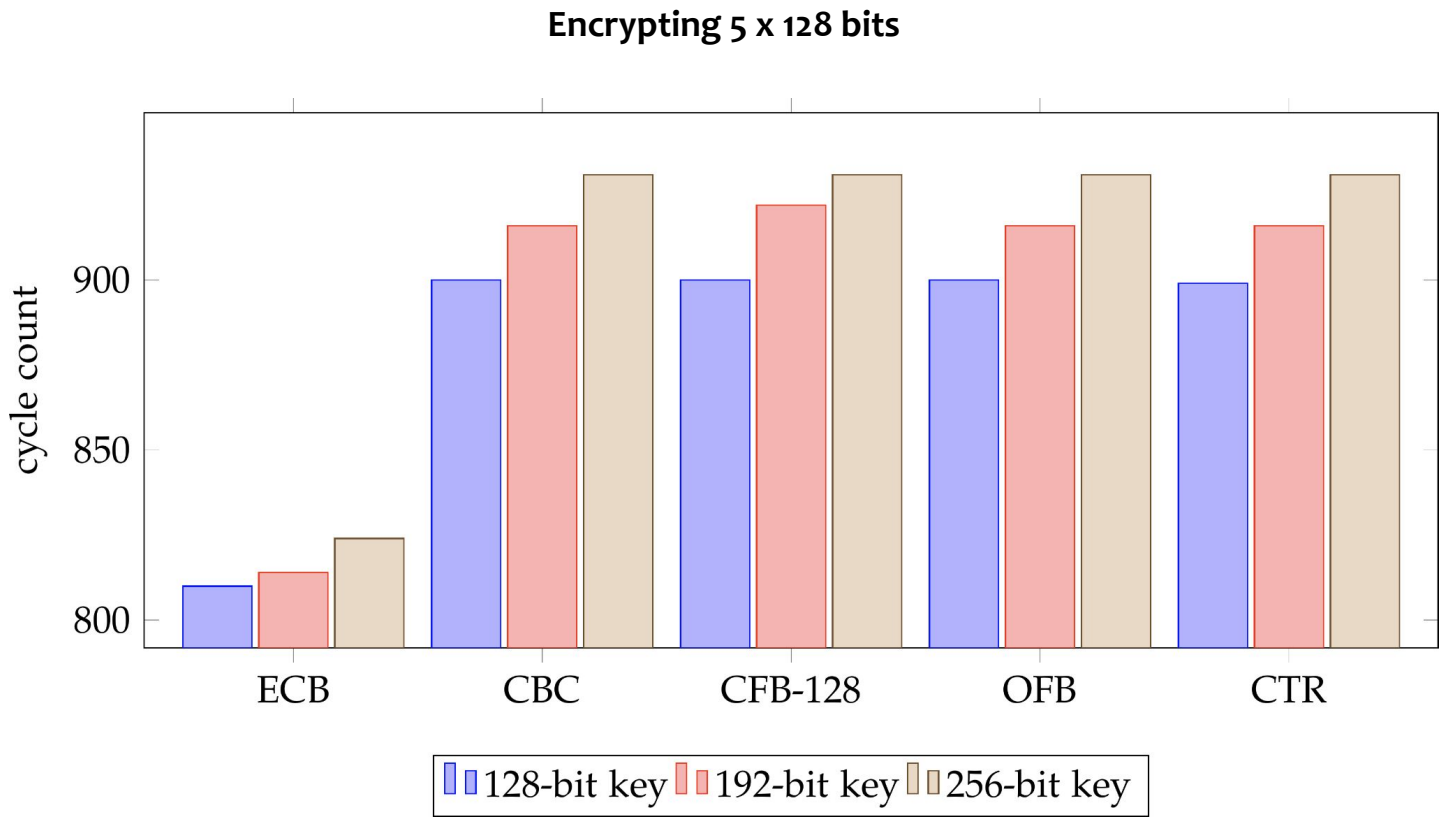
- Verilator
- Earl Grey:

  - Ibex Core with **Writeback Stage enabled**

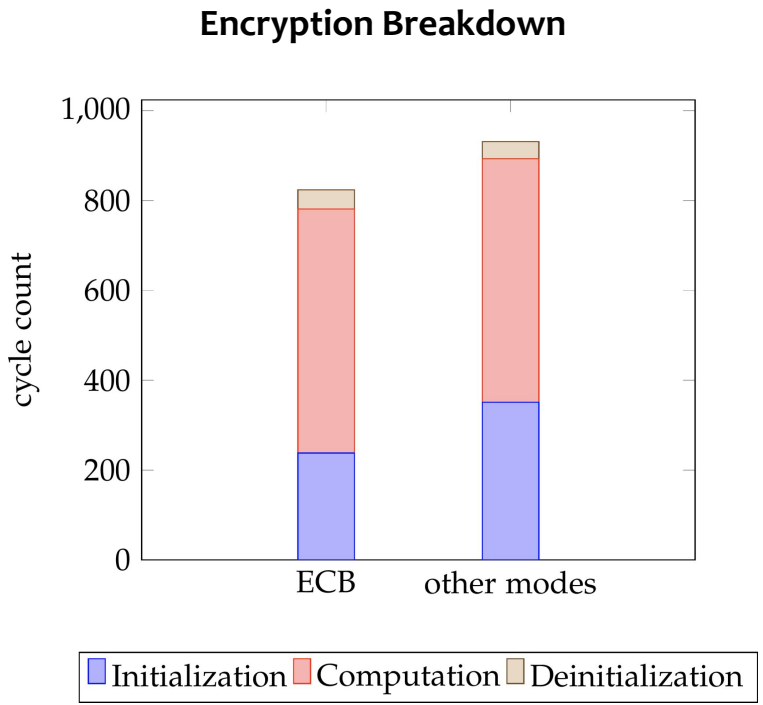
  - AES IP Core with **Masking enabled**

  - **Two CMOD IP cores**
- Measurements **Overhead** is **included**

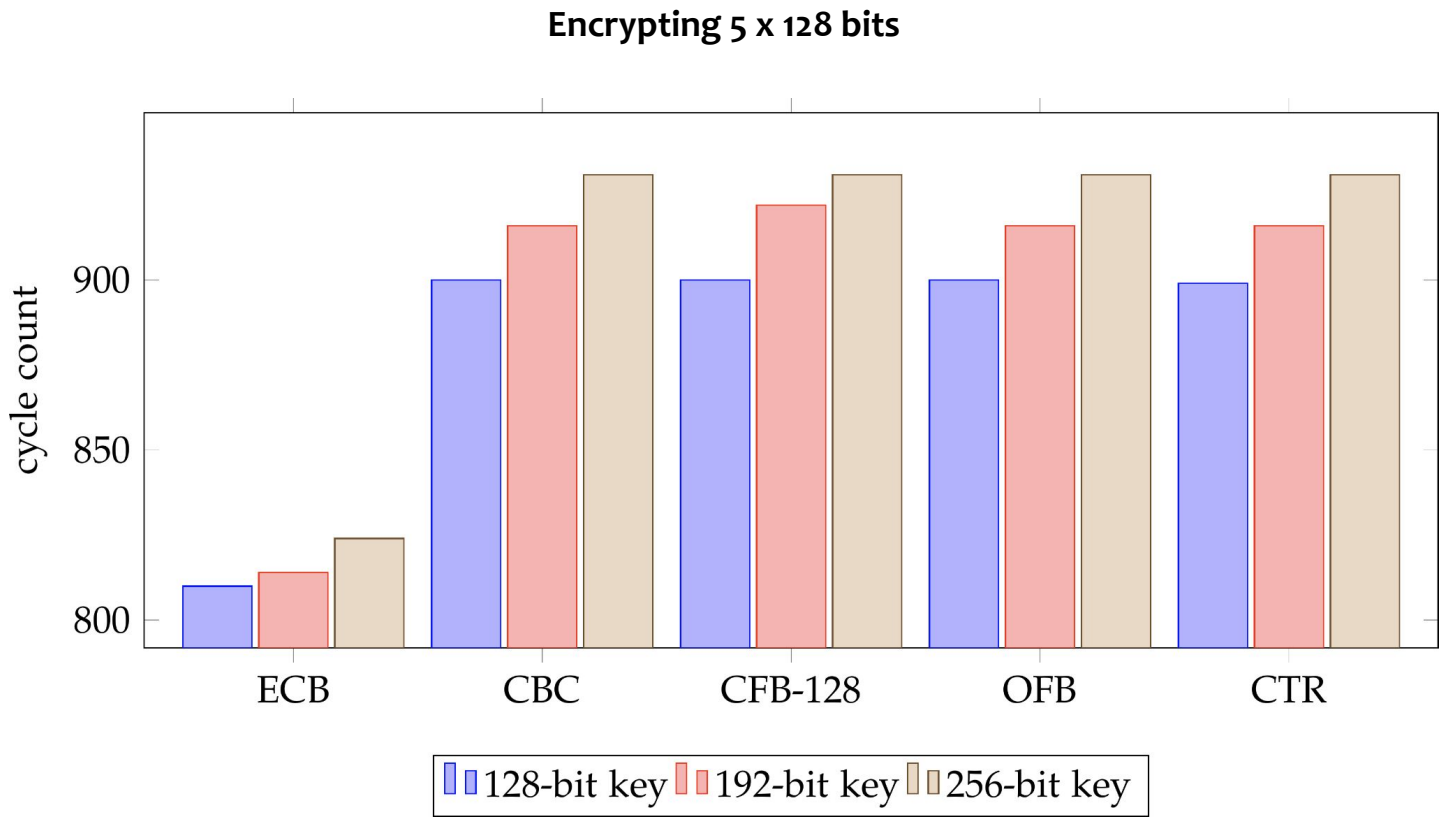
Evaluation: AES IP Core



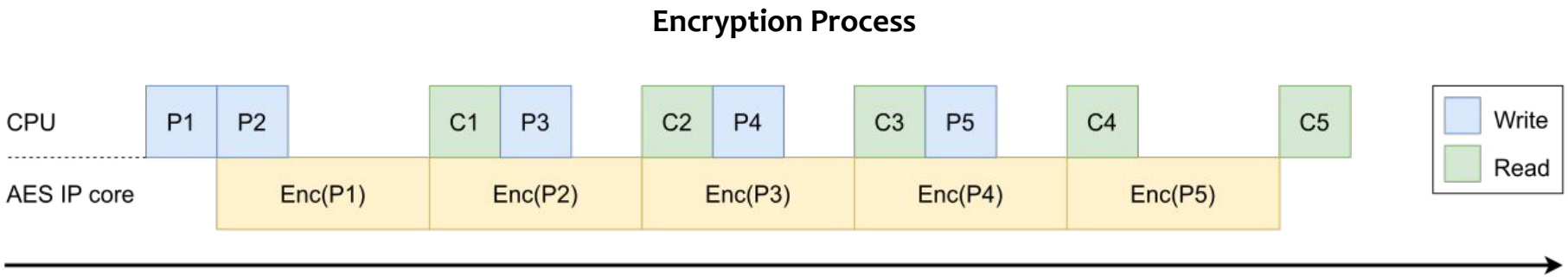
Evaluation: AES IP Core



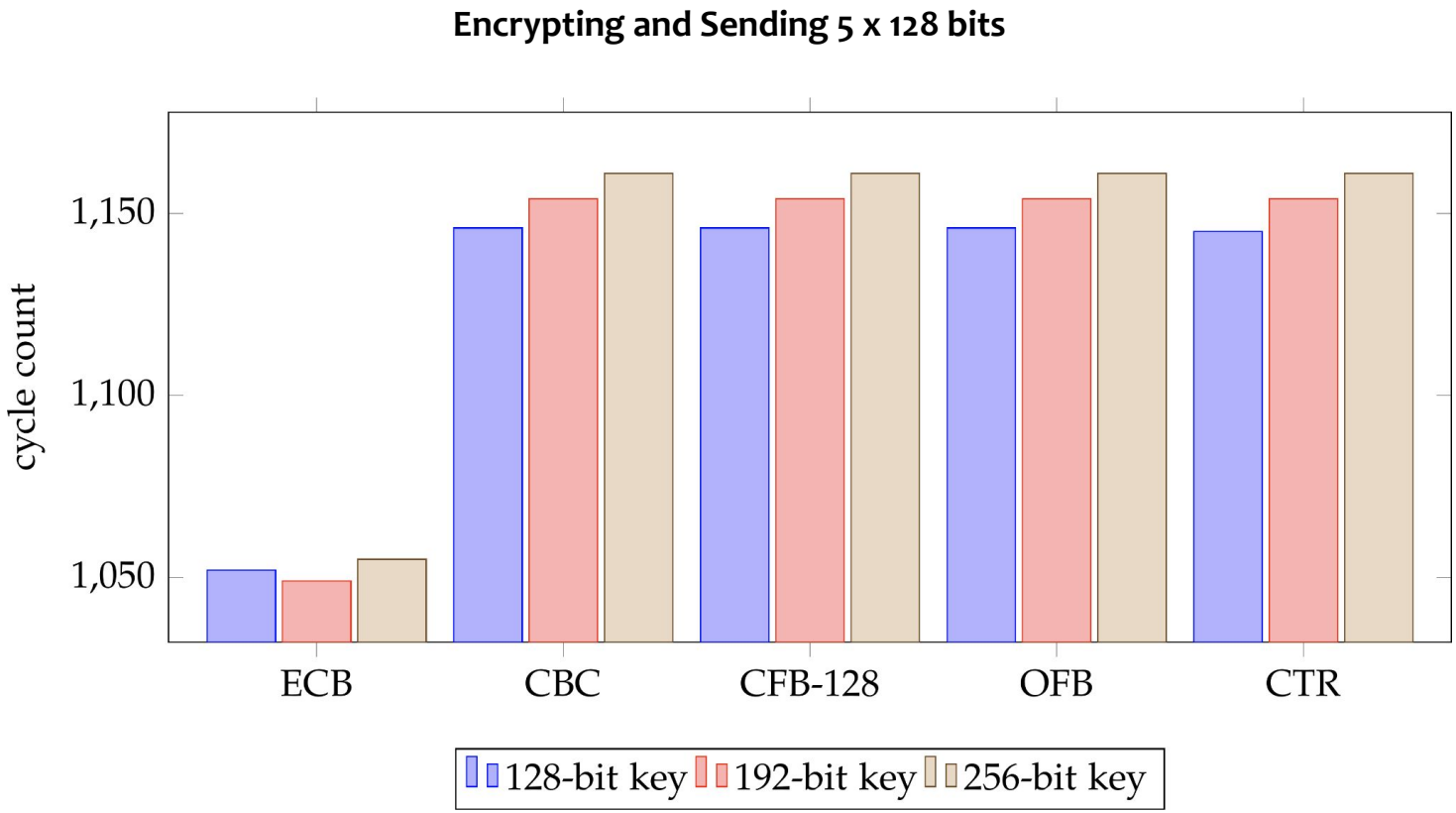
Evaluation: AES IP Core



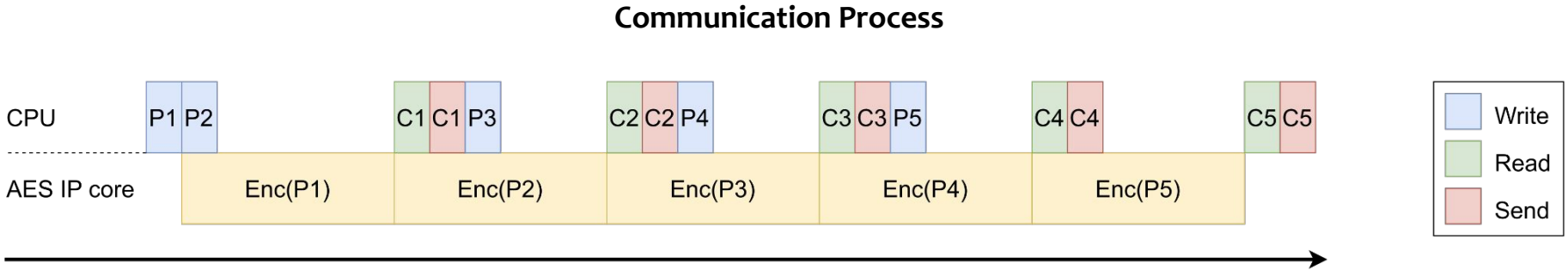
Evaluation: AES IP Core



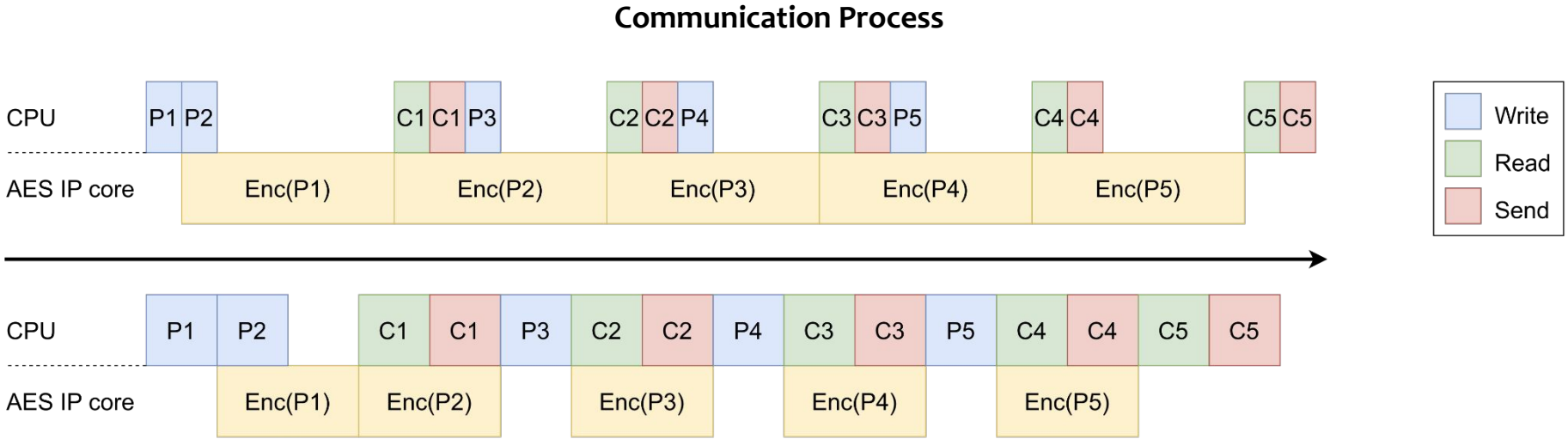
Evaluation: End-to-End Encrypted Communication



Evaluation: End-to-End Encrypted Communication

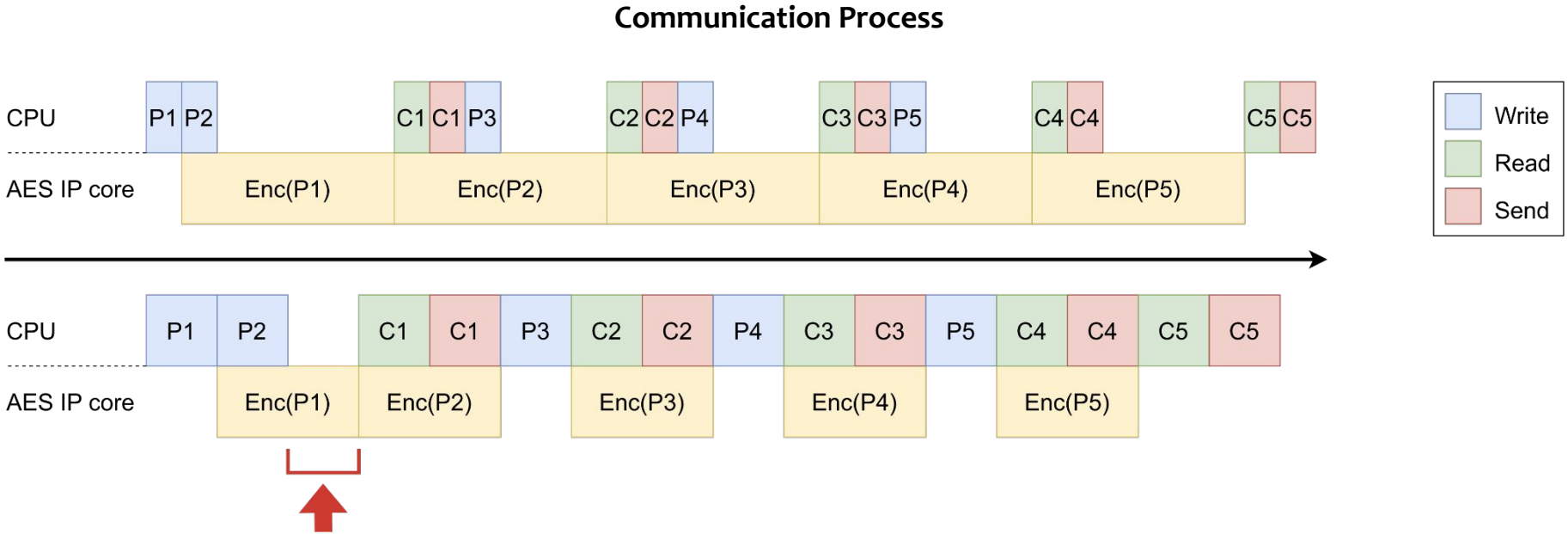


Evaluation: End-to-End Encrypted Communication



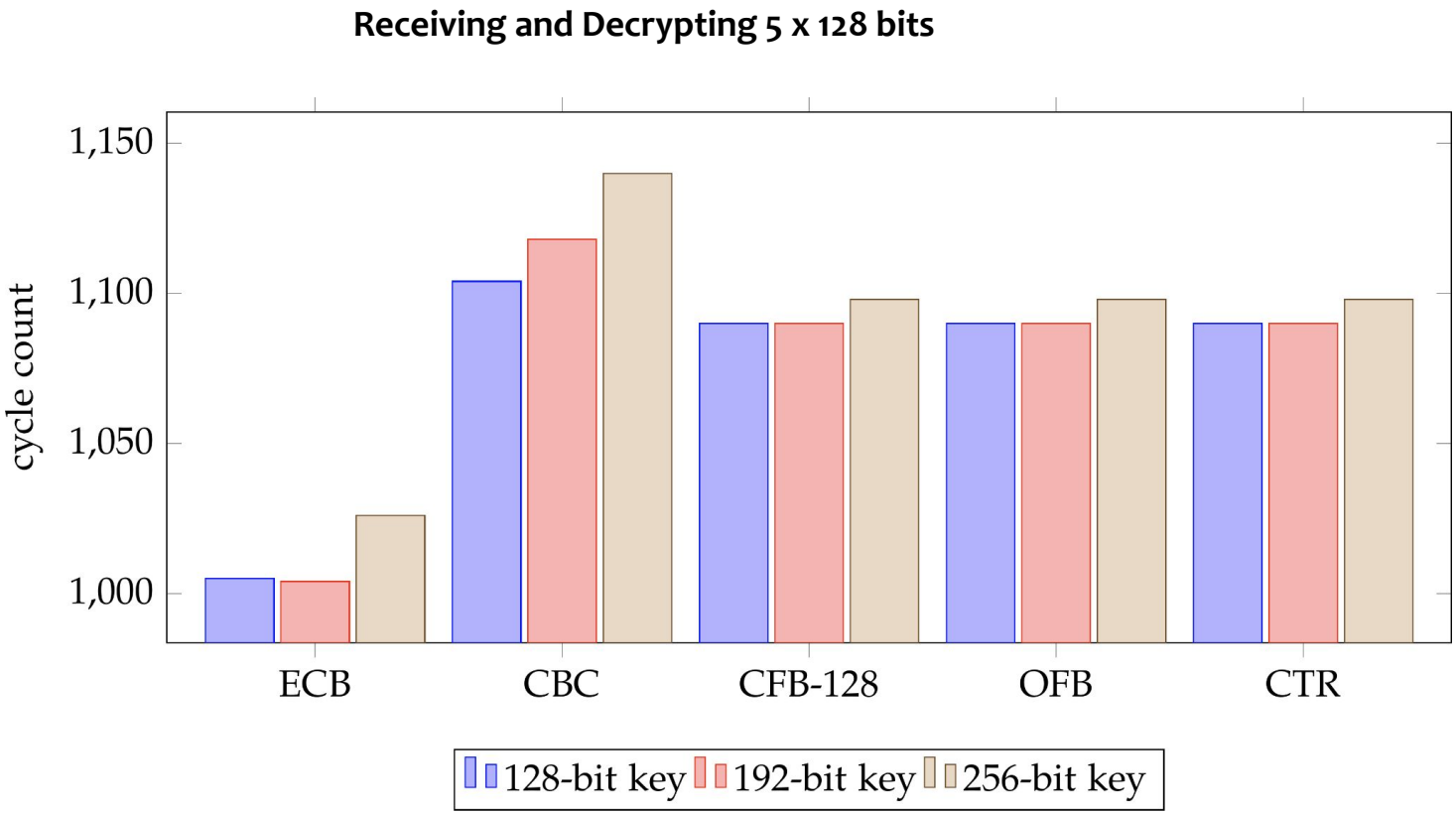
(top: dominant encryption; bottom: dominant read/write/send)

Evaluation: End-to-End Encrypted Communication



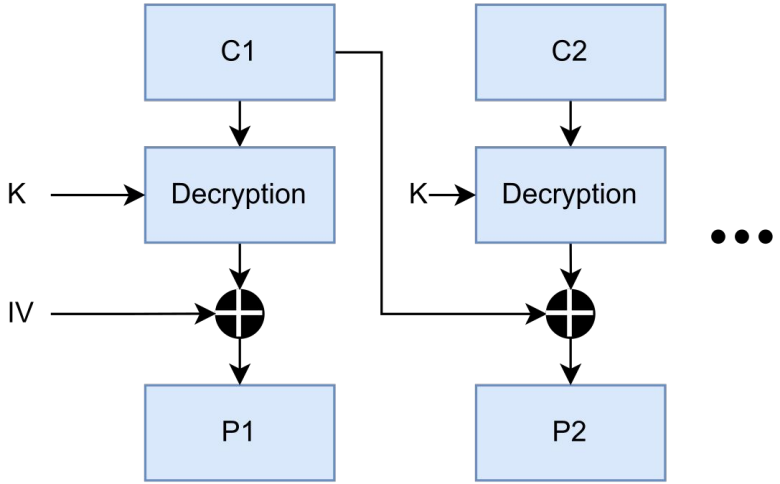
(top: dominant encryption; bottom: dominant read/write/send)

Evaluation: End-to-End Encrypted Communication

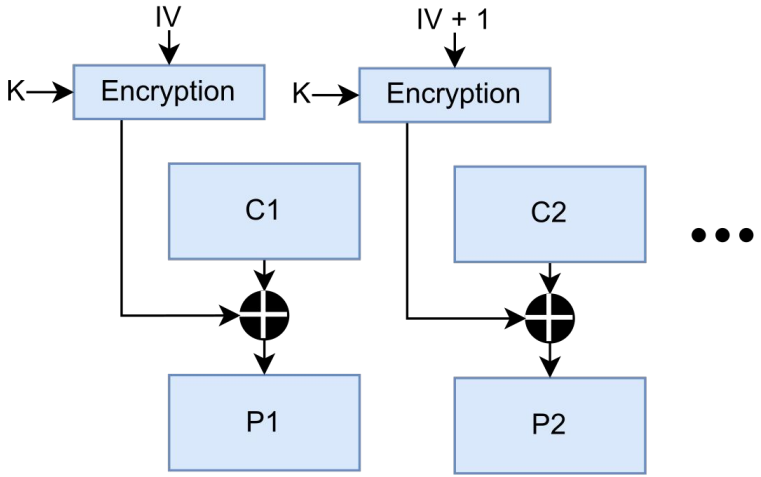


Evaluation: End-to-End Encrypted Communication

CBC Mode: Decryption



CTR Mode: Decryption

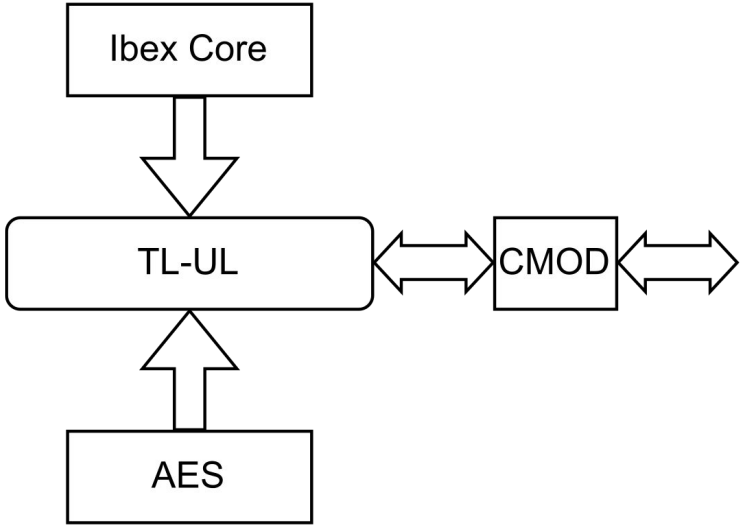


Conclusion

Read and writes are bottleneck for the encryption and sending process, and the reception and decryption process when using 128 bit and 192 bit keys.

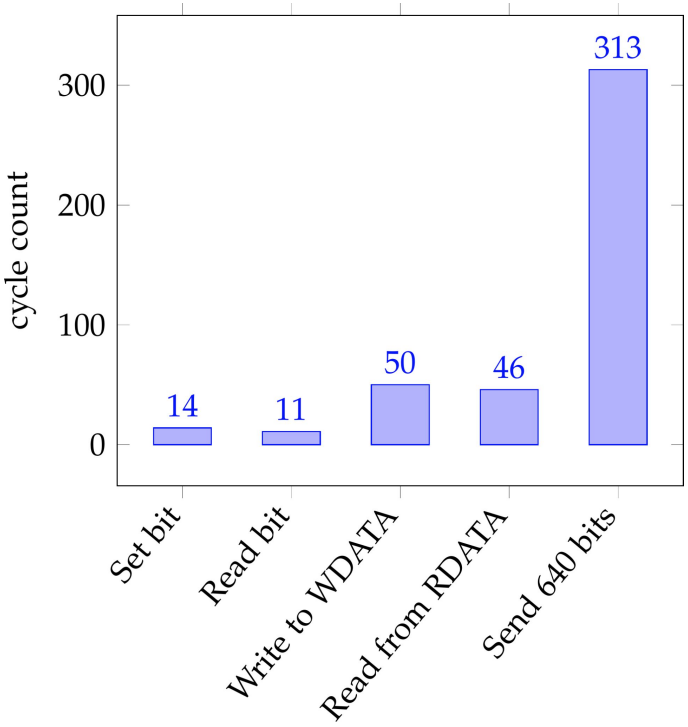
Future Work

- Add pipelining
- Add support for multiple CPUs



Backup

CMOD IP Core Measurements



Register Overview

Name	Offset	Length	Description
cmod.INTR_STATE	0x0	4	Interrupt State Register
cmod.INTR_ENABLE	0x4	4	Interrupt Enable Register
cmod.INTR_TEST	0x8	4	Interrupt Test Register
cmod.ALERT_TEST	0xc	4	Alert Test Register
cmod.CTRL	0x10	4	CMOD control register
cmod.STATUS	0x14	4	CMOD status register
cmod.WDATA_0	0x18	4	Data to be transmitted.
cmod.WDATA_1	0x1c	4	Data to be transmitted.
cmod.WDATA_2	0x20	4	Data to be transmitted.
cmod.WDATA_3	0x24	4	Data to be transmitted.
cmod.RDATA_0	0x28	4	If cmod.RXVALID is 1, data that was received.
cmod.RDATA_1	0x2c	4	If cmod.RXVALID is 1, data that was received.
cmod.RDATA_2	0x30	4	If cmod.RXVALID is 1, data that was received.
cmod.RDATA_3	0x34	4	If cmod.RXVALID is 1, data that was received.

Control Register

cmod.CTRL @ 0x10
Reset default = 0x0, mask 0x3f

Bits	Type	Reset	Name	Description
0	rw	0x0	TXTRIGGER	Starts the transmission of a new message.
1	rw	0x0	TXEND	Marks that the last data block of a message was written to the cmod.WDATA registers.
2	rw	0x0	RXCONFIRM	Confirms that the end of a message was noticed and allows the reception of a new message.
4:3	rw	0x0	TXILVL	Trigger level for tx_watermark interrupt. If the FIFO depth is smaller to the setting, it raises a tx_watermark interrupt. 0x0 txlvl2 2 data blocks 0x1 txlvl3 3 data blocks 0x2 txlvl4 4 data blocks 0x3 Reserved
6:5	rw	0x0	RXILVL	Trigger level for rx_watermark interrupt. If the FIFO depth is greater or equal to the setting, it raises a rx_watermark interrupt. 0x0 rxlvl1 1 data block 0x1 rxlvl2 2 data blocks 0x2 rxlvl3 3 data blocks 0x3 Reserved

Status Register

cmod.STATUS @ 0x14 Reset default = 0x0, mask 0xfff				
Bits	**Type**	**Reset**	**Name**	**Description**
0	ro	x	TX	The CMOD unit is idle (0) or busy(1). This bit is '0' if no message is currently transmitted. This bit is '1' one of the following is the case: i) The transmission has started (TXTRIGGER set), and the last data block of the message was not yet received (TXEND was not set yet), ii) The last data block of a message was received but still needs to leave the outgoing buffer.
1	ro	x	TXFULL	The buffer for outgoing data is full.
2	ro	x	RXFULL	The buffer for incoming data is full.
3	ro	x	TXEMPTY	The buffer for outgoing data is empty.
4	ro	x	TXINPUT_READY	The CMOD unit is ready (1) or not ready (0) to receive new input via the cmod.WDATA registers. If this bit is '0', it means that either the transmission of a new message was not started yet or that the buffer for outgoing data is full.
5	ro	x	RXVALID	The CMOD unit has no valid output (0) or valid output data.
8:6	ro	x	TXLVL	The current fill level of the buffer for outgoing data.
11:9	ro	x	RXLVL	The current fill level of the buffer for incoming data.
12	ro	x	RXLAST	If set, it means that the last data block of a message was already read. To allow the reception of a new message, the cmod.RXCONFIRM bit needs to be set.