# Hacking a Telescope:
## Motor Control for PAT Development

Spencer Hart

Group 66 Intern Presentation

08/01/2025

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.
This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001 or FA8702-25-D-B002. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.
© 2025 Massachusetts Institute of Technology.
Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

# About Me

- **Hometown: Newton, MA**
- **Education: University of Vermont**
  - **Electrical Engineering**
  - **Rising senior**
- **Other**
  - **Lacrosse**
  - **Skiing**

# Outline

- Motivation
- Motor & Encoder
- Elmo
- FPGA
- Future Work

# Motivation

- **Sensors required to test PAT capabilities using stars are available in advance of certain actuators (*e.g.*, gimbal)**
- **Need to test/calibrate sensors**
  - **Need control over COTS actuators to allow for sensor testing**
- **Goal: converge on the target quickly and smoothly**
- **Need high-rate position and time updates from gimbal axes and actuating mirrors (current rate: 1Hz, desired rate: 100Hz)**
- **Faster position/time updates = faster misalignment model updates = faster error correction**



**COTS = Commercial Off-The-Shelf** **PAT = Pointing, Acquisition, and Tracking**

# Motivation: Optical Path Misalignment Model

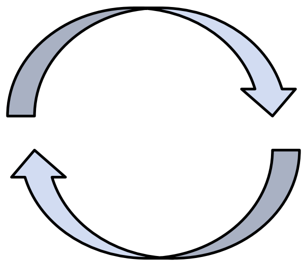
| Gimbal Axis | Entrance Mirror |
|---|---|
| G1 (Elevation) | M5 |
| G2 (Azimuth) | M7 |

| Mirror | Description |
|---|---|
| M1:4 | Telescope |
| M5 | El Periscope |
| M6 | Coudé Relay |
| M7 | Az Periscope |
| M8 | Out-of-plane Fold |
| M9 | FSM |
| M10 | LBM |
| M11 | PBS |

Mirror Actuation

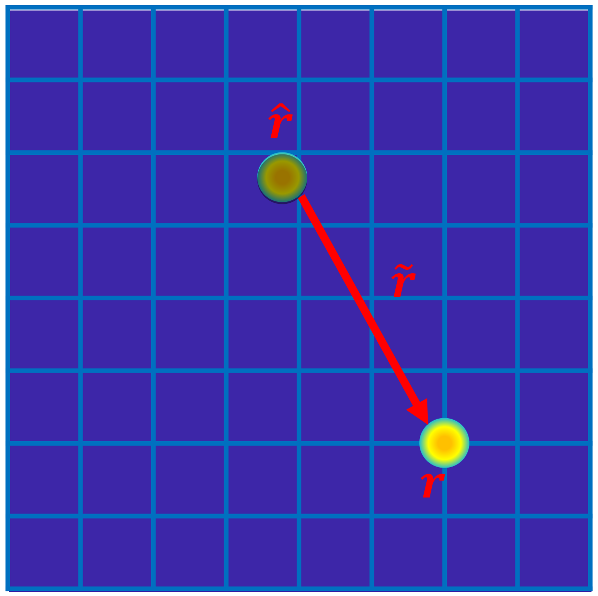
Gimbal Actuation

**Expected target spot location: $\hat{r}$**



**Misalignment Updates:**

$$\hat{\epsilon}^{+} = \hat{\epsilon}^{-} + \frac{\partial \tilde{r}}{\partial \epsilon} \tilde{r}$$




**FPA Measurements**

**Optical model can be updated on measurements of FSM and gimbal positions**

FSM = Fast Steering Mirror
LBM = Look Behind Mirror
FPA = Focal Plane Array

$\epsilon$ = position update
$\hat{\epsilon}^{+}$ = new position update
$\hat{\epsilon}^{-}$ = previous position update

$r$ = actual target location
$\hat{r}$ = expected target location
$\tilde{r} = (r - \hat{r})$ = target position error

# Outline

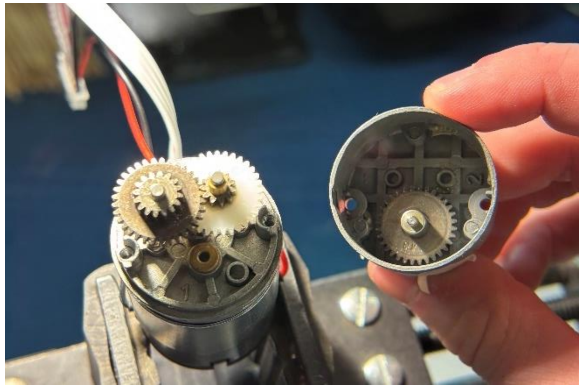
- Motivation
- Motor & Encoder
- Elmo
- FPGA
- Future Work



LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

# Motor & Encoder

- **Celestron NexStar Telescope**
- **Azimuth and elevation controlled by brushed DC motors**
- **54:1 Gear ratio**
- **6-pin connector feeds into motor driver board**
  - **Motor +**
  - **Motor –**
  - **+5V**
  - **GND**
  - **B (quadrature signal)**
  - **A (quadrature signal)**





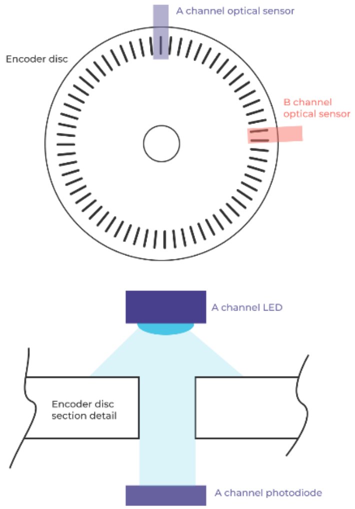
LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

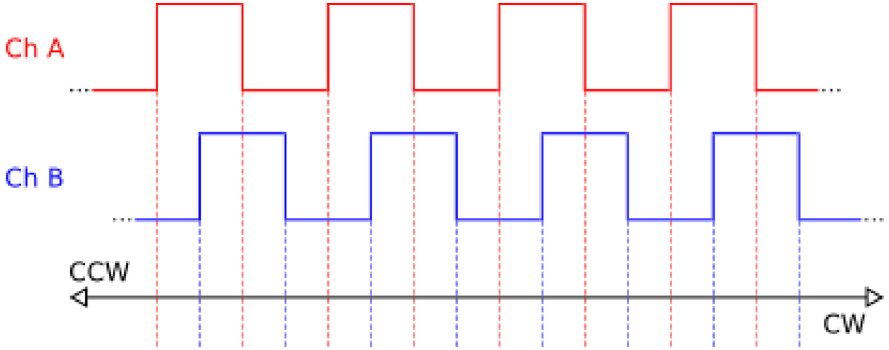
# Motor & Encoder

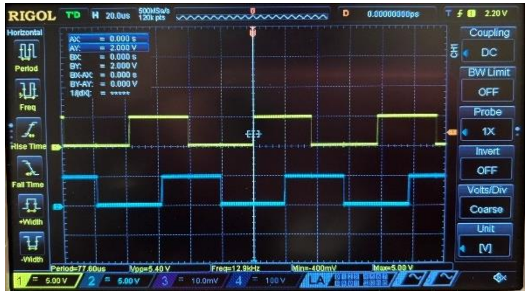
- US Digital E4T optical encoder
- Outputs single-ended A & B quadrature signals
- 400 $\text{CPR}_{\text{internal}}$ → 1,600 $\text{PPR}_{\text{internal}}$
- Use gear ratio to find $\text{PPR}_{\text{external}}$

$$\rightarrow \frac{1\ External\ Revolution}{54\ Internal\ Revolutions} = \frac{1{,}600\ PPRinternal}{1\ Internal\ Revolution}$$

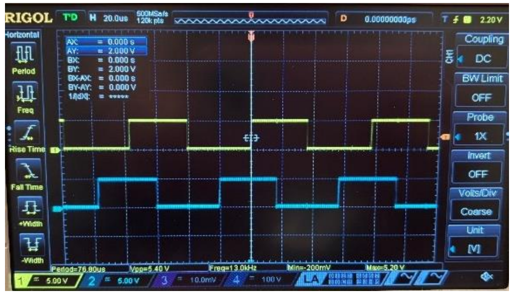
$$\rightarrow 54 \times 1,600 = 86,400\text{PPR}_{\text{external}}$$



A (yellow) leads B (blue)



B (blue) leads A (yellow)



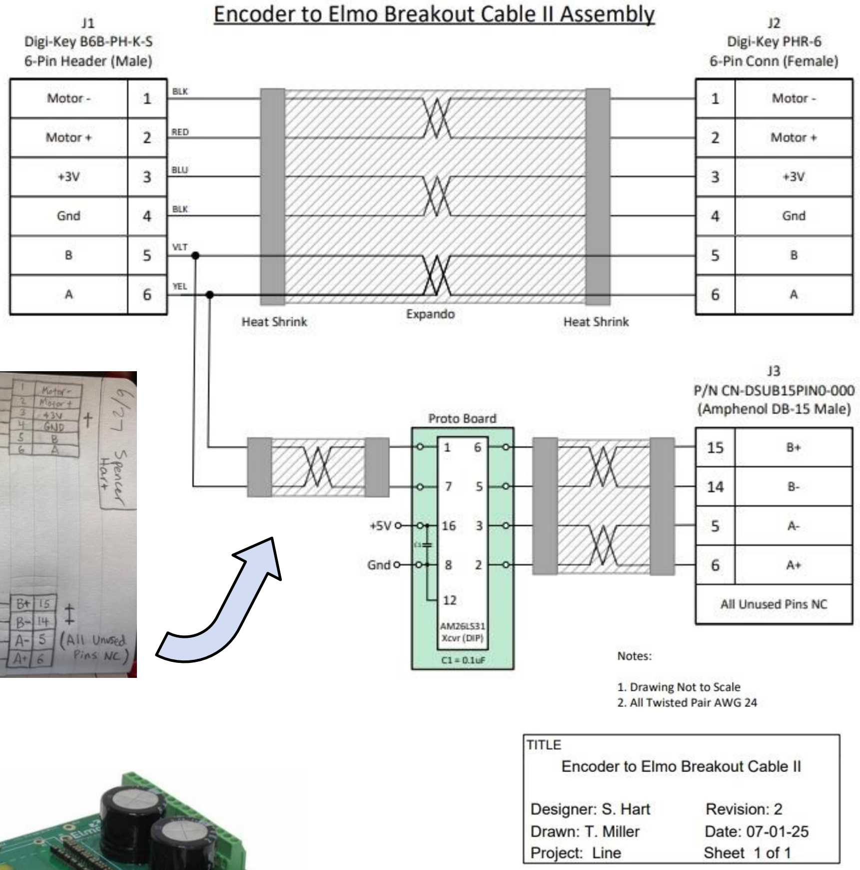
CPR = Cycles Per Revolution PPR = Pulses Per Revolution

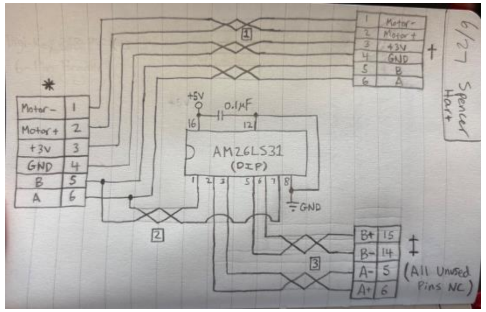
# Outline

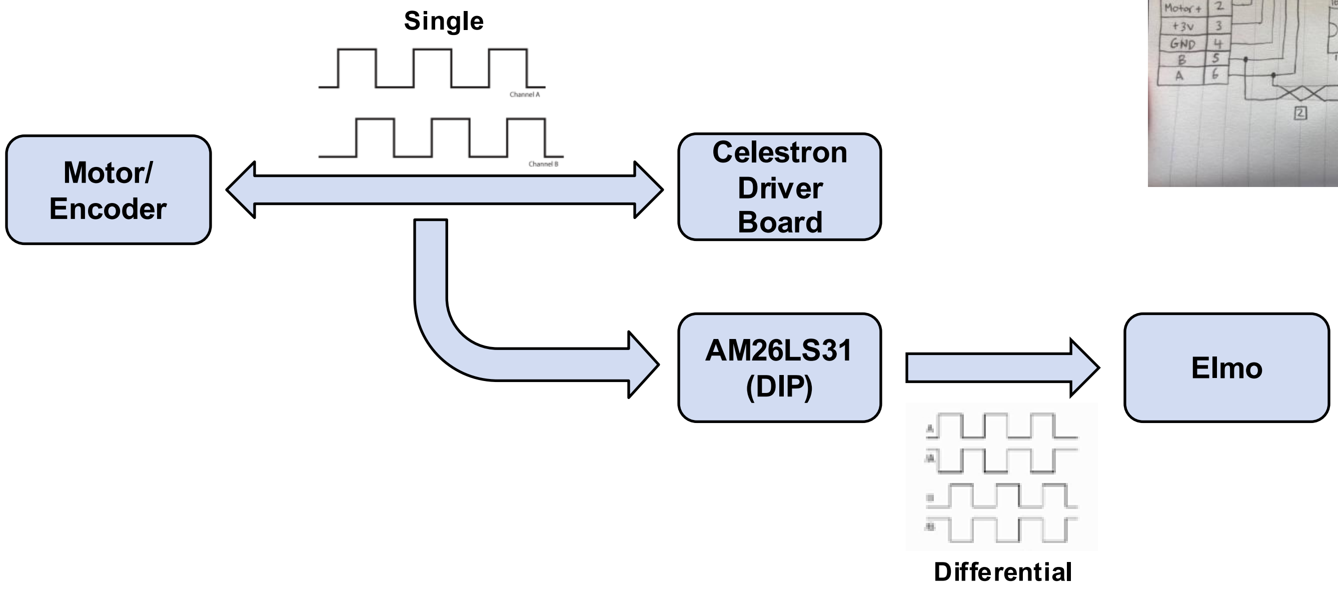
- Motivation
- Motor & Encoder
- Elmo
- FPGA
- Future Work

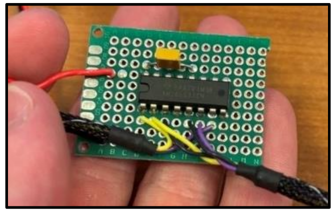
# Elmo

- **Elmo Motion Control Evaluation Board**
- **Uses RS232 or CAN to communicate**
- **Use case 1: “sniff” encoder for position value**
- **Expects differential quadrature signals**
- **Single-differential converter required**





**RS232 = Recommended Standard 232 (communication protocol)**

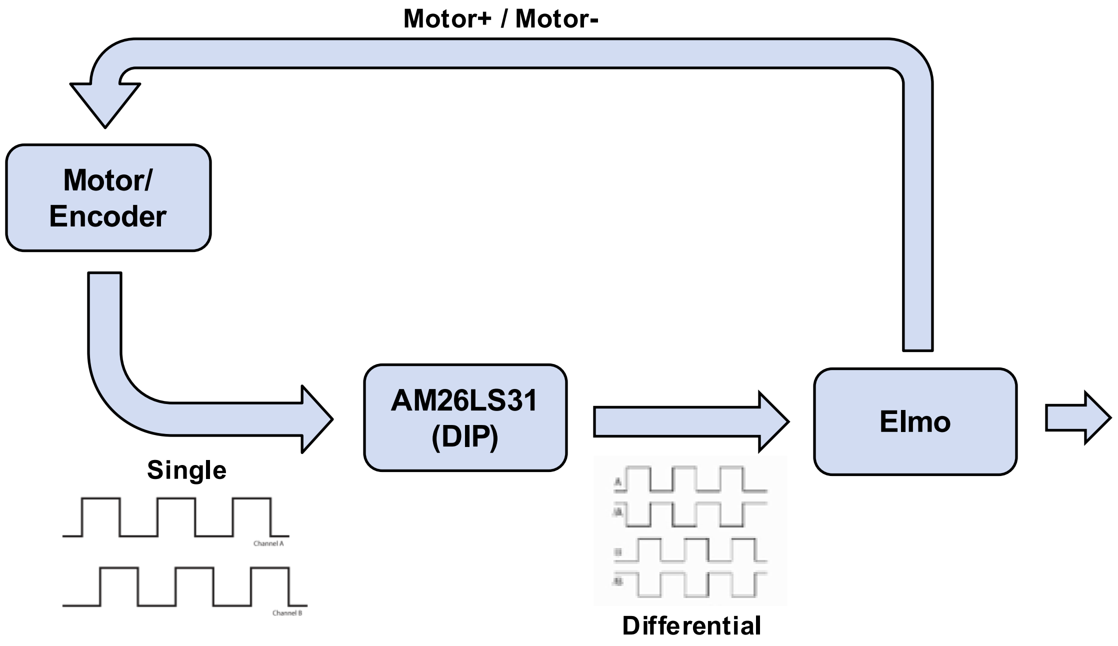
**CAN = Controller Area Network (communication protocol)**

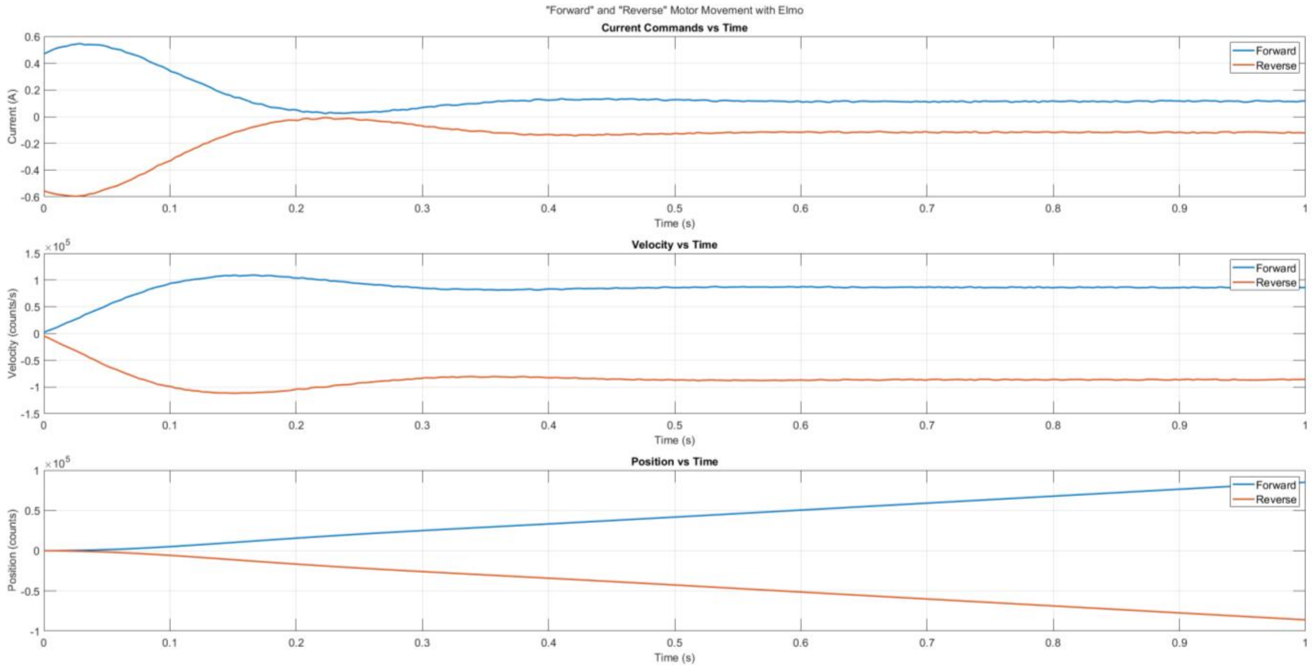
**DIP = Dual Inline Package (integrated circuit)**


LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

# Elmo

- **Use case 2: drive motor and “sniff”**
- **Eliminates use of Celestron driver board**
- **Currently: start, stop, forward, reverse capabilities with velocity control**






LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

# Outline

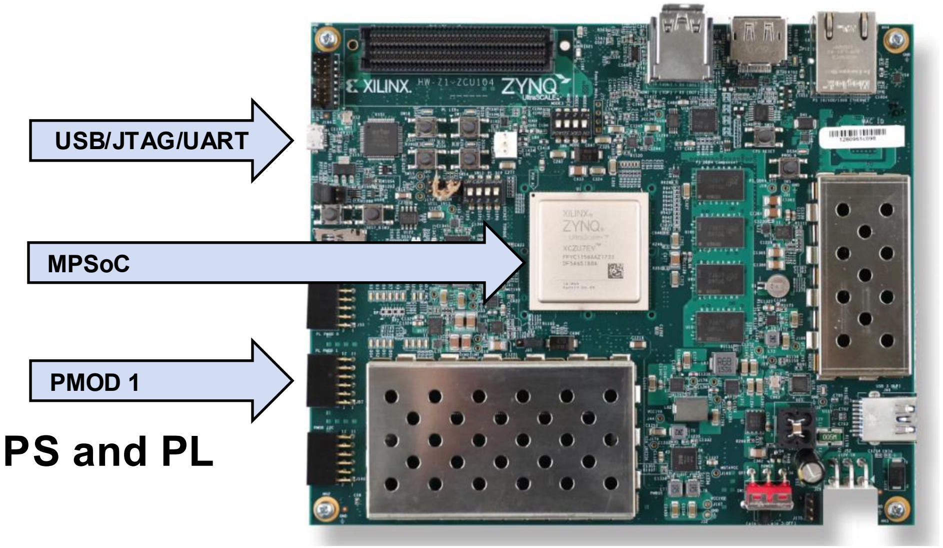
- Motivation
- Motor & Encoder
- Elmo
- FPGA
- Future Work



LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

# FPGA

- Xilinx Zynq UltraScale+ MPSoC ZCU104
- MPSoC = Multi-Processor System on Chip
  - Single integrated circuit with both PS and PL
    - PS = Processing System → APU, RPU, GPU
    - PL = Programmable Logic → FPGA fabric
    - AXI = Advanced eXtensible Interface → Connections between PS and PL
- FPGA = Field Programmable Gate Array
  - High clock rates and parallel processing
    - Low latency and high throughput
  - Rapid, dynamic prototyping, unlike an ASIC
    - Simulation
  - Interfacing with other hardware components
    - Real-time visualization and control of both internal and external components
  - No overhead of an OS → handled by PS in the case of Zynq



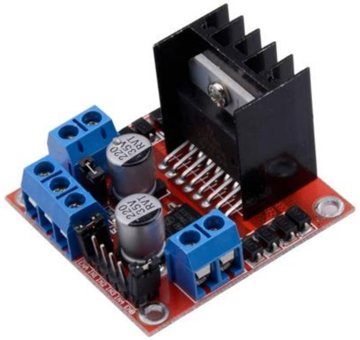


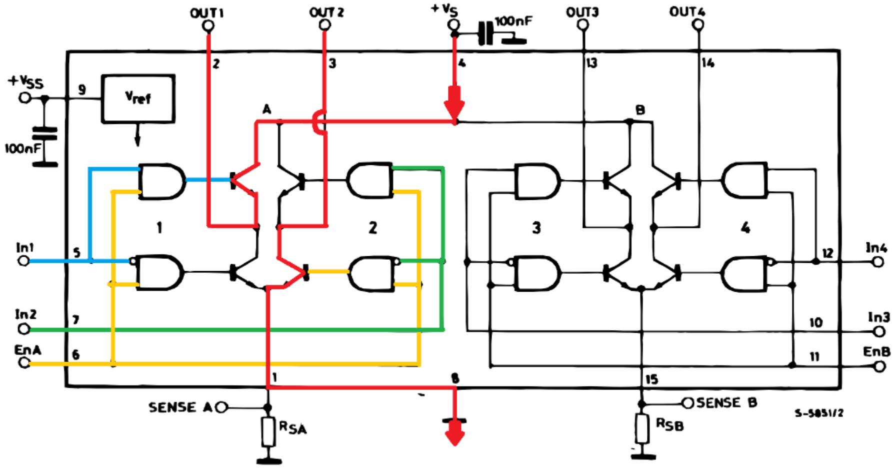

ASIC = Application Specific Integrated Circuit
OS = Operating Software
PMOD = Peripheral Module
USB/JTAG/UART = Communication Protocols/Serial Buses

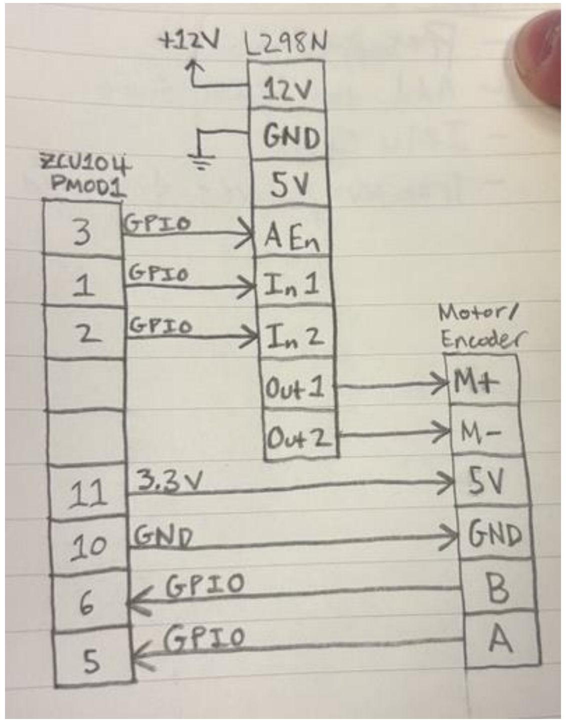
APU = Application Processing Unit
RPU = Real-time Processing Unit
GPU = Graphical Processing Unit

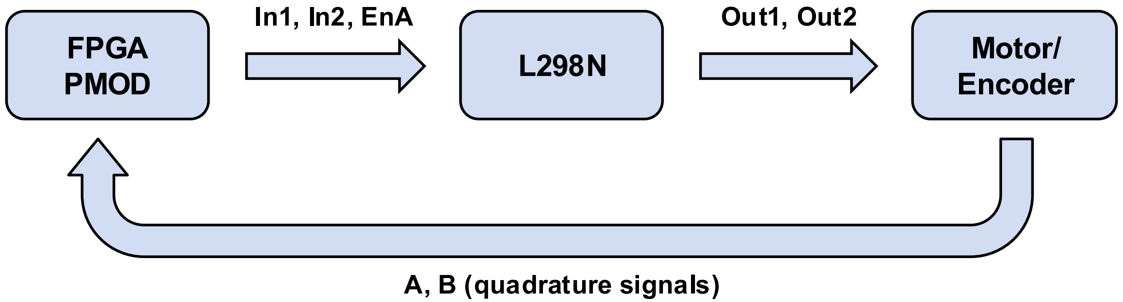
# FPGA

- **FPGA motor control**
- **L298N H-bridge (motor driver)**
  - EnA → motor on/off or PWM
  - In1, In2 → control motor spin direction
  - Out1, Out2 → drive motor (motor+ / motor-)
- **PMOD**
  - GPIOs → output direction and PWM, take in quadrature







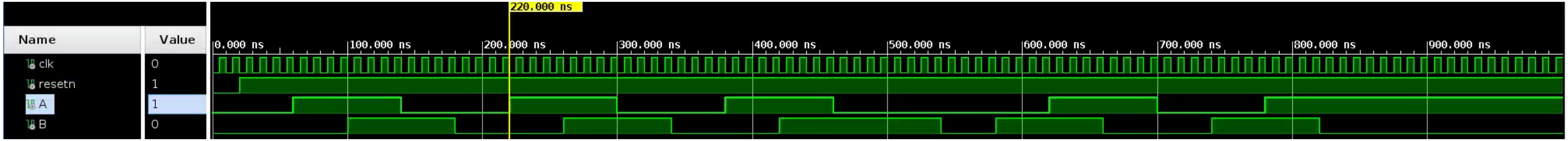



PWM = Pulse Width Modulation    GPIO = General Purpose Input/Output

# FPGA

- **Vivado Design Suite**
  - **HDL design, simulation, synthesis, and implementation**
- **Testbench design / simulation**

```
    -- Clock period
    constant clk_period : time := 10 ns;

    -- Procedure (or function to be "called") to generate quadrature pulses
    procedure quadrature_step(
        signal A : out std_logic;       -- "out" because it will be generated as an output to simulation
        signal B : out std_logic;       -- "out" because it will be generated as an output to simulation
        forward  : in boolean           -- Boolean to dictate direction of simulated motor
    ) is
    begin
        if forward then      -- if forward is TRUE
            -- Sequence: 00 → 01 → 11 → 10 → 00
            A <= '0'; B <= '0'; wait for 4 * clk_period;        -- 40 ns wait period bewteen signal changes
            A <= '0'; B <= '1'; wait for 4 * clk_period;
            A <= '1'; B <= '1'; wait for 4 * clk_period;
            A <= '1'; B <= '0'; wait for 4 * clk_period;
        else                 -- if forward is FALSE
            -- Reverse: 00 → 10 → 11 → 01 → 00
            A <= '0'; B <= '0'; wait for 4 * clk_period;
            A <= '1'; B <= '0'; wait for 4 * clk_period;
            A <= '1'; B <= '1'; wait for 4 * clk_period;
            A <= '0'; B <= '1'; wait for 4 * clk_period;
        end if;
    end procedure;

-- Stimulus process
stim_proc : process      -- Do stuff!
begin
    -- Reset
    wait for 20 ns;
    resetn <= '1';

    -- Forward rotation: 3 steps
    for i in 1 to 3 loop
        quadrature_step(A, B, false);       -- Call procedure "quadrature_step", set forward to false
    end loop;

    -- Wait and print
    wait for 40 ns;

    -- Reverse rotation: 2 steps
    for i in 1 to 2 loop
        quadrature_step(A, B, true);        -- Call procedure "quadrature_step", set forward to false
    end loop;

    -- Wait and finish
    wait for 100 ns;
    assert false report "Simulation completed" severity note;
    wait; -- forever
end process;
```

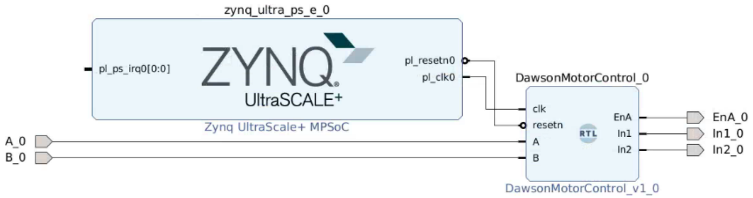


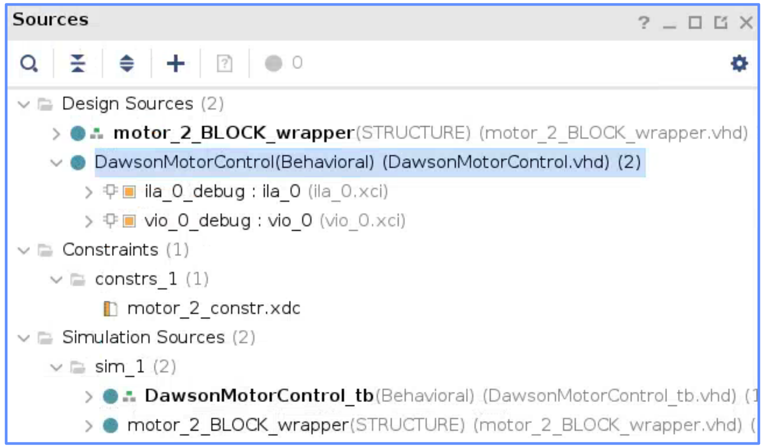



HDL = Hardware Description Language

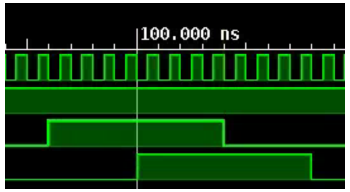
# FPGA

- **Block design**
- **Top level HDL design**
- **ILA / VIO IP blocks**

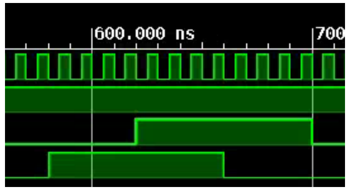



```
-- Quadrature Decoder process
process(clk)          -- "process" is the main description area of the behavior of the "thing"
                      -- Processes run in parallel!
                      -- "process(clk)" indicates the process uses synchronous logic
begin       -- begin process
    if rising_edge(clk) then                        -- Logic sequence evaluated upon rising clock edges
        if resetn = '0' then                        -- If asynch. reset LOW --> reset values to 0
            position_vec <= (others => '0');        -- Reset values to zero upon reset
            A_prev <= '0';
            B_prev <= '0';
        else                  -- else, register A & B values and increment position_vec according to the logic sequence below
            A_reg <= A;
            B_reg <= B;

            if (A_reg /= A_prev) or (B_reg /= B_prev) then      -- (Best illustrated with drawings...)
                if (A_prev = '0' and B_prev = '0') then
                    if (A_reg = '1') then
                        position_vec <= position_vec + 1;
                    elsif (B_reg = '1') then
                        position_vec <= position_vec - 1;
                    end if;
                elsif (A_prev = '1' and B_prev = '0') then
                    if (B_reg = '1') then
                        position_vec <= position_vec + 1;
                    elsif (A_reg = '0') then
                        position_vec <= position_vec - 1;
                    end if;
                elsif (A_prev = '1' and B_prev = '1') then
                    if (A_reg = '0') then
                        position_vec <= position_vec + 1;
                    elsif (B_reg = '0') then
                        position_vec <= position_vec - 1;
                    end if;
                elsif (A_prev = '0' and B_prev = '1') then
                    if (A_reg = '1') then
                        position_vec <= position_vec - 1;
                    elsif (B_reg = '0') then
                        position_vec <= position_vec + 1;
                    end if;
                end if;
            end if;

            A_prev <= A_reg;        -- Update _prev values
            B_prev <= B_reg;

        end if;
    end if;
end process;

-- Direction Control process
process(clk)
begin       -- begin process
    if rising_edge(clk) then
        if resetn = '0' then
            In1 <= '1';     -- Input values should be opposite for motor movement in either direction (CW or CCW) (see diagram/search L298N)
            In2 <= '0';
        else
            if (dir(0) = '0') then
                In1 <= '1';
                In2 <= '0';
            else
                In1 <= '0';
                In2 <= '1';
            end if;
        end if;
    end if;
end process;
```




**A leads B**

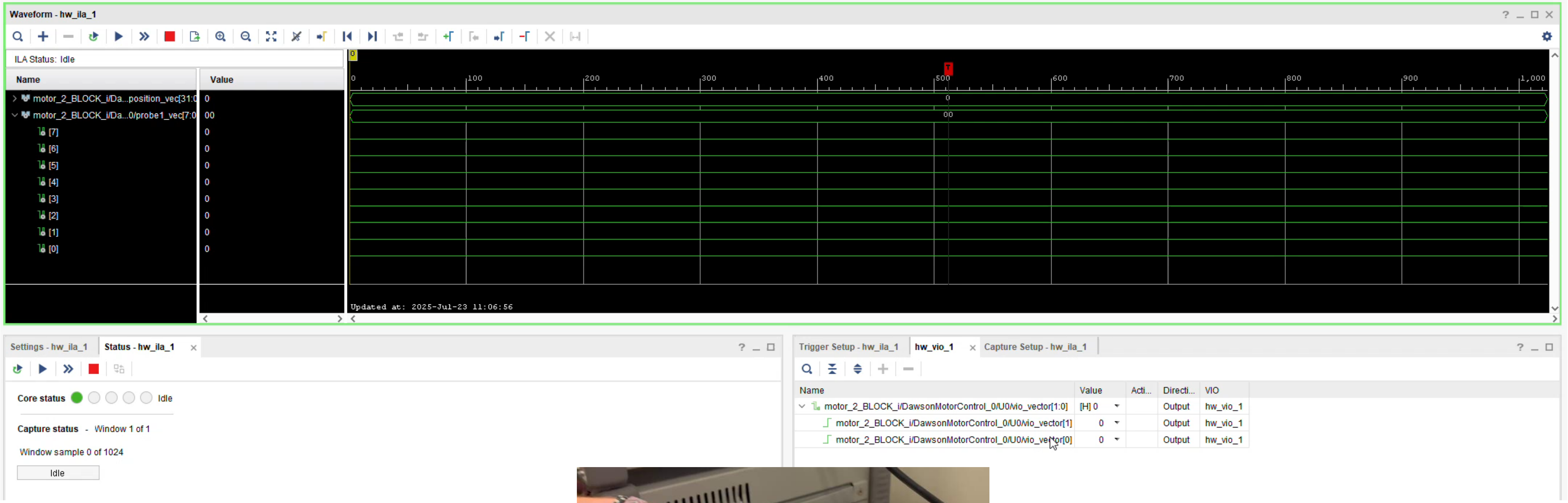


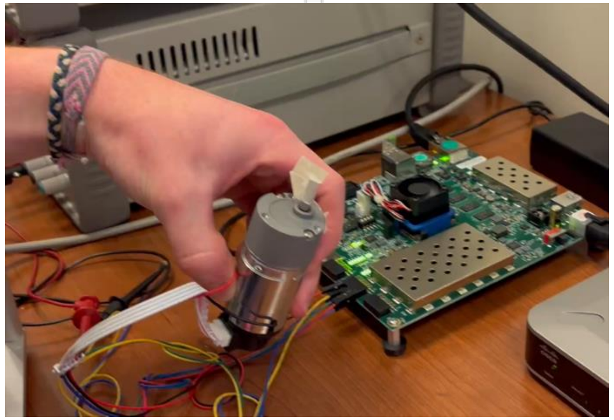

**B leads A**



**ILA = Integrated Logic Analyzer** **VIO = Virtual Input/Output**

# FPGA






LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

# Outline

- Motivation
- Motor & Encoder
- Elmo
- FPGA
- Future Work

# Future Work

- **Elmo path or Zynq path both viable for motion control**
- **Elmo path**
  - **Pre-existing software, known hardware**
  - **Convenient motion control and graphing capabilities**
  - **Sometimes difficult to navigate, limited examples online**
- **Zynq path**
  - **Can do what the Elmo does**
  - **Can also read FPA frames → "One-Stop Shop Controller"**
    - **Reduce hardware footprint on testbed**
  - **Would implement PWM and PID control for smoother velocity and tracking control**
  - **Would implement seamless interface between PS and PL using AXI bus**
- **Dual motor control to replicate true gimbal with azimuth and elevation axes**



**PID = Proportional, Integral, Derivative**

# Thank You!

- Alicia Volpicelli
- Andrew Gattineri
- Andrew Meccariello
- Angelina Zhang
- Christopher Foy
- Dawson Cohen
- David Geisler
- David Spencer
- Frank Mola
- Jonah Tower
- Joy Ma
- Kendra Sauer
- Lilly Johannan
- Michele Weatherwax
- Pamela Serrilla
- Richard Kaminsky
- Robert D'Ambra
- Robert Murphy
- Tasha LaSpisa
- Timothy Yarnall
- Tobias Chang
- Tom Miller
- Group 66
- Group 72