From Datasheets to Digital Logic

synthesizing an FPGA SPI slave “from the gates”

Joshua Vasquez
March 26, 2015

The Road Map

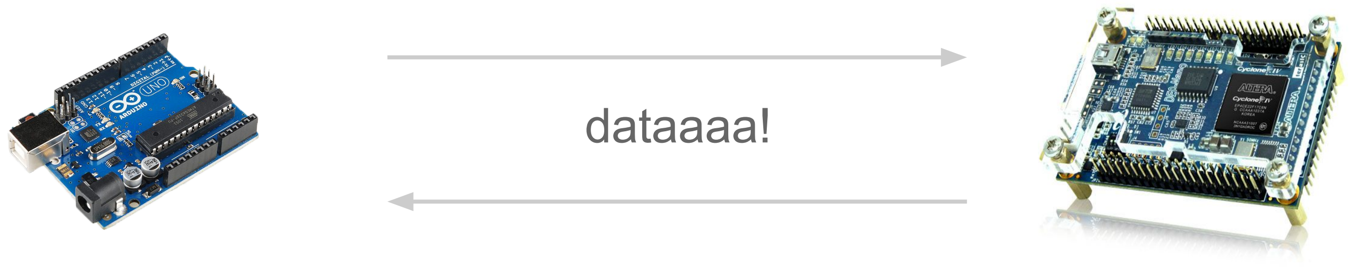
- Top-Level Goal
    - Motivation
- What is SPI?
    - SPI Topology
    - SPI Wiring
    - SPI Protocol*
- Defining a Protocol
    - Inspired by MEMs Sensors
- Example Reading Encoders
- Building up the SPI Communication Hardware
- Building up the Controller Hardware
- Wrap-up

The Top-Level Goal

- Develop a protocol for both sending and receiving data between a microcontroller and an FPGA

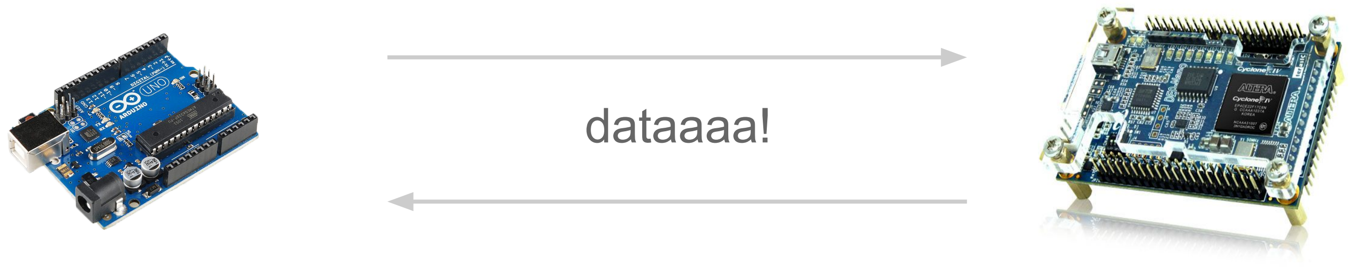
The Top-Level Goal

- Develop a protocol for both sending and receiving data between a microcontroller and an FPGA



The Top-Level Goal

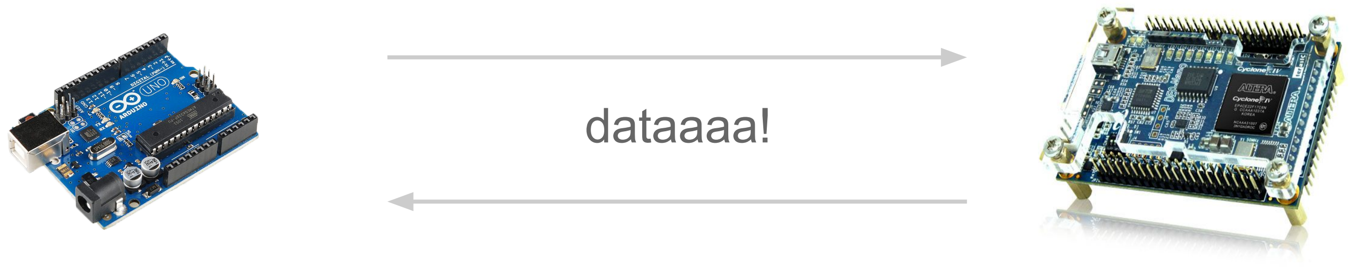
- Develop a protocol for both sending and receiving data between a microcontroller and an FPGA



Why?

The Top-Level Goal

- Develop a protocol for both sending and receiving data between a microcontroller and an FPGA



Why?

- read encoders
- drive (lots) of hobby servos
- turn the FPGA into a generic command-accepting slave
- etc..

SPI

- What is it?

SPI

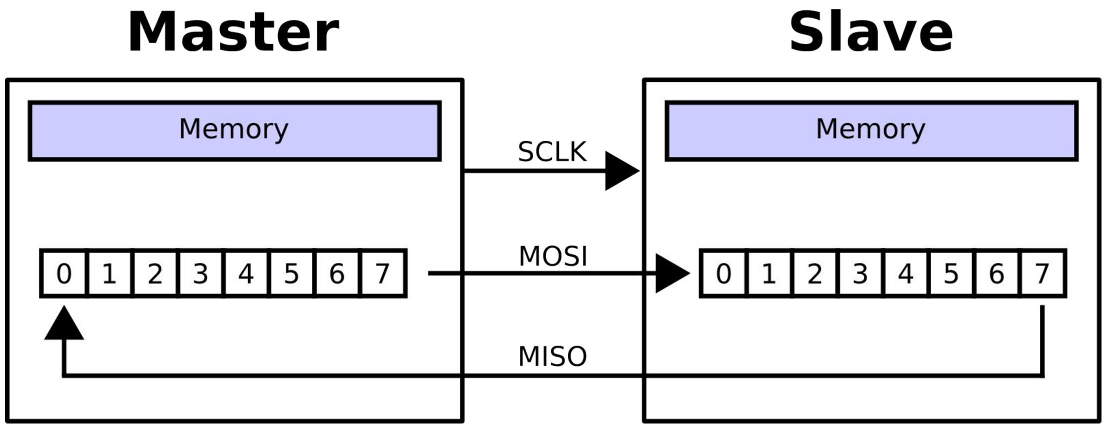
- What is it?
    - Method of synchronous serial communication

SPI

- What does it look like?

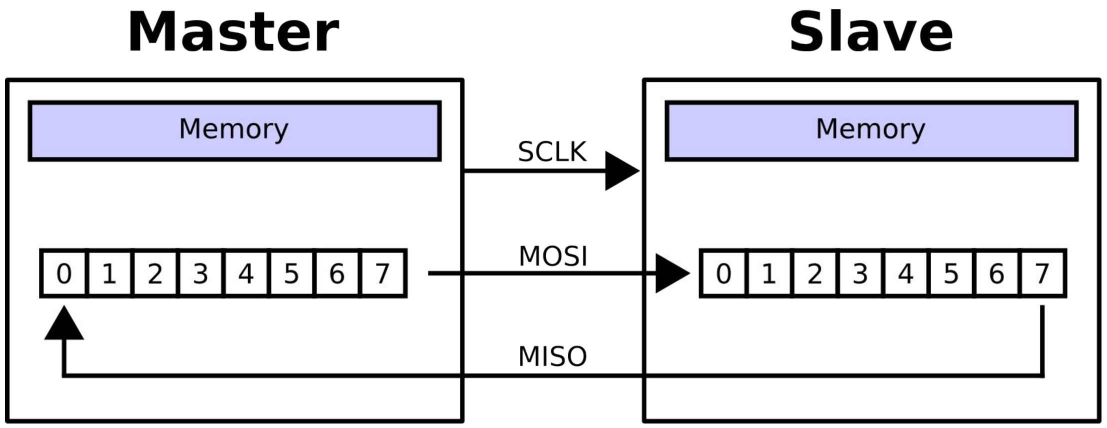
SPI Topology

- What does SPI look like?



SPI Topology

- What does SPI look like?

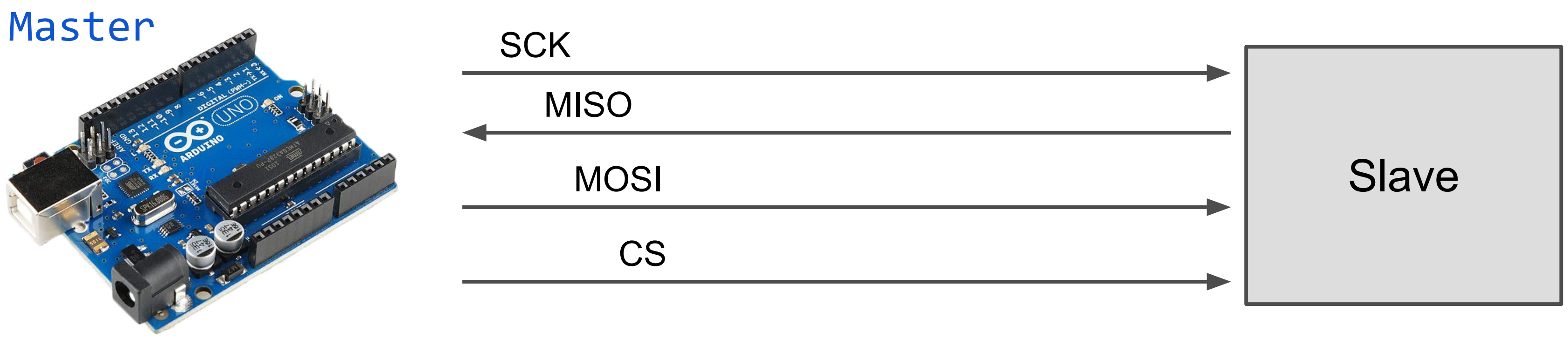


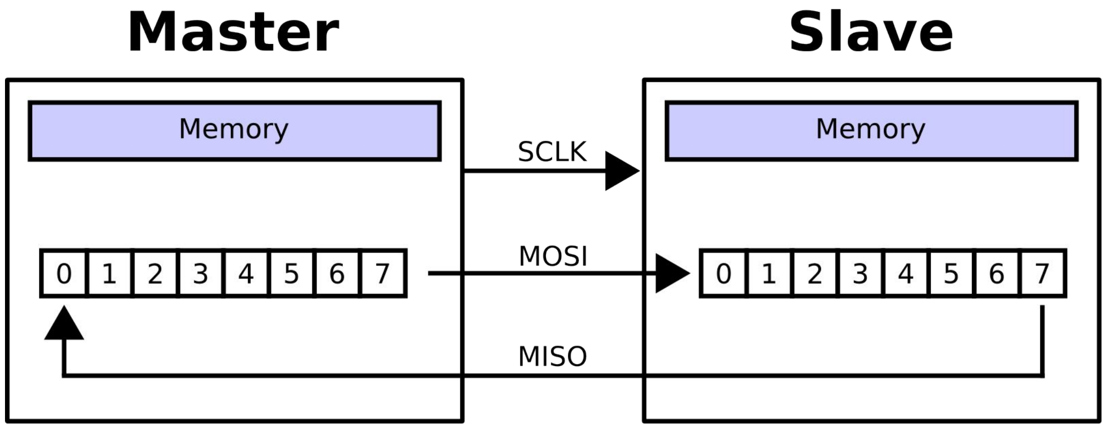
- Bidirectional data transfer synchronized with a clock.
- Frame (in this case) is 8-bits per transfer
- Multiple slaves supported

SPI Topology (contd)

Bidirectional Data Transfer

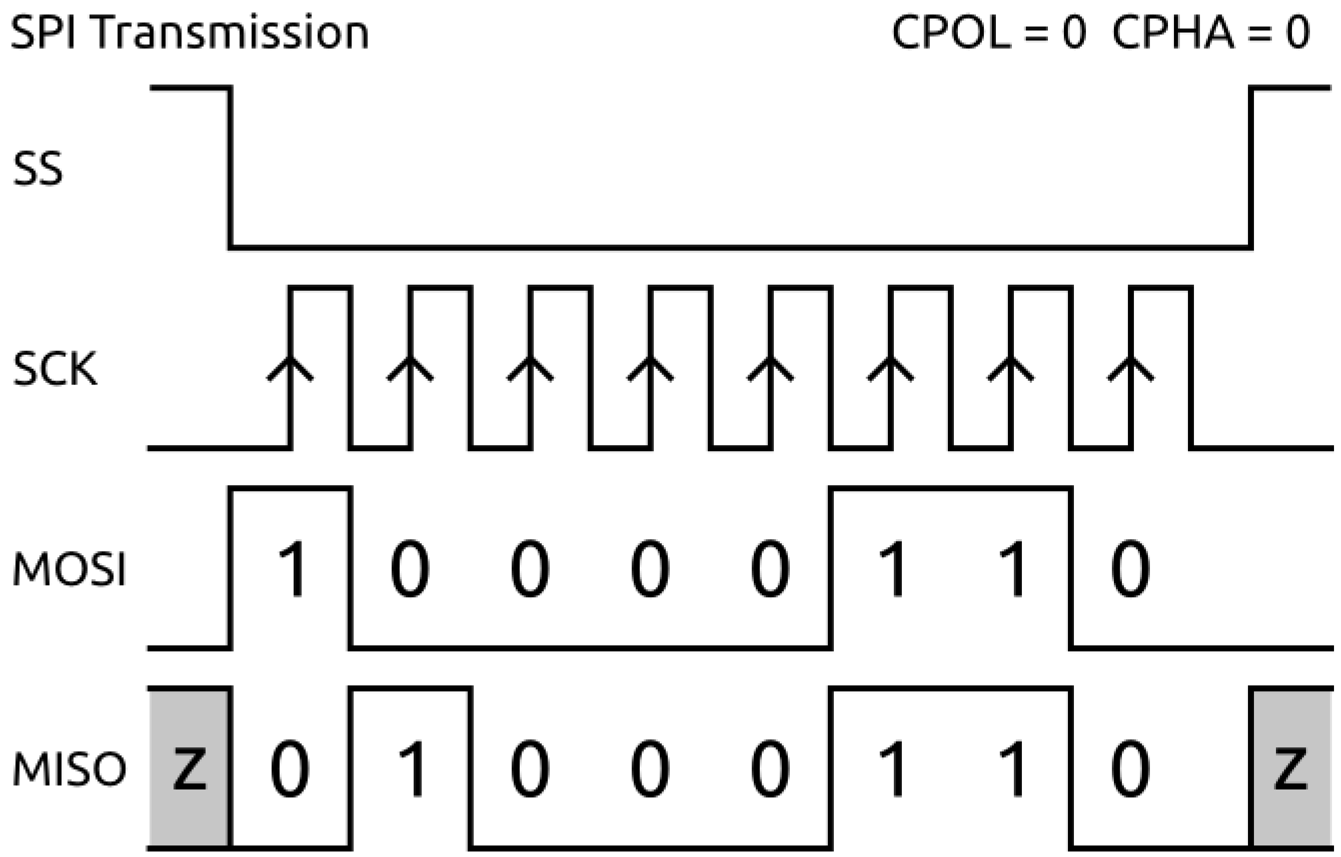
via MISO and MOSI





SPI Topology (contd)

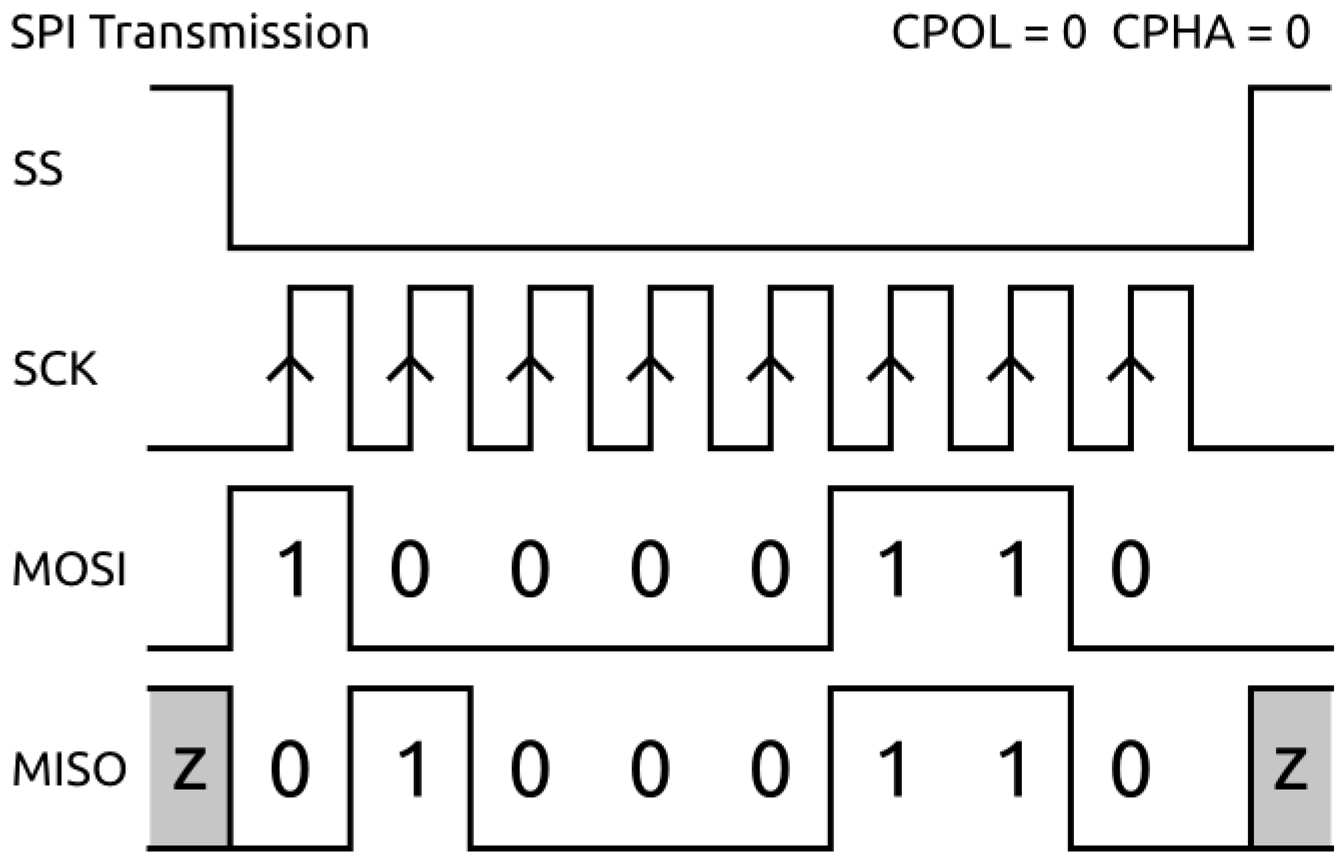
A Single 8-bit transfer



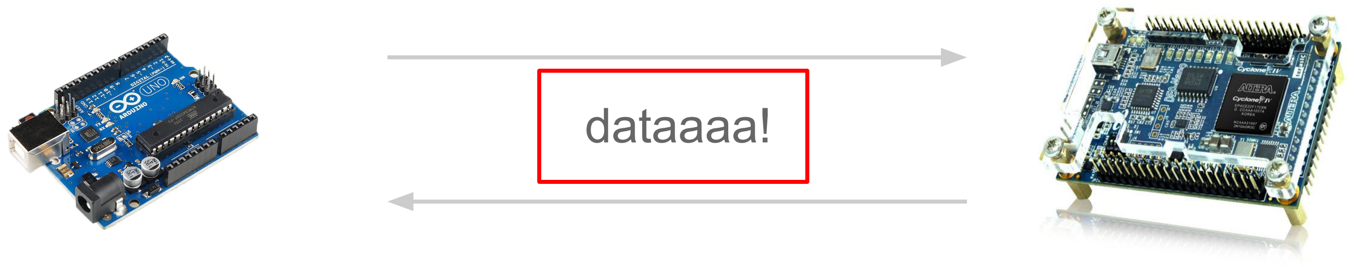
SPI Topology (contd)

NOTE: Protocol is undefined.

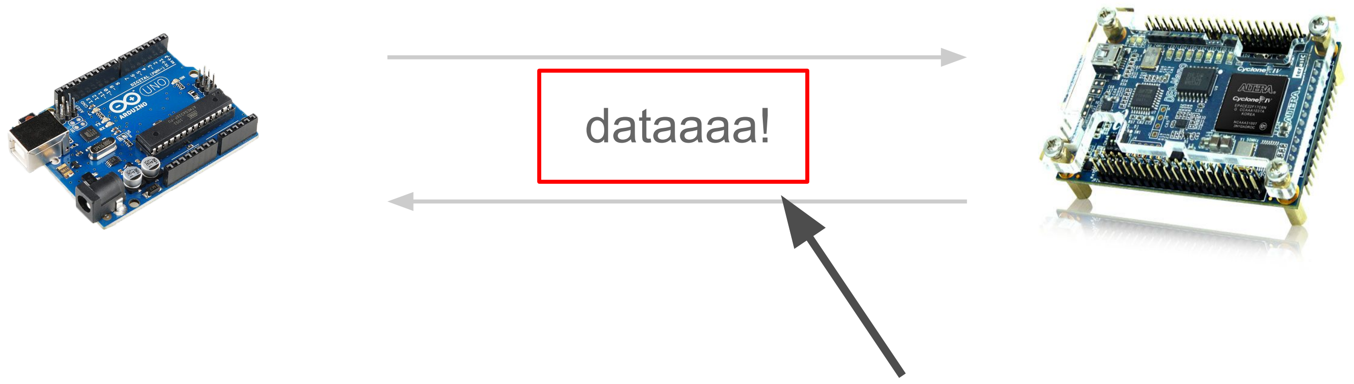
A Single 8-bit transfer



Defining a Protocol



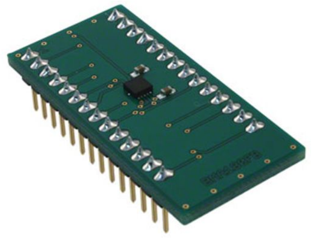
Defining a Protocol



We need to agree here

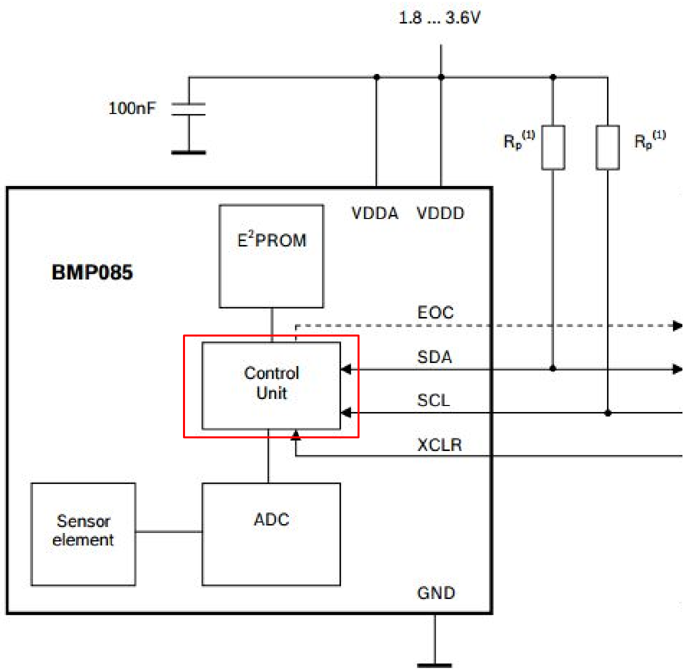
Protocol Inspiration:

- MEMs Sensor Interface
  - All the same communication interface!



Protocol Inspiration: MEMs

- Reading and Writing to Registers inside of the Chip's Internal memory
  - READ: get data from the chip
  - WRITE: tweak settings on the chip



Example Chip:
Pressure Sensor
with i2c interface

Protocol Inspiration: MEMs

Read/Write   Address   Data0, Data1,… DataN

Read/Write Address Data0

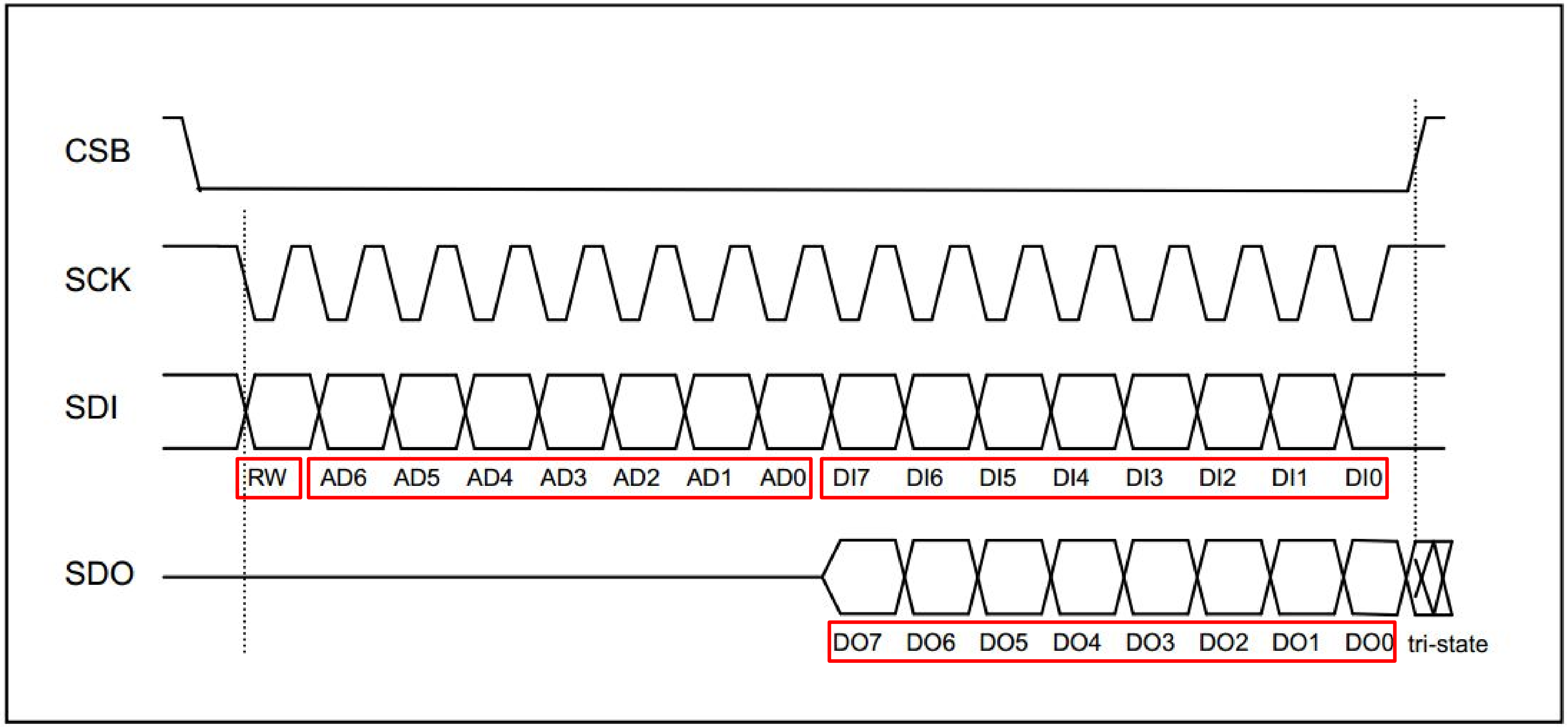
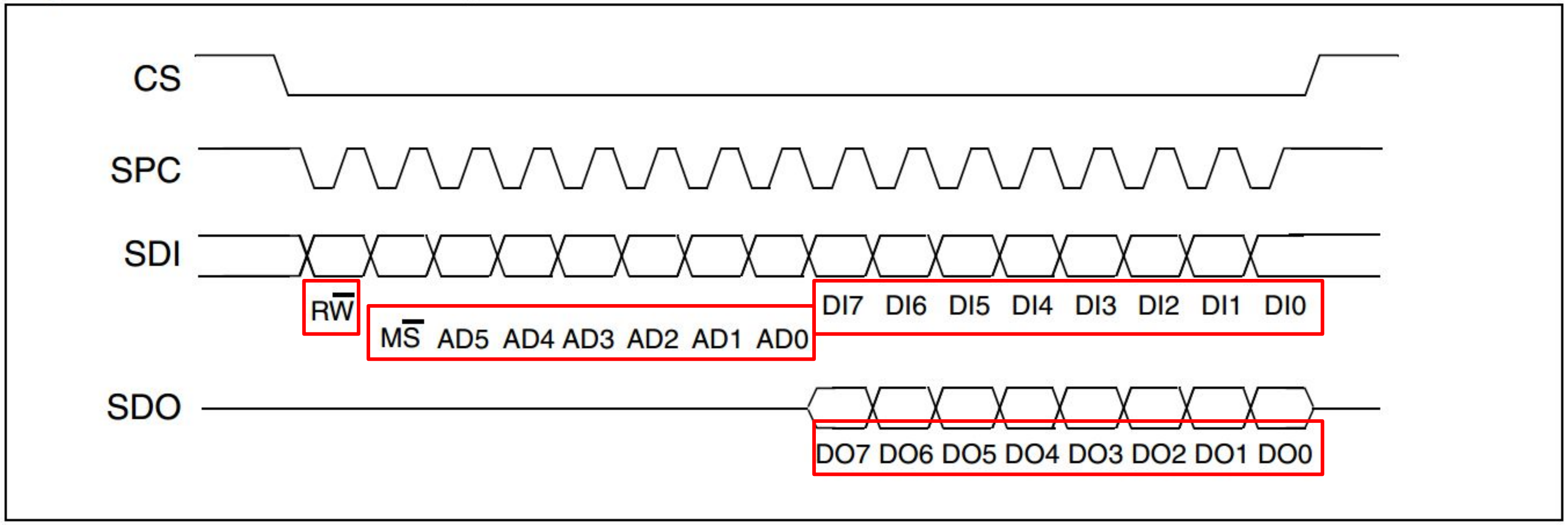


Figure 17: 4-wire SPI sequence

Bosch MEMs Sensors

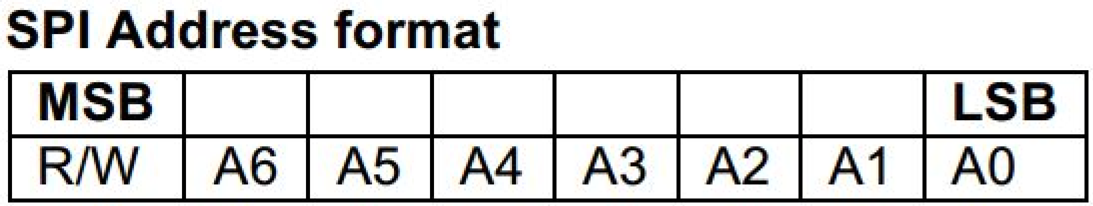
Read/Write Address Data0

Figure 6. Read & write protocol


CS
SPC
SDI
SDO
R$\overline{W}$
M$\overline{S}$ AD5 AD4 AD3 AD2 AD1 AD0
DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0

Invensense Sensors

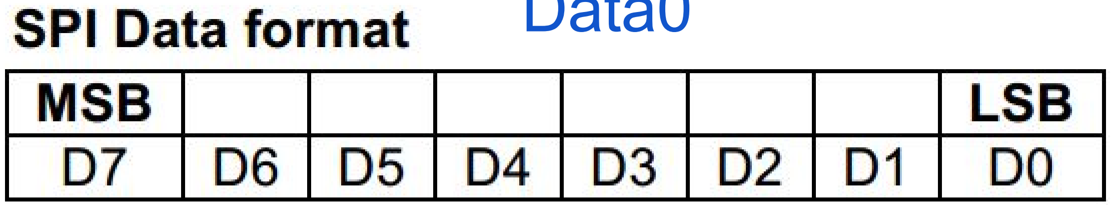
Read/Write Address



SPI Address format

MSB							LSB
R/W	A6	A5	A4	A3	A2	A1	A0

Data0



SPI Data format

MSB							LSB
D7	D6	D5	D4	D3	D2	D1	D0

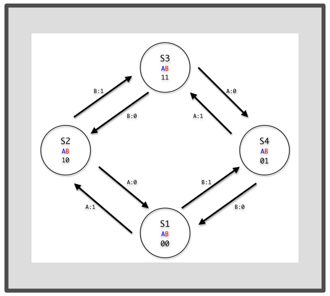
ST Sensors

The Road Map

- Top-Level Goal
    - Motivation
- What is SPI?
    - SPI Topology
    - SPI Wiring
    - SPI Protocol*
- Defining a Protocol
    - Inspired by MEMs Sensors
- Example Reading Encoders
- Building up the SPI Communication Hardware
- Building up the Controller Hardware
- Wrap-up

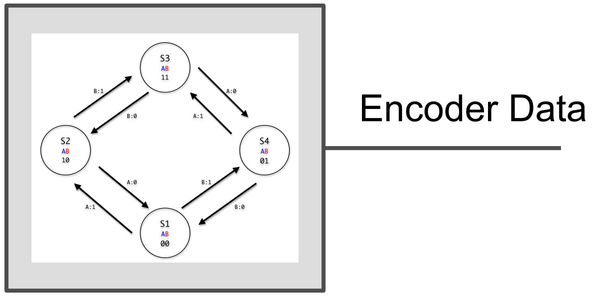
Application: reading encoders

Application: reading encoders

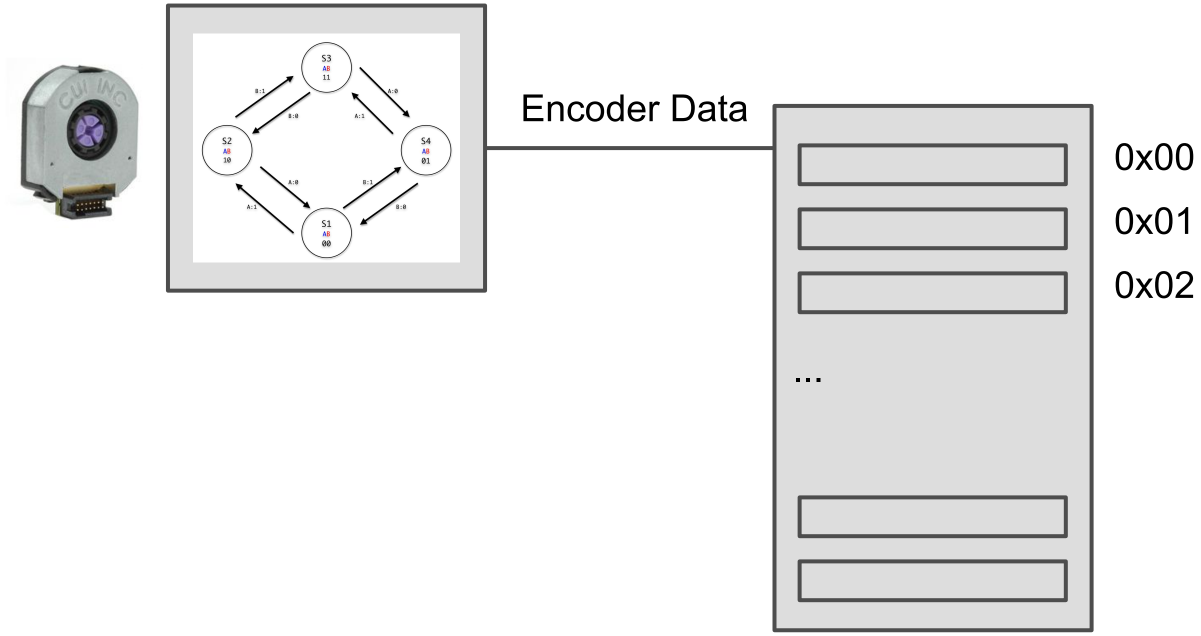


Application: reading encoders

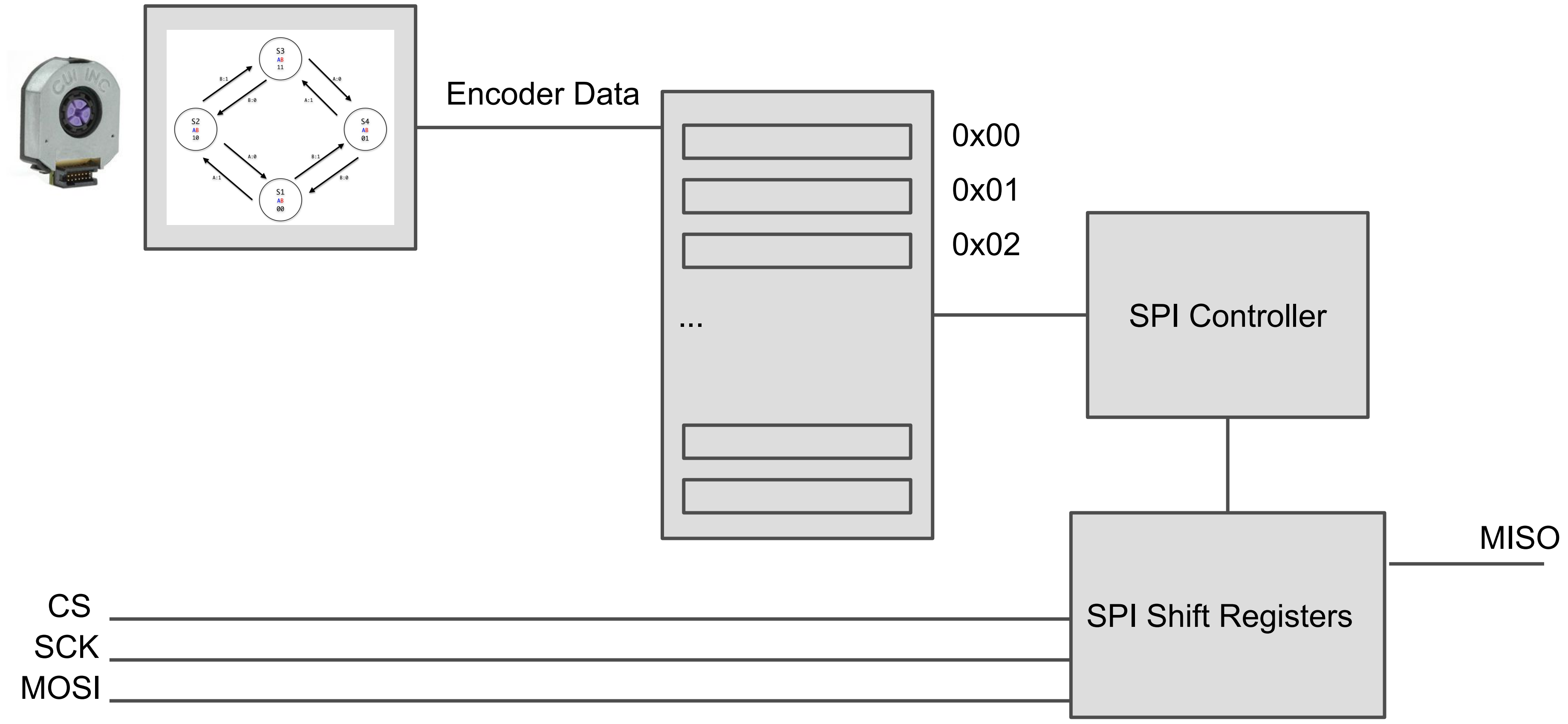
Rotary Encoder Example (Reading)



Application: reading encoders

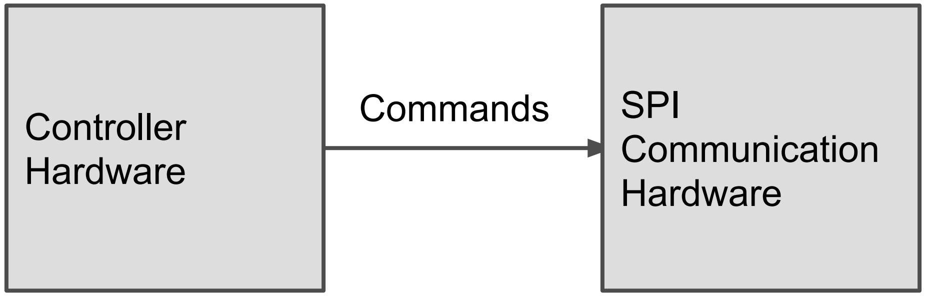


Application: reading encoders



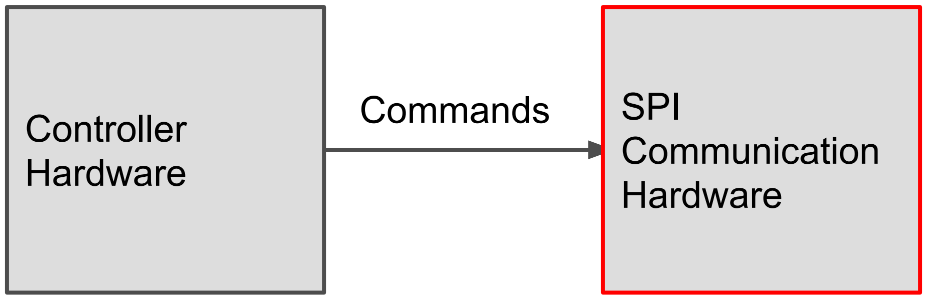
Building it up From Gates

Two Parts

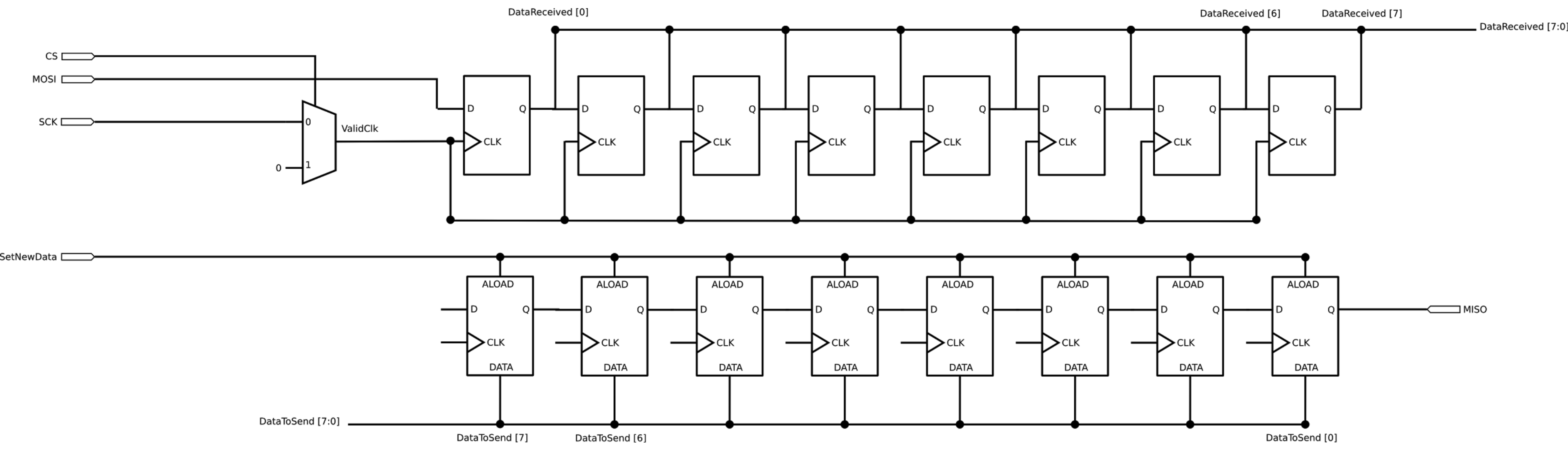


Building it up From Gates

Two Parts

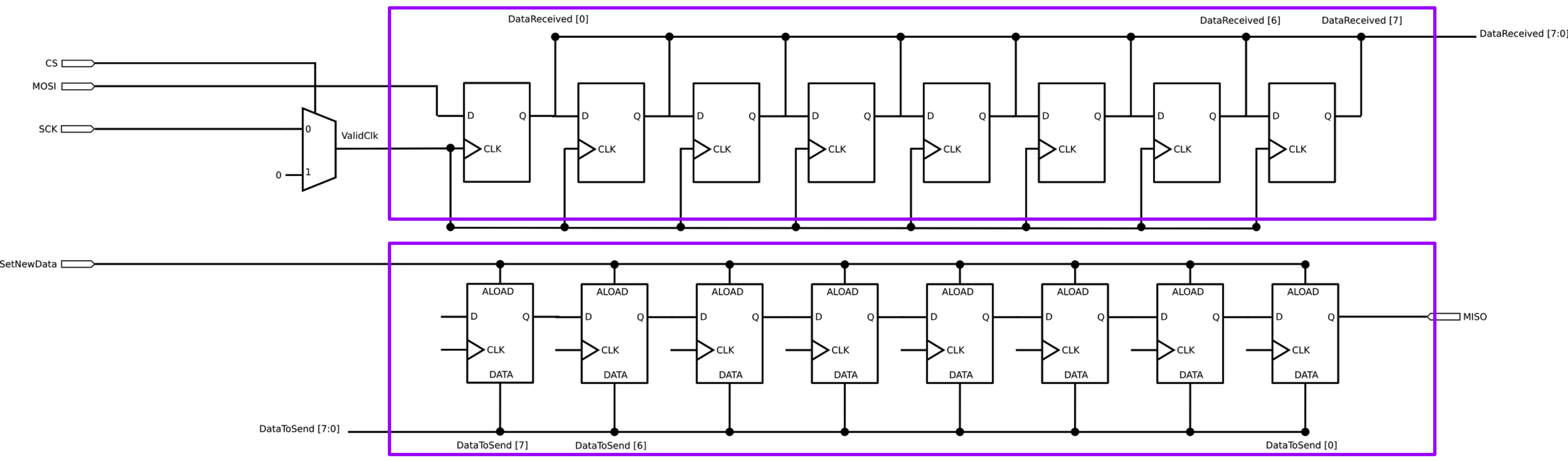


SPI Communication Hardware



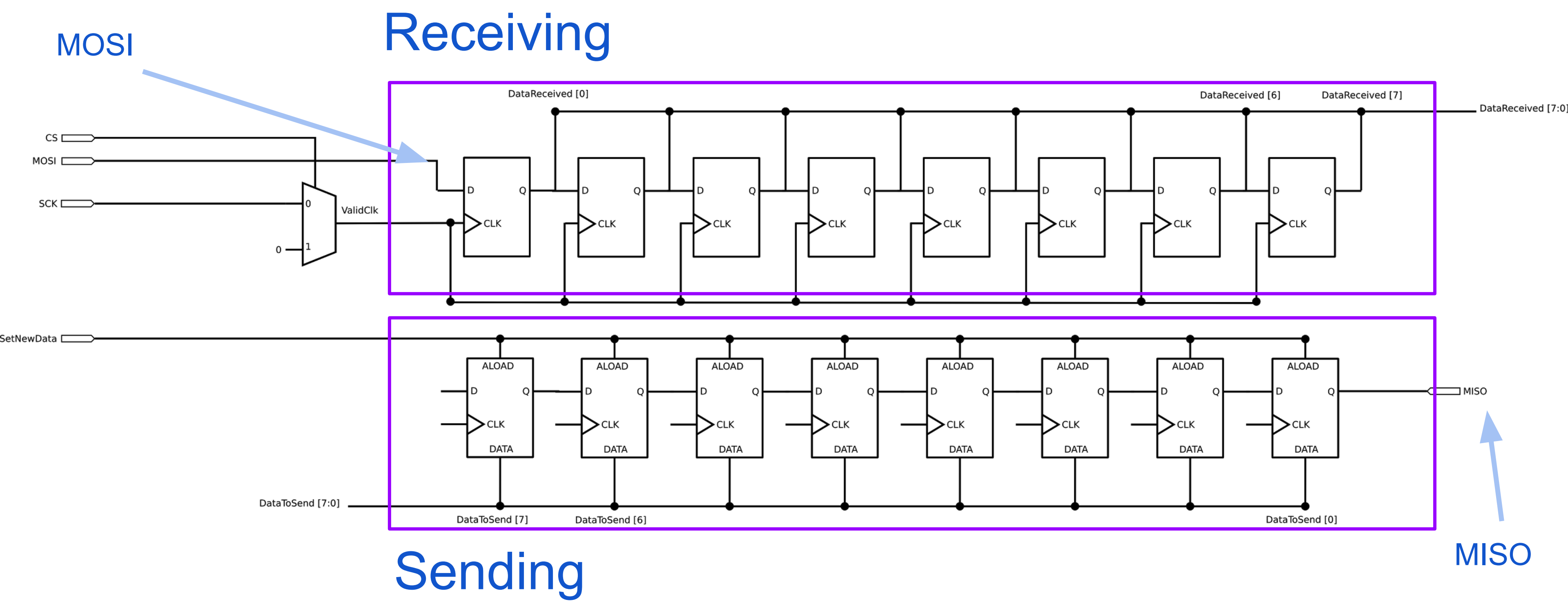
Two 8-bit Buffers

Receiving

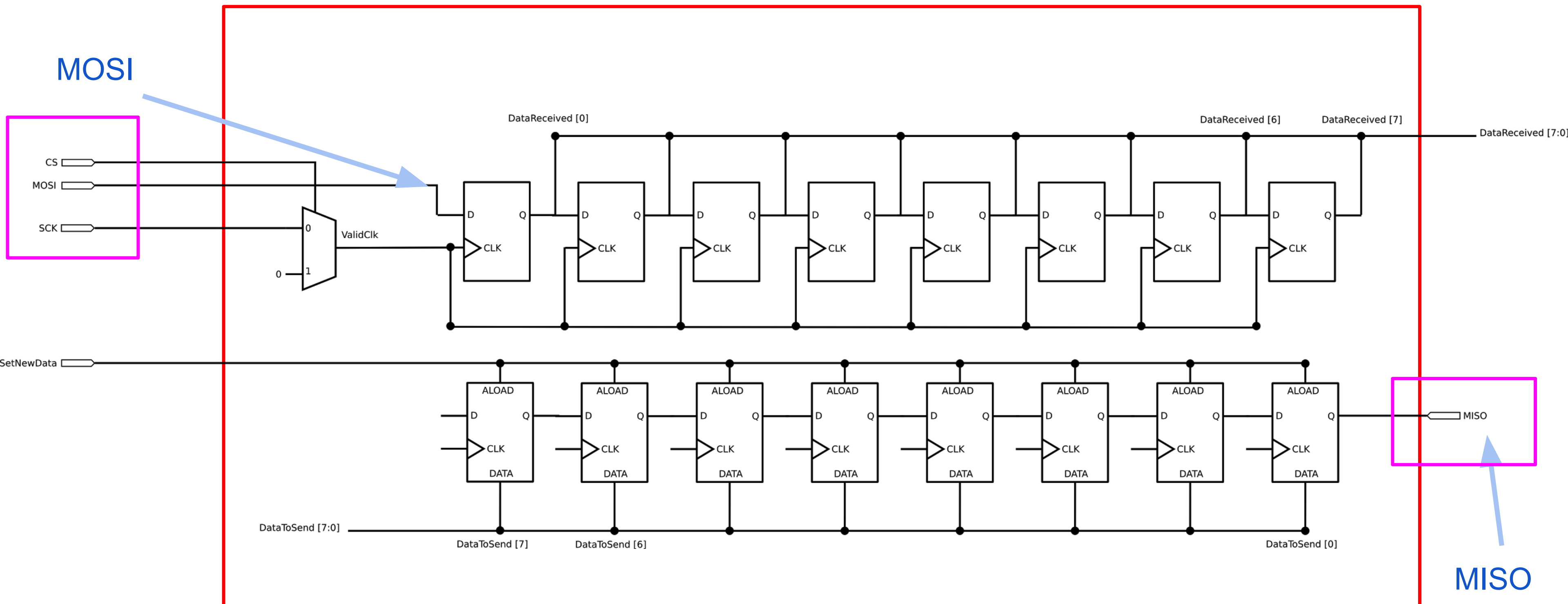


Sending

Two 8-bit Buffers

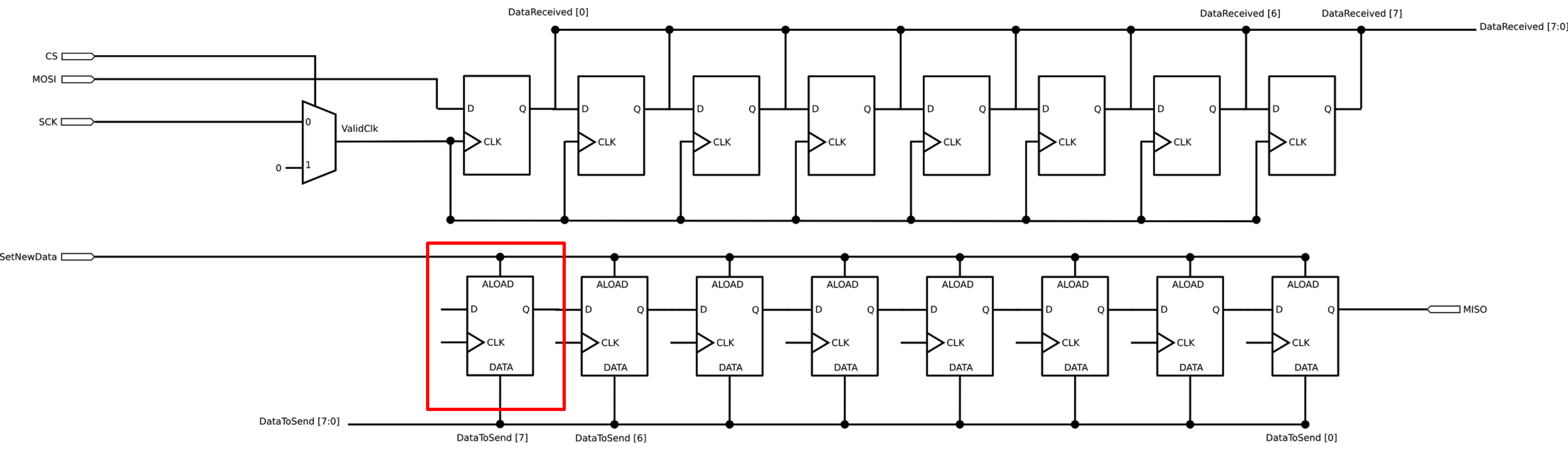


Two 8-bit Buffers



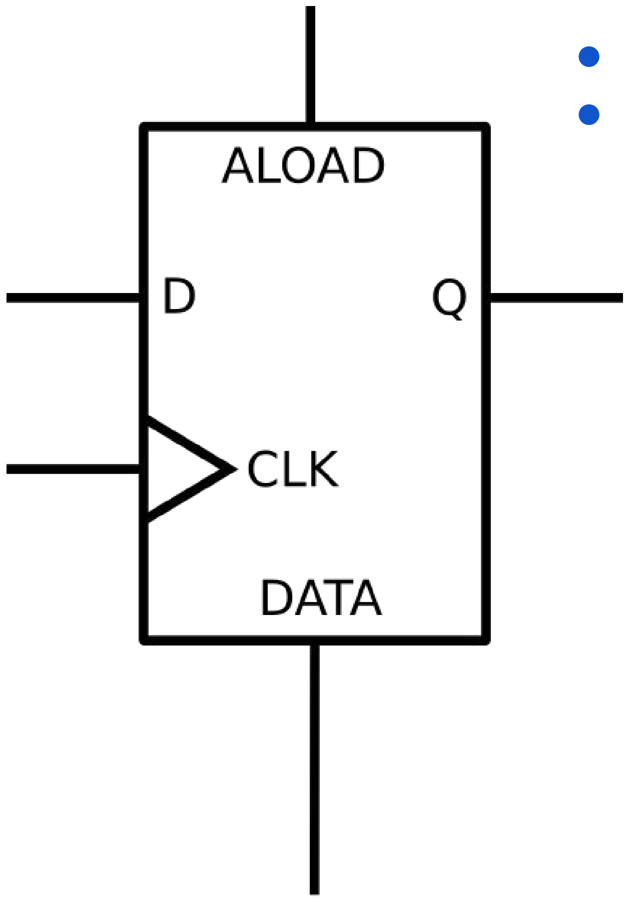
Note internal vs external logic

Diving into the details



Sending Data

D-Flip Flop with Asynchronous Load



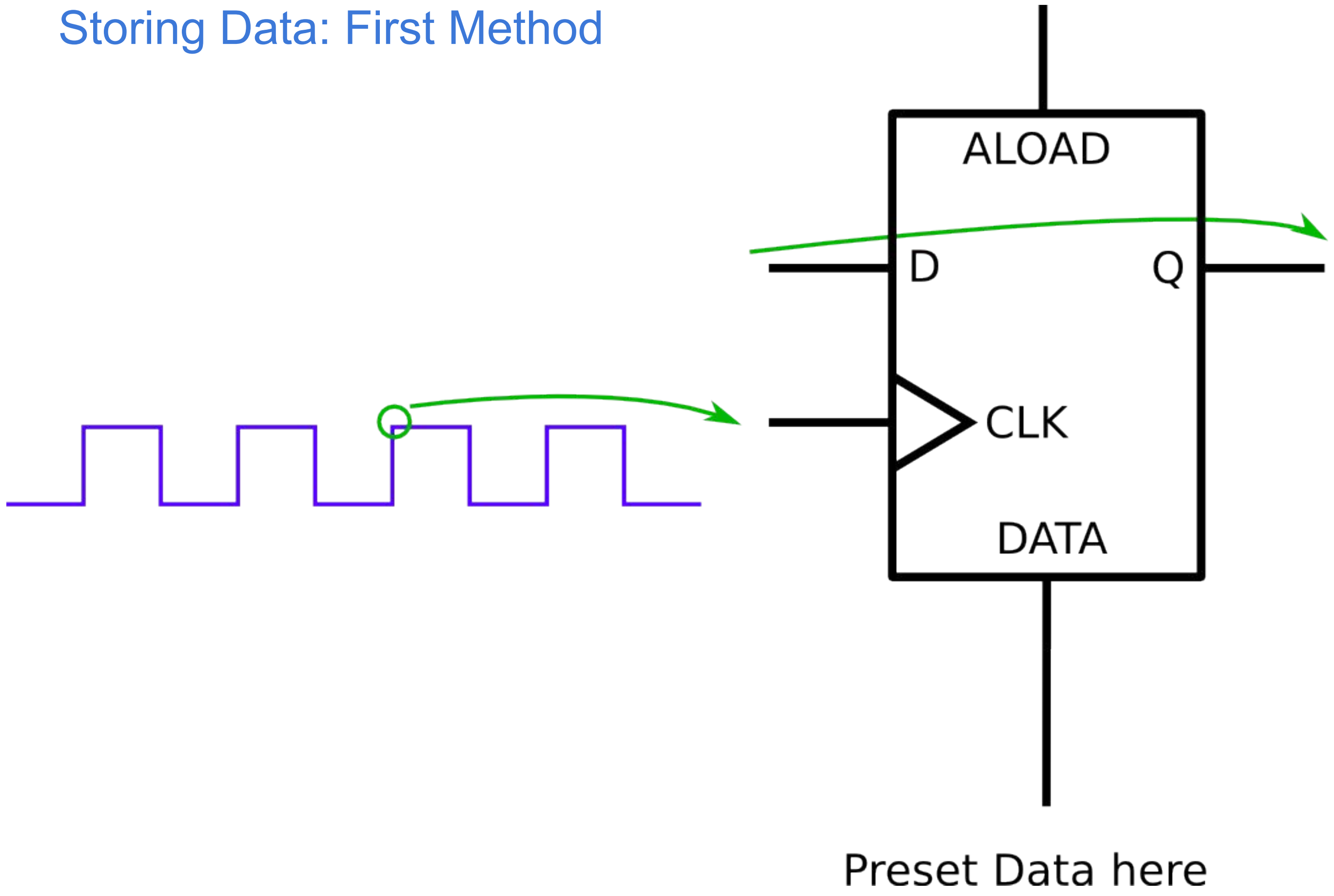
Preset Data here

- Stores 1 bit
- Can be pre-loaded with data at any moment in time

Sending Data

- Data gets passed from D to Q (aka: stored) on rising edge of clock signal

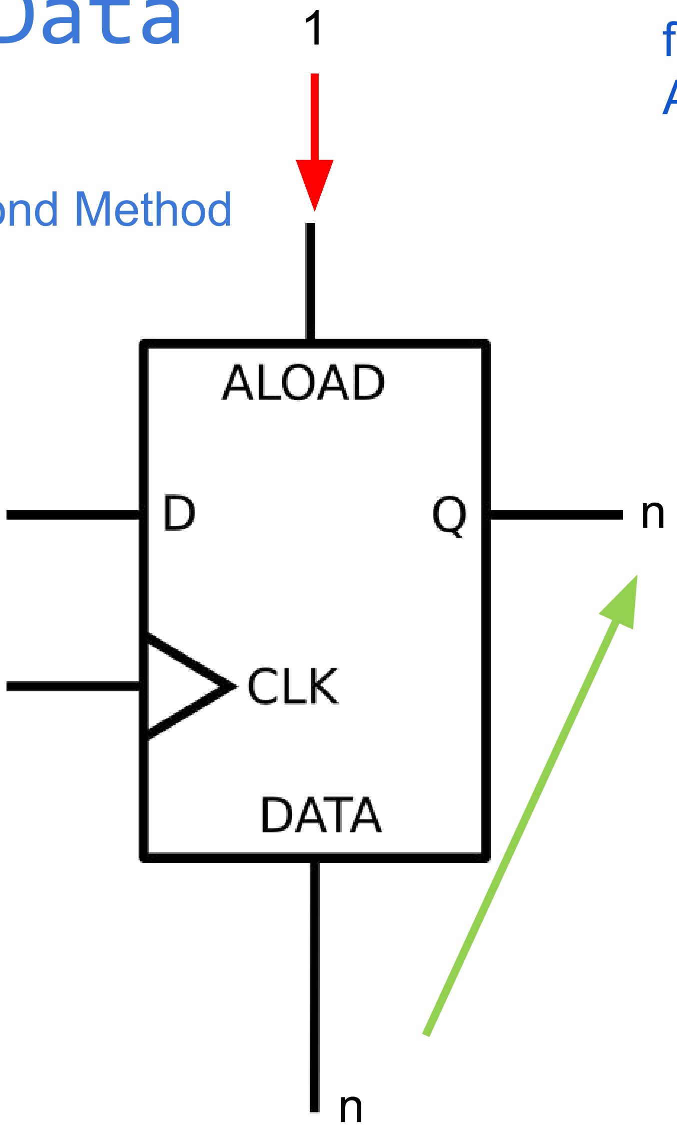
Storing Data: First Method



Sending Data

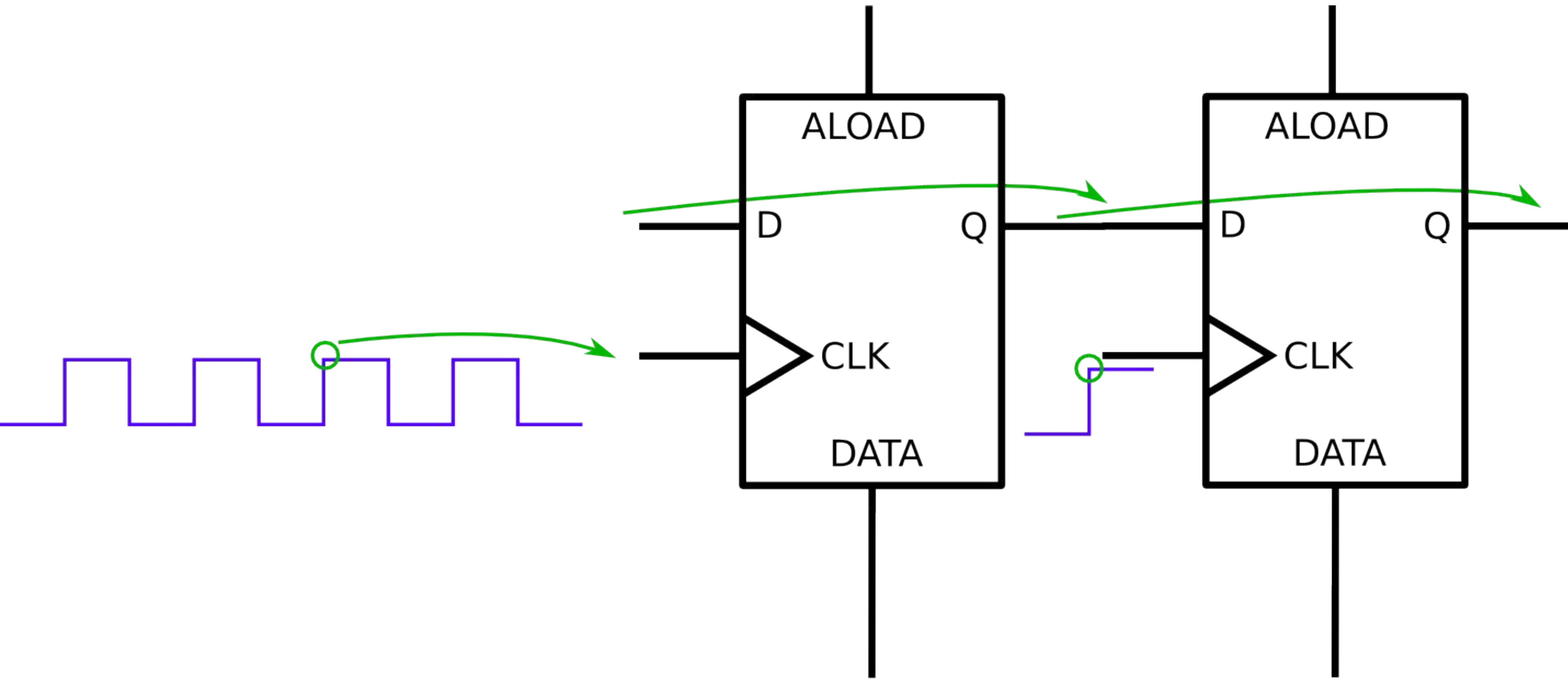
Storing Data: Second Method

- Data gets loaded (stored) from DATA to Q when ALOAD is asserted



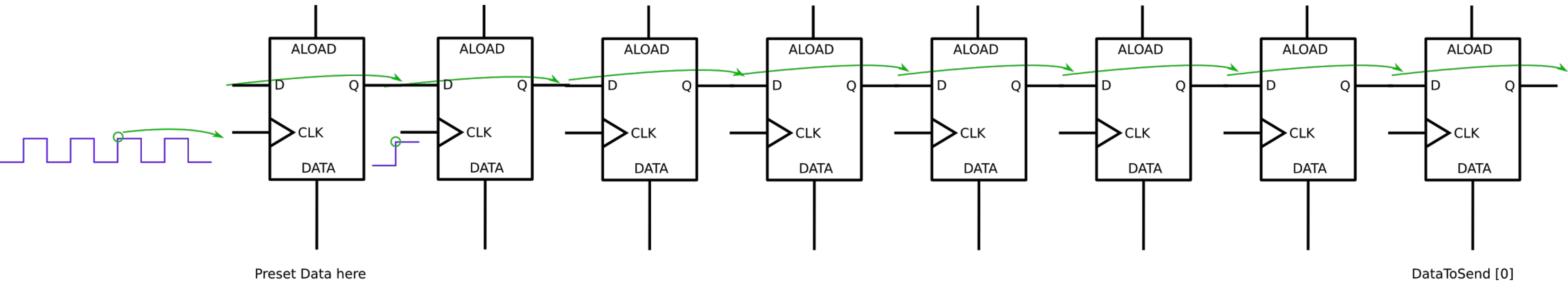
Preset Data here

Sending Data



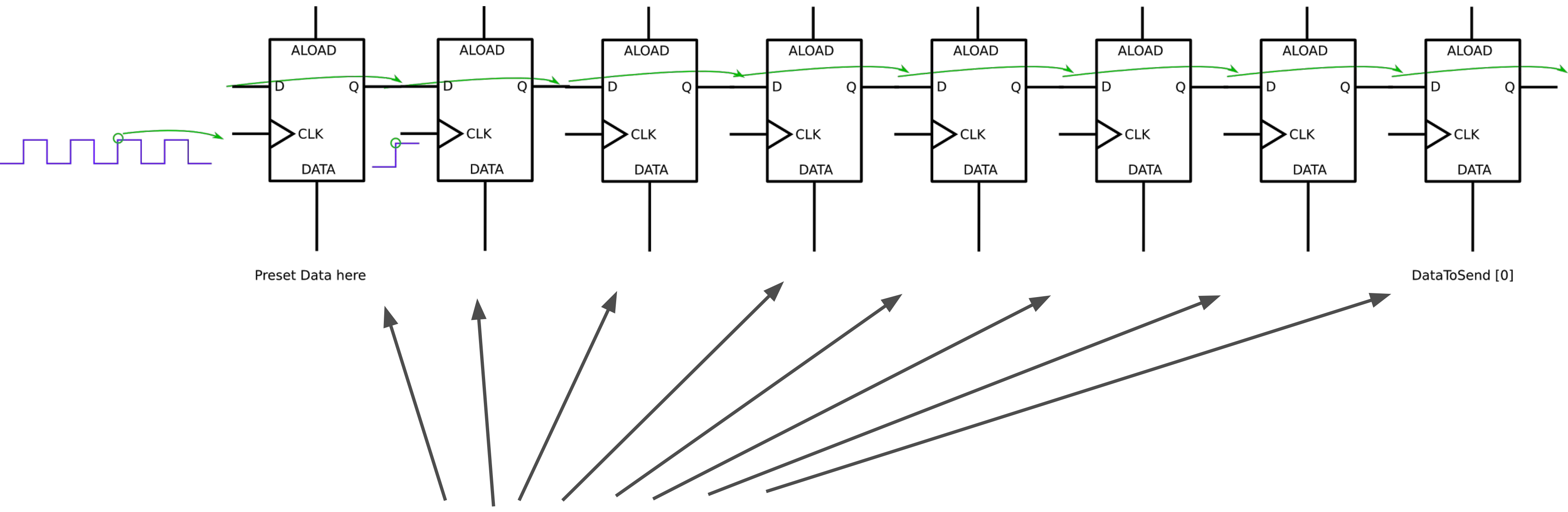
Preset Data here

Sending Data



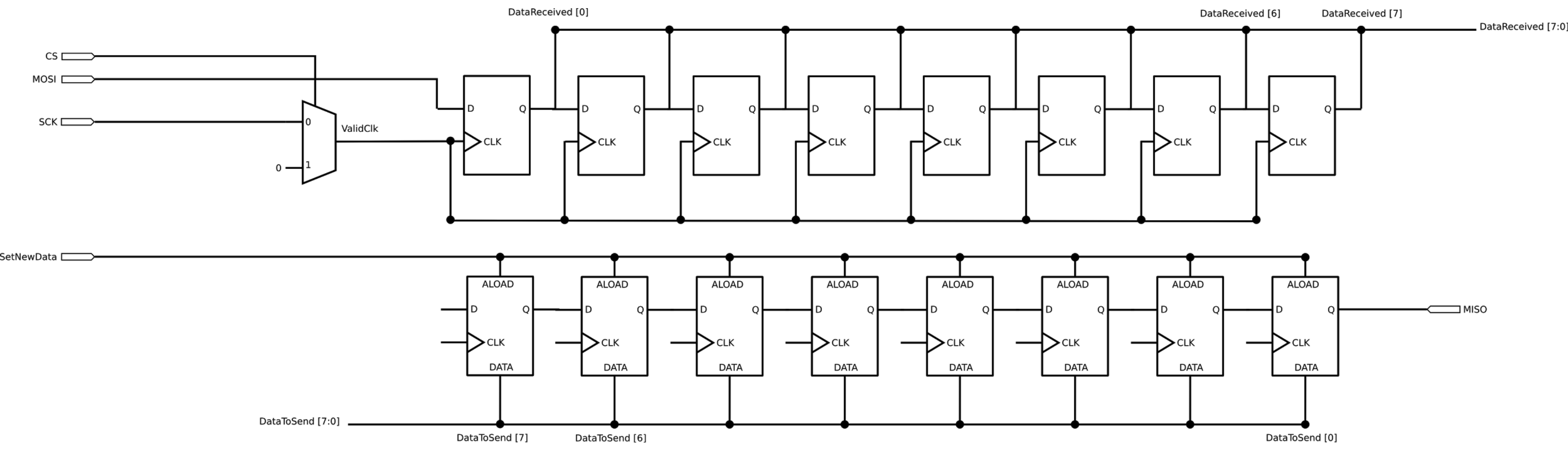
8-bit Shift Register!

Sending Data

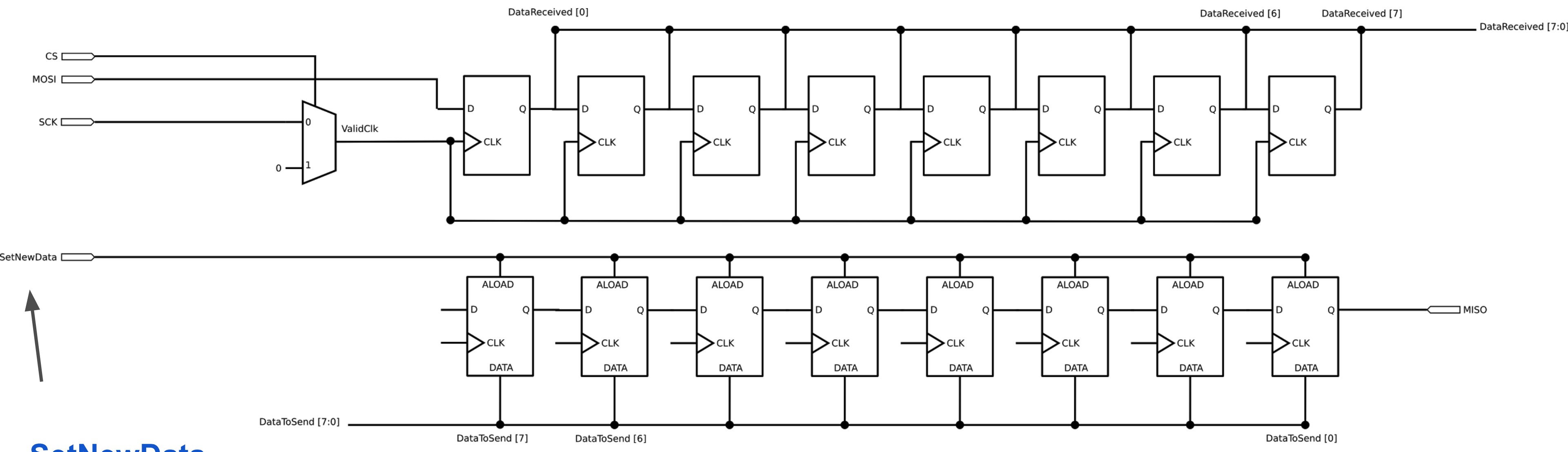


"Load a byte in parallel" with the ALOAD signal to be clocked out serially!

Sending and Receiving



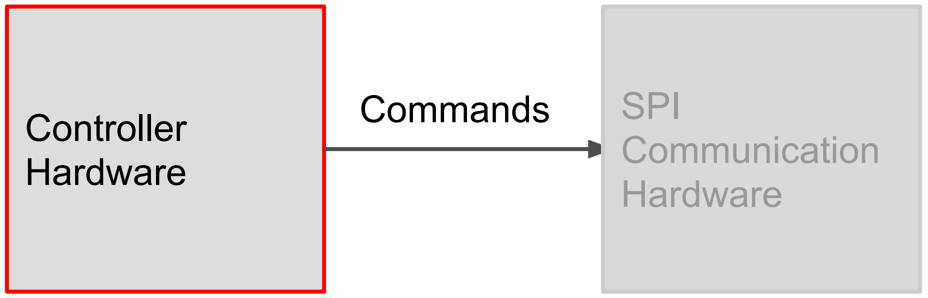
Sending and Receiving



SetNewData logic defined in controller

Building it up From Gates

Part Two: the Controller



Controller

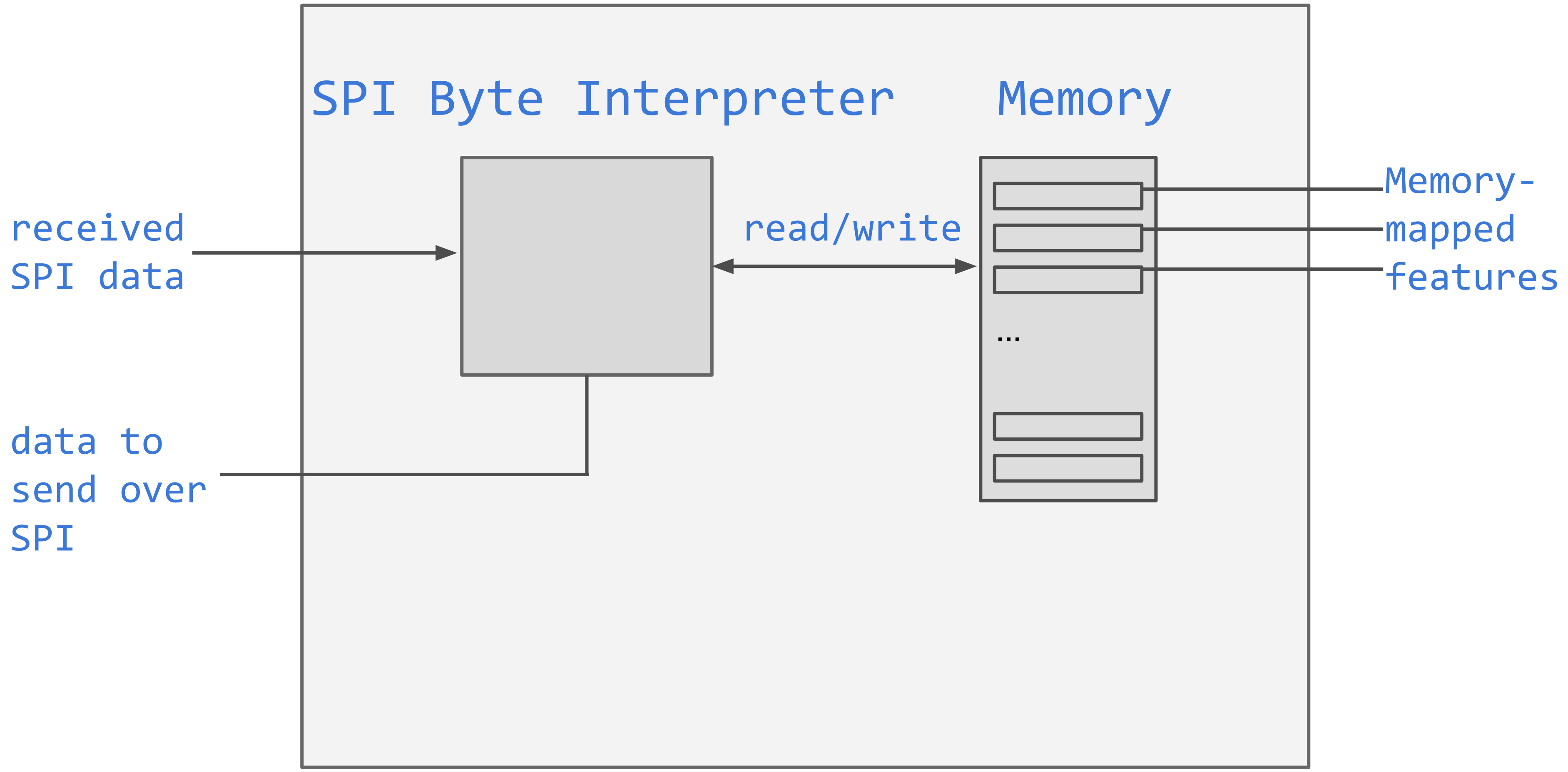
Organizes sent/received data

Allows us to read/write to/from register in internal memory.

Memory Addresses are mapped to store data for controlling/reading external features.

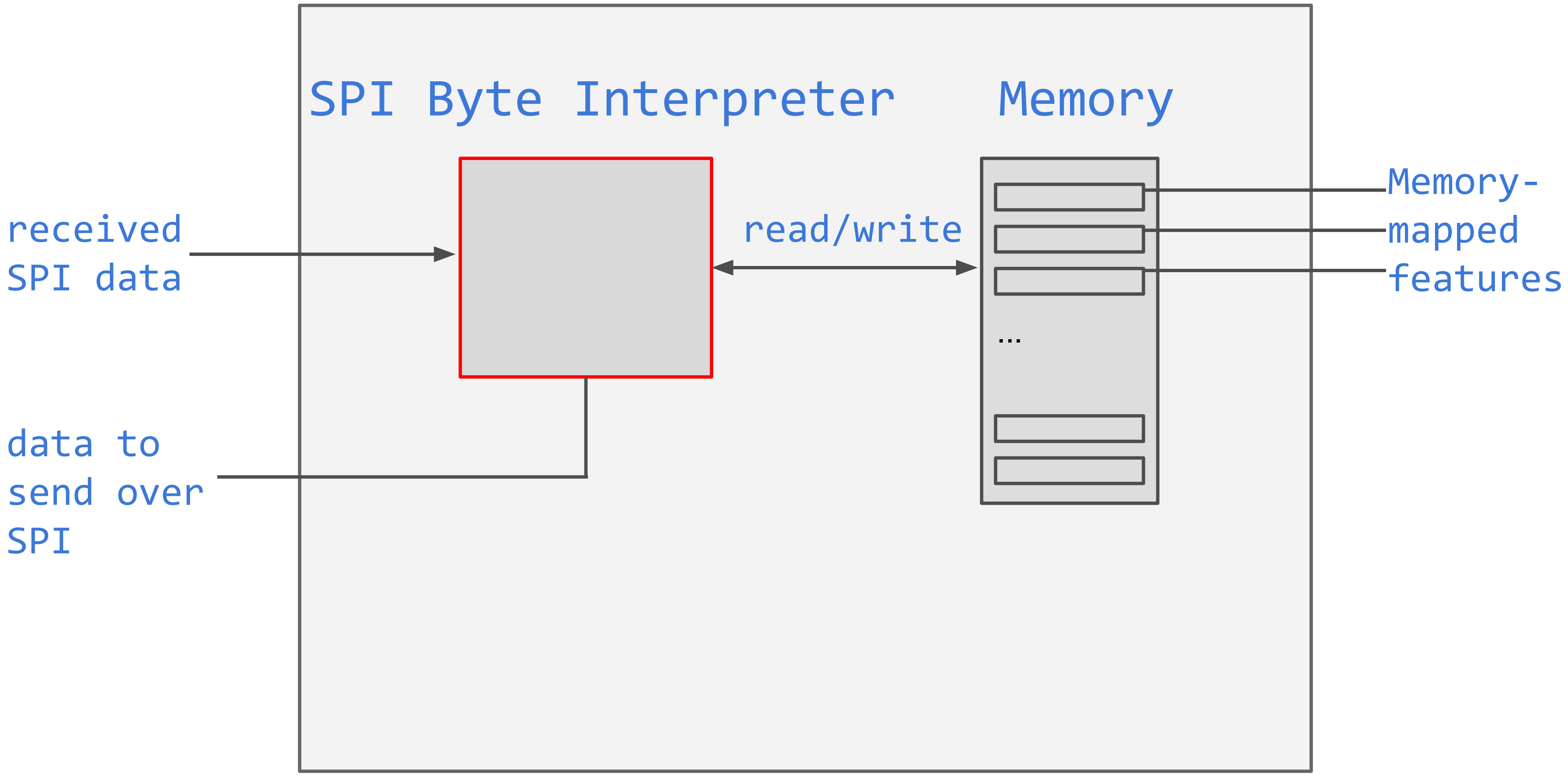
Controller

Organizes sent/received data



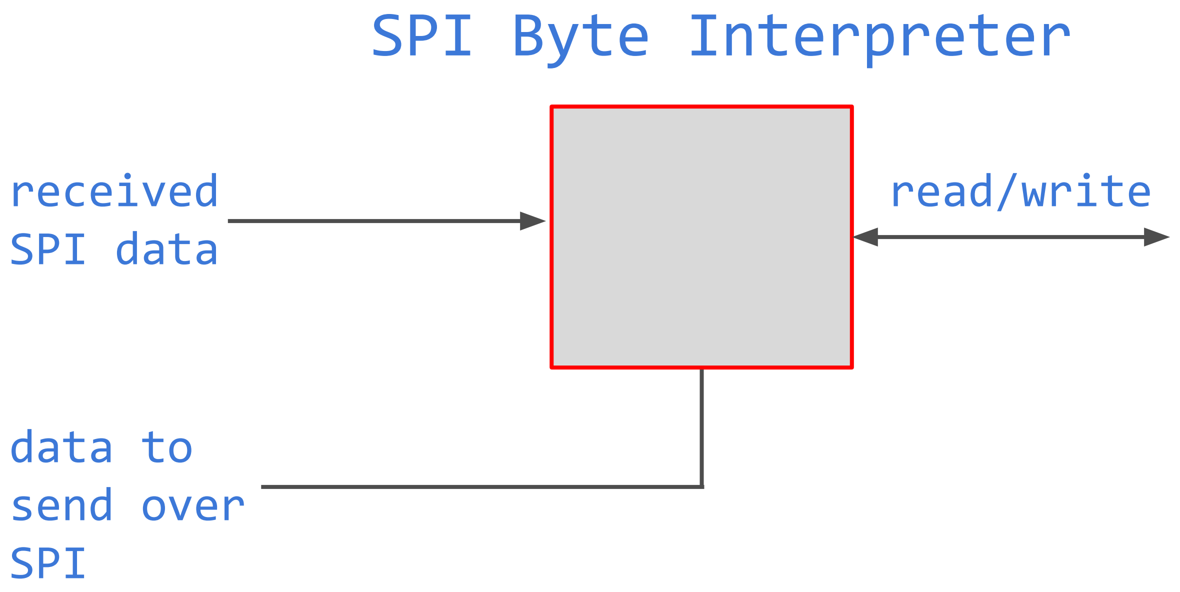
Controller

Organizes sent/received data

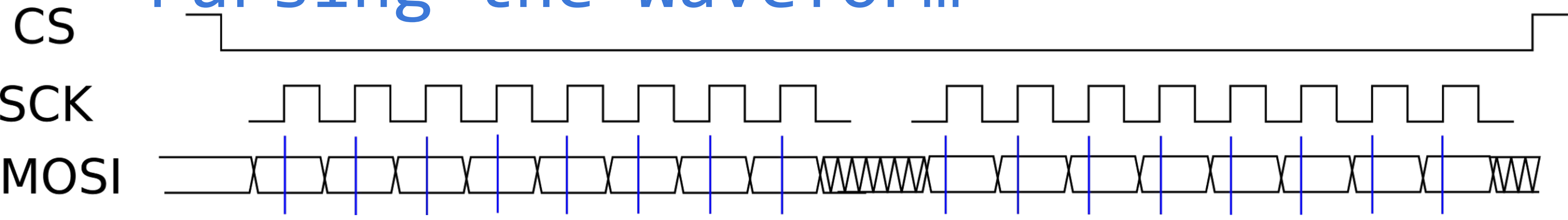


SPI Byte Interpreter

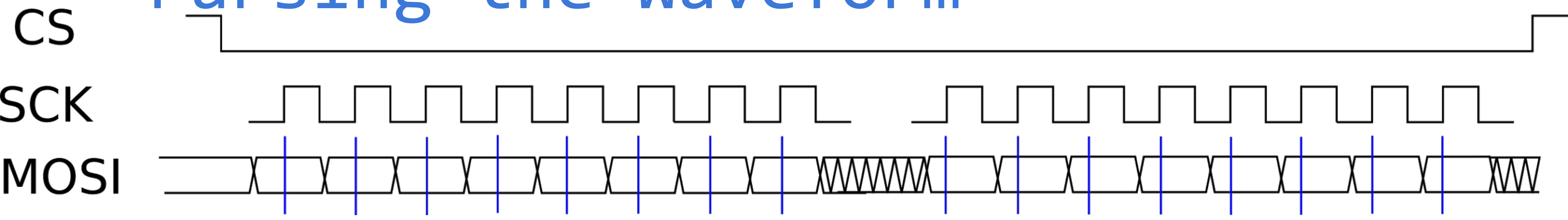
- Identifies start of SPI transfer
- Identifies *read* or *write* command
- Fetches data (if *read*)
- sets data (if *write*)



Parsing the Waveform

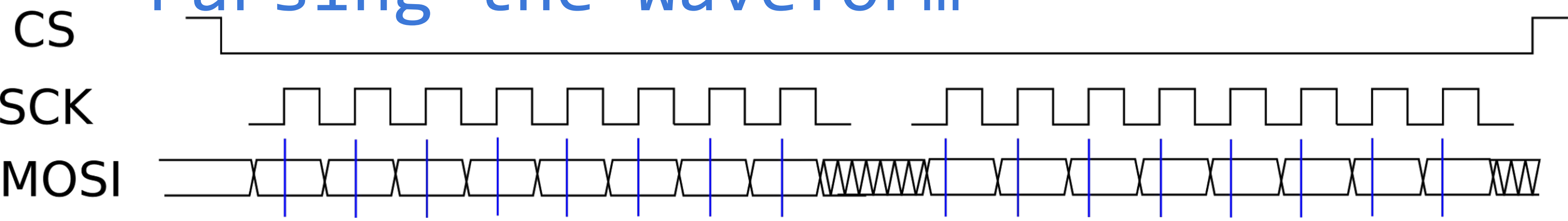


Parsing the Waveform

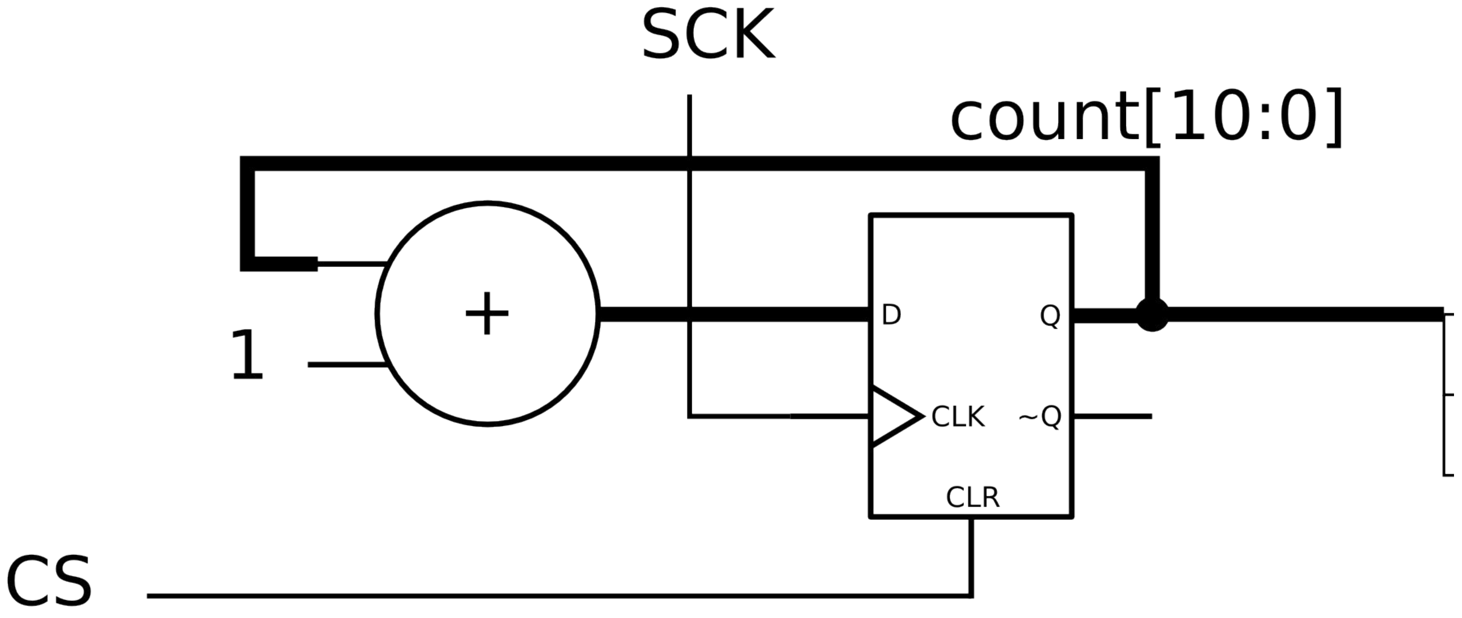


First Step: count bits

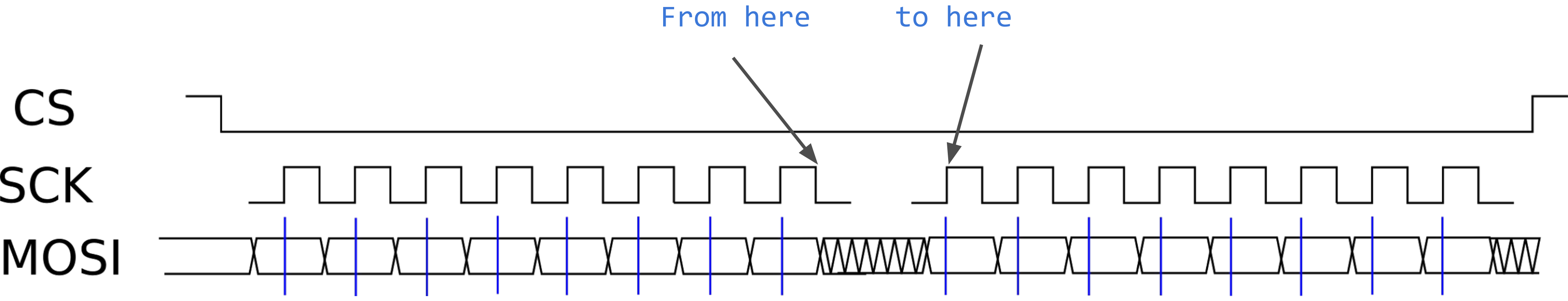
Parsing the Waveform



Counter!

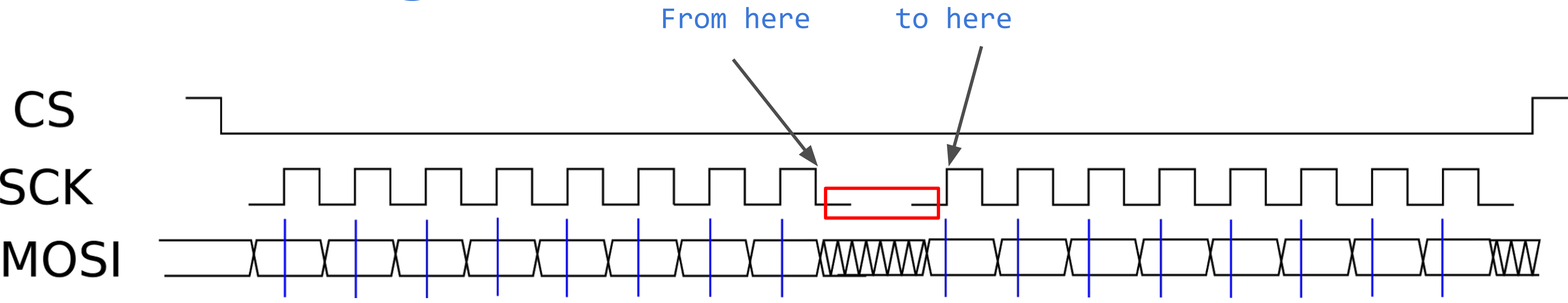


Parsing the Waveform

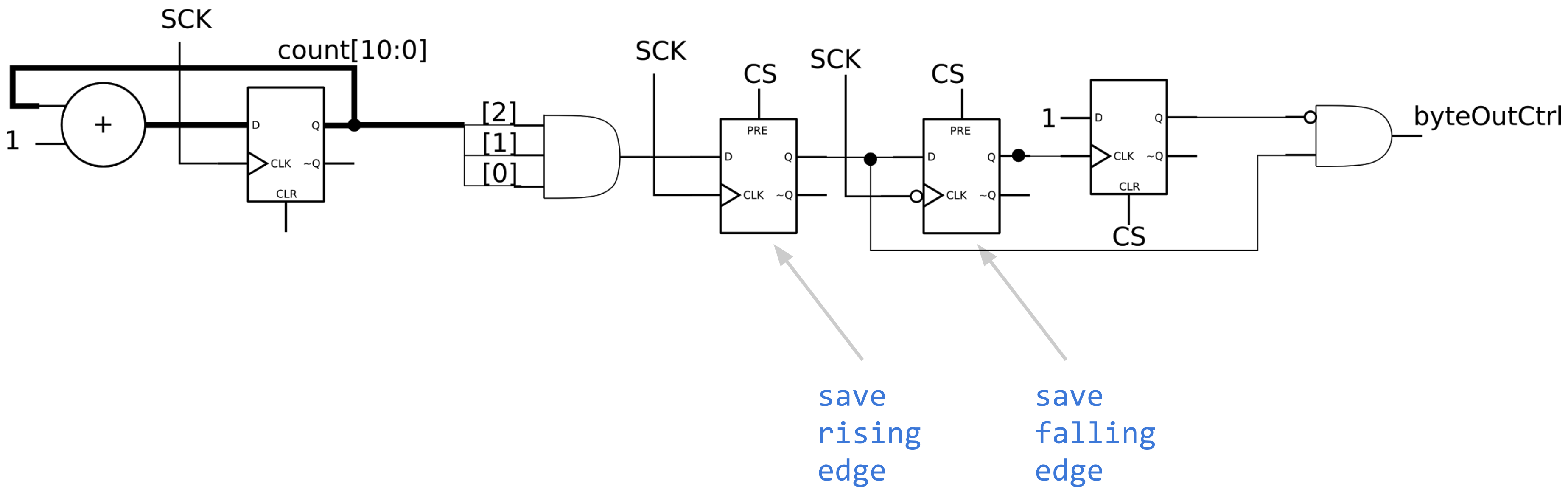


Next Step: identify when 1st byte has arrived.
i.e: count to 8 and catch clock falling edge.

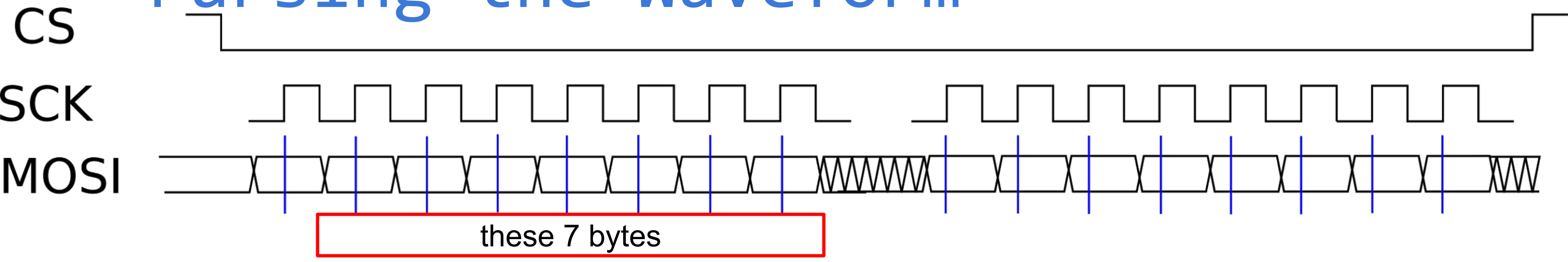
Parsing the Waveform



three flip flops and the counter!

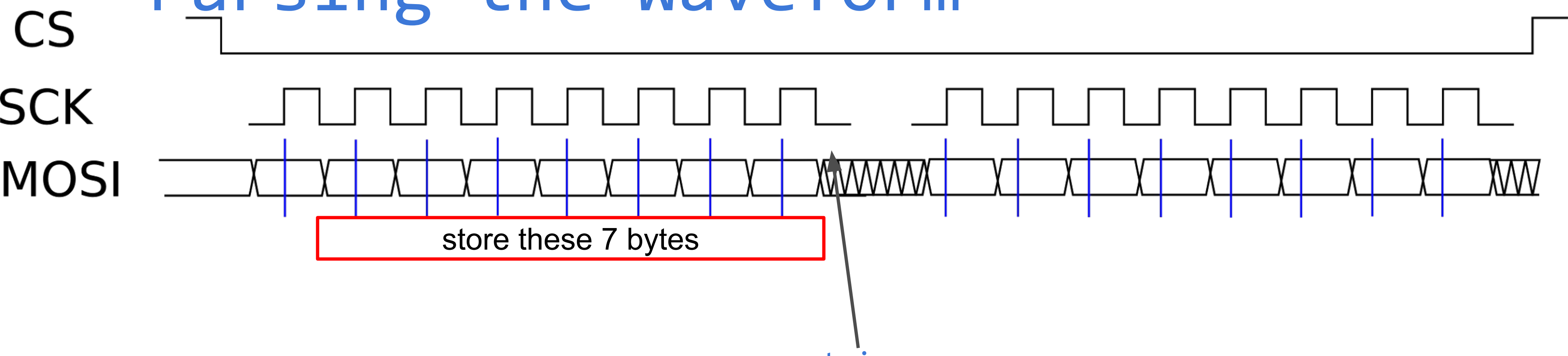


Parsing the Waveform

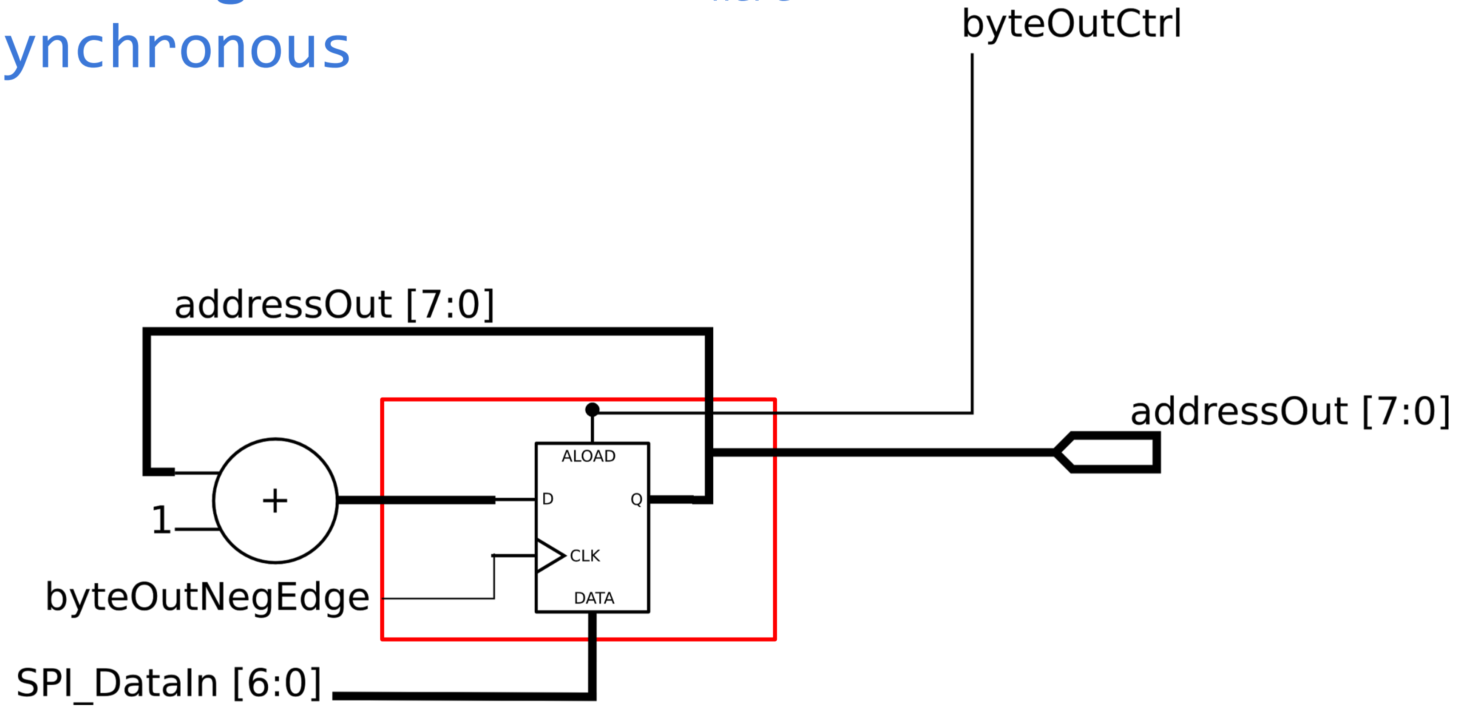


Next Step: Store the starting address

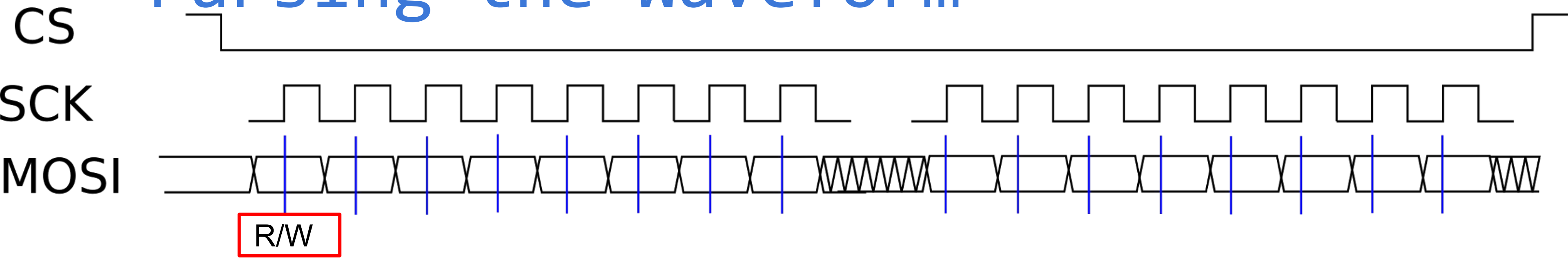
Parsing the Waveform



8-bit Shift-Register (with asynchronous loading)

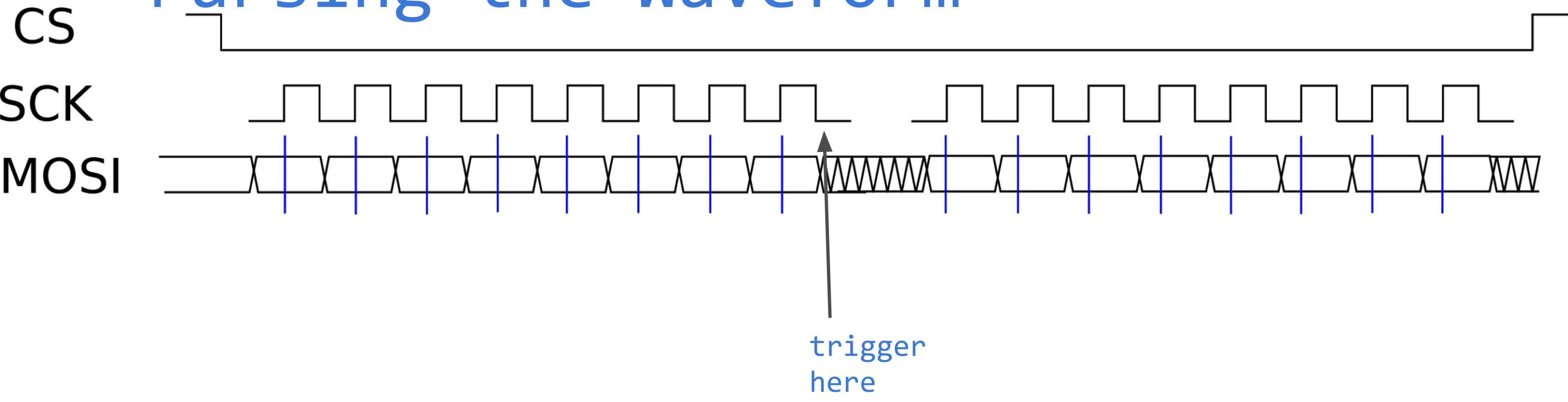


Parsing the Waveform

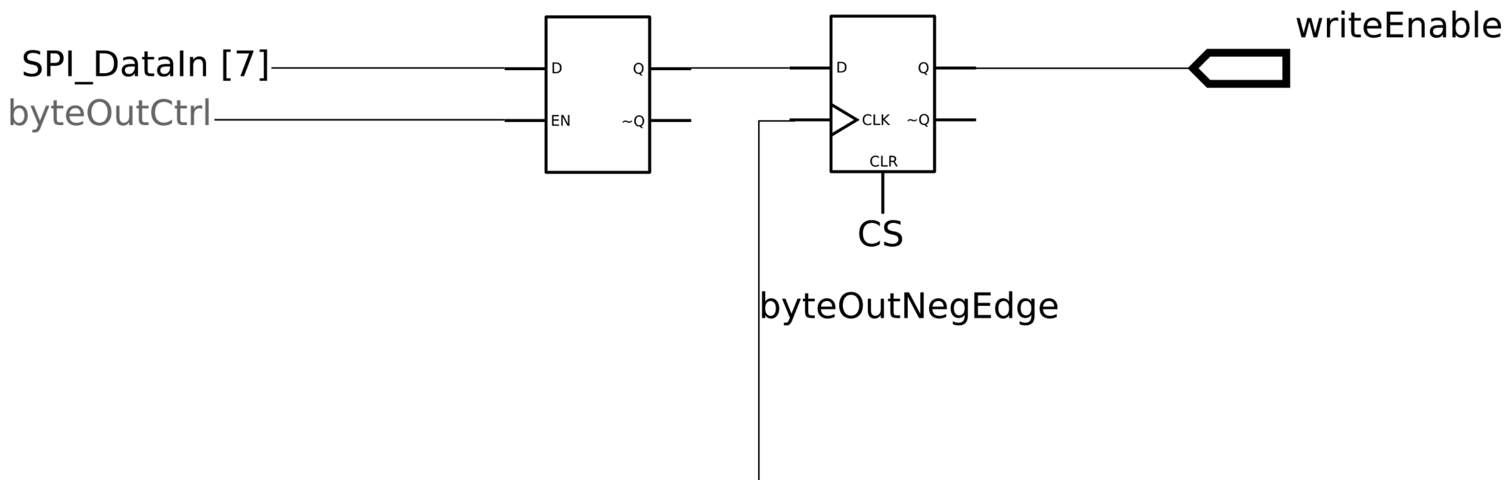


Next Step: identify read or write

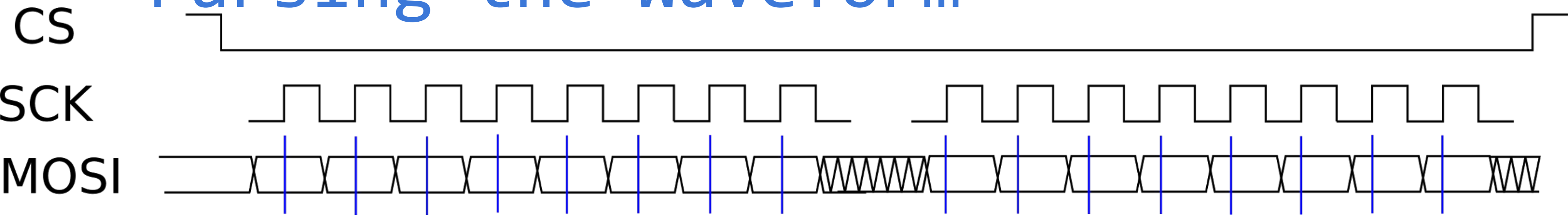
Parsing the Waveform



Save MSbit from 1st byte transferred.

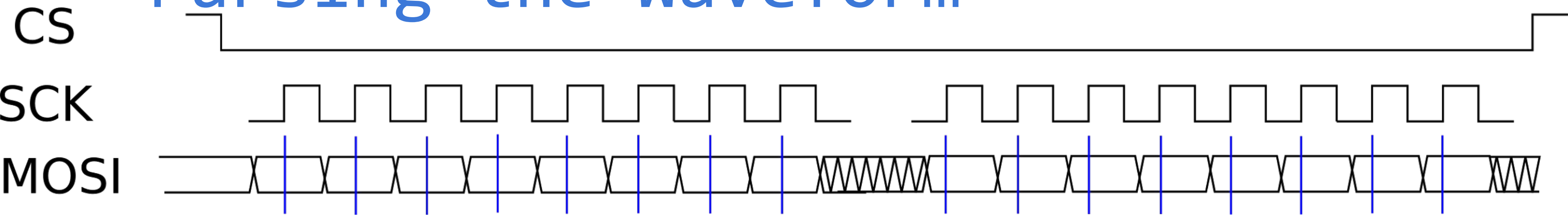


Parsing the Waveform

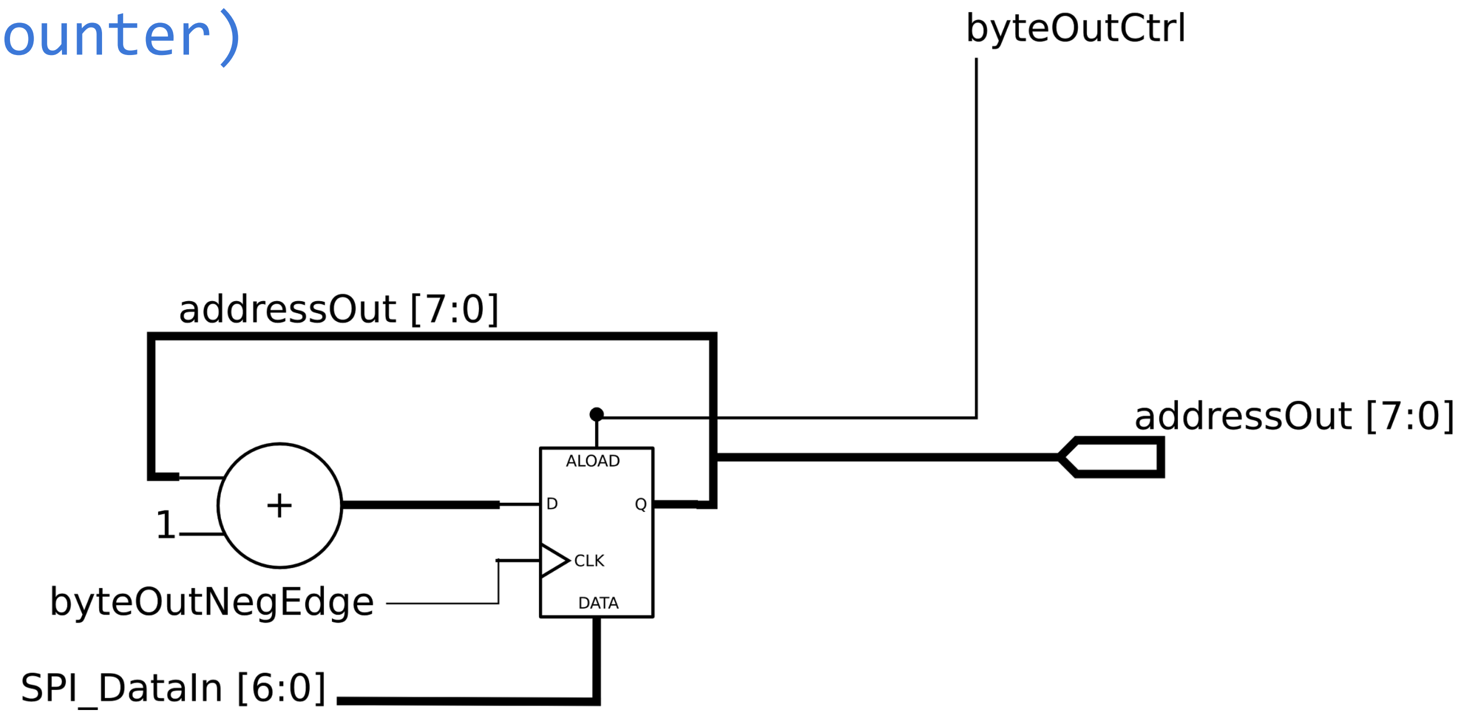


Next Step: increment the address after each byte.

Parsing the Waveform

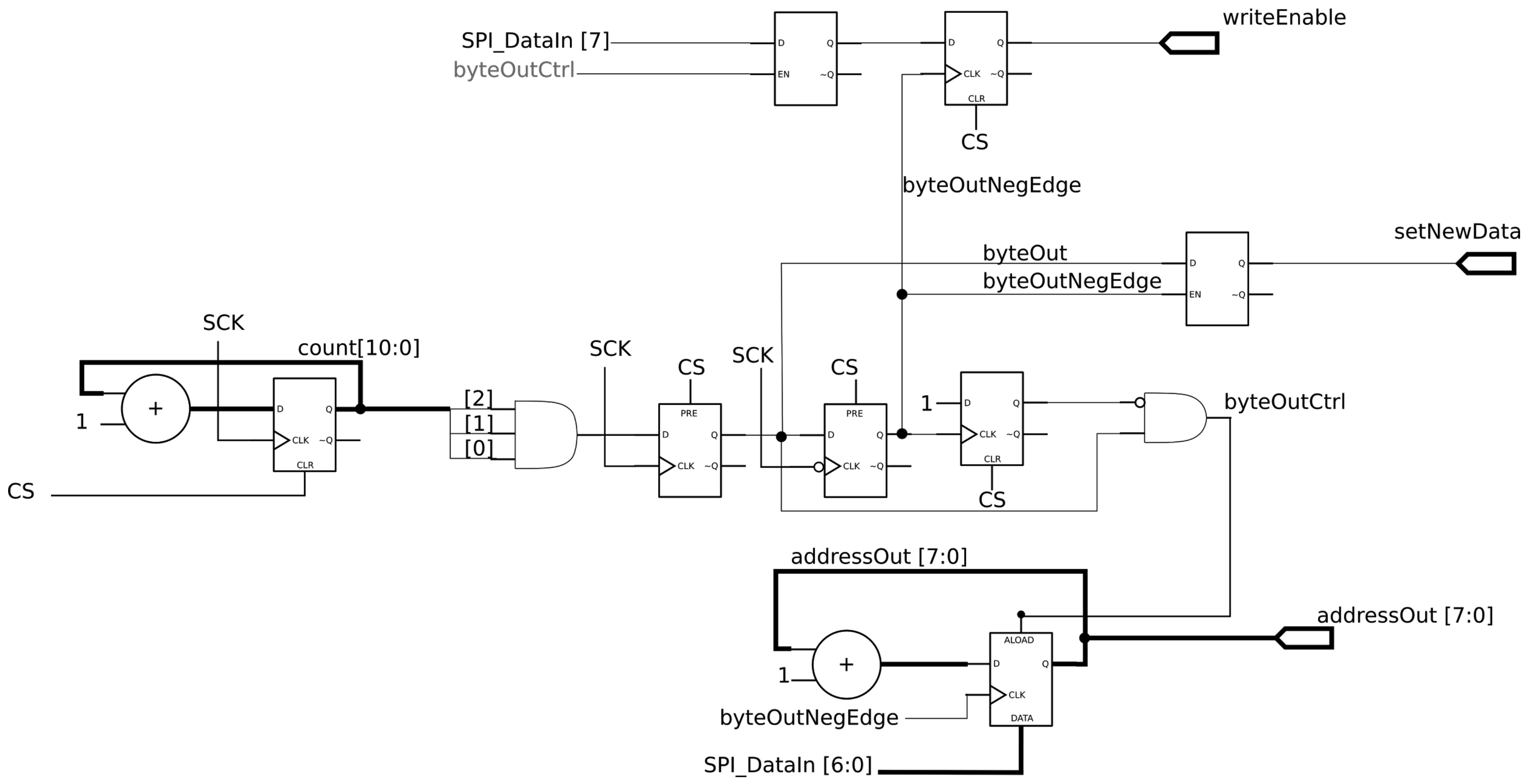


Another counter! (count up once per byte using bit-counter)



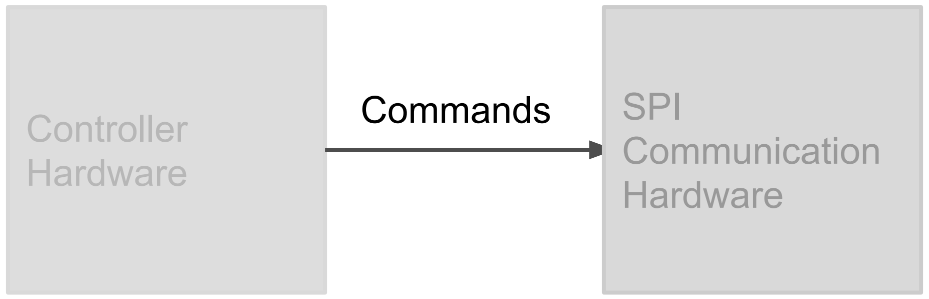
Parsing the Waveform

SPI Controller Complete

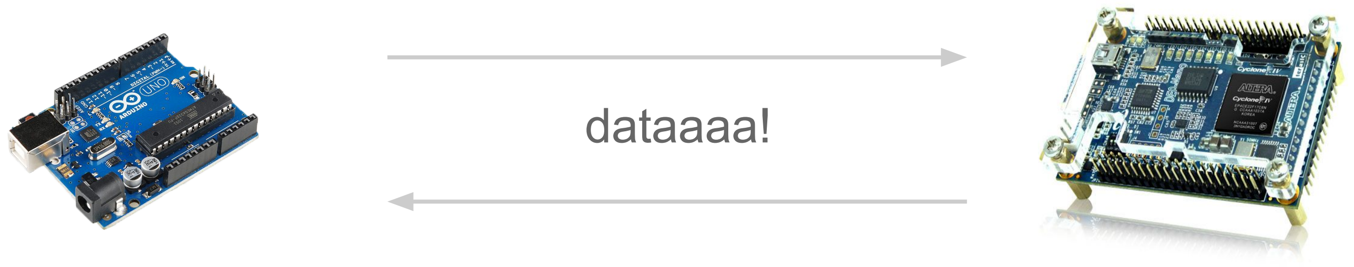


Building it up From Gates

Both Parts Complete!



Wrap-up



Wrap-up

Questions?

- Reading Example
  - https://github.com/Poofjunior/HardwareModules/tree/master/SPI_EncoderReader
- Writing Example
  - https://github.com/Poofjunior/HardwareModules/tree/master/ServoExtender