FPGA Field Oriented Control

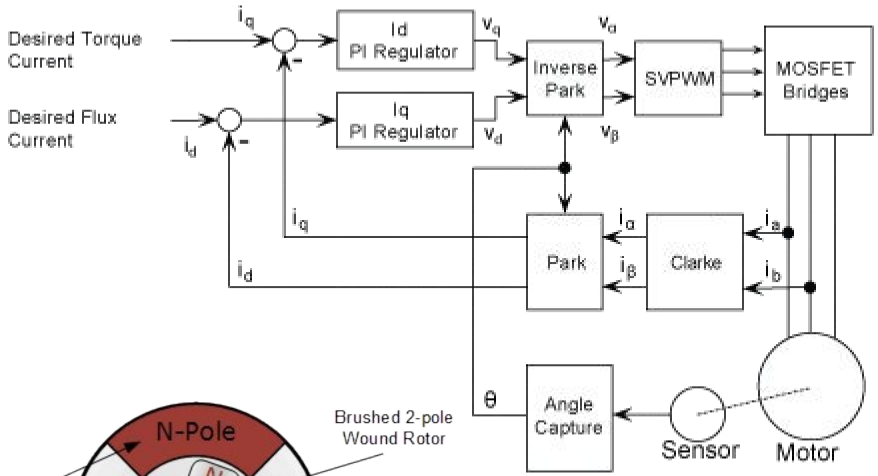
Jackson Gray, Aaron Yeiser

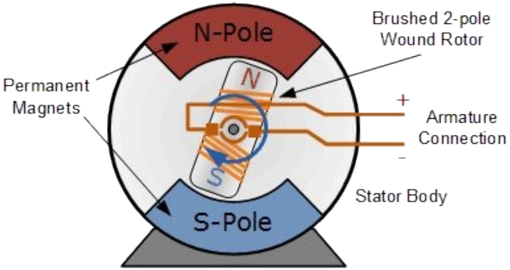
Brushless motors are cool and controlling them properly is hard

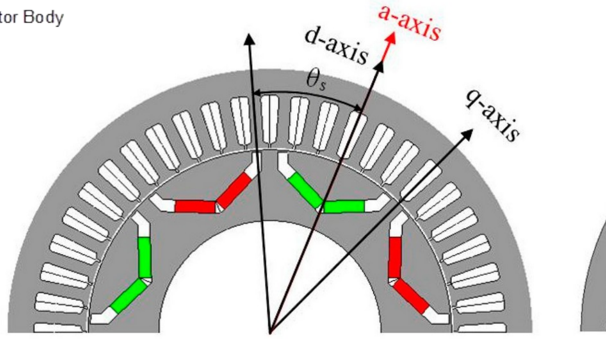
- Brushless motors are three phase synchronous motors that need active commutation.
- Simple controllers exist; do block commutation---behavior similar to brushed motor commutation
  - Usually no current feedback is used
- Better controllers feed sinusoidal currents to the three phases
  - This requires current sensors, well-designed power electronics, and a low latency controller
  - The dominant control method used here is field oriented control, or FOC
- Low inductance motors can yield higher powers at high RPM, but may require a very fast switching inverter, which would require a very fast, low latency control loop.
- Implementing FOC in hardware is useful for miniaturization, smaller motors which running at higher rpms can be quite difficult to control, requiring faster control loops.

FOC overview

- An advanced control strategy for high performance 3-phase brushless motor drives.
- Abstracts the three phase currents into Q and D fields, relative to the rotor position.
- Objective: regulate a set Q and D to their setpoints, which in turn results in magnetic field being applied to the rotor with some amount of phase advanced.
- Torque is primarily produced by the Q axis.
- Depending on the motor geometry, D axis current may also produce reluctance torque.
- Useful for a variety of three phase synchronous motors.

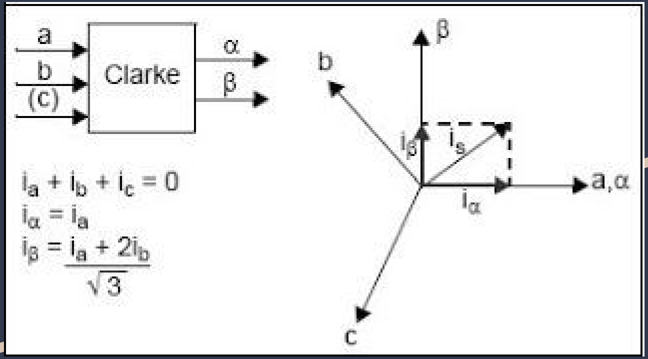






(a)

Clark/Park transforms



- Clark transform: Converts three phase currents to two equivalent orthogonal currents (alpha and beta)
  - Mathematically equivalent but much easier to work with
  - Works very nicely when phase currents sum to zero
- Park transform: rotate alpha and beta currents backwards by the motor phase angle
  - *Direct* and *Quadrature* currents
  - These roughly correspond to field strength inside the motor and torque
- Inverse Clark/Park transforms exist to convert D/Q commands to three phase motor commands

Space Vector PWM

- PWM switching method optimized for driving three phase motors and minimizing switching
- Uses the fact that holding all phases high is equivalent to holding all phases low

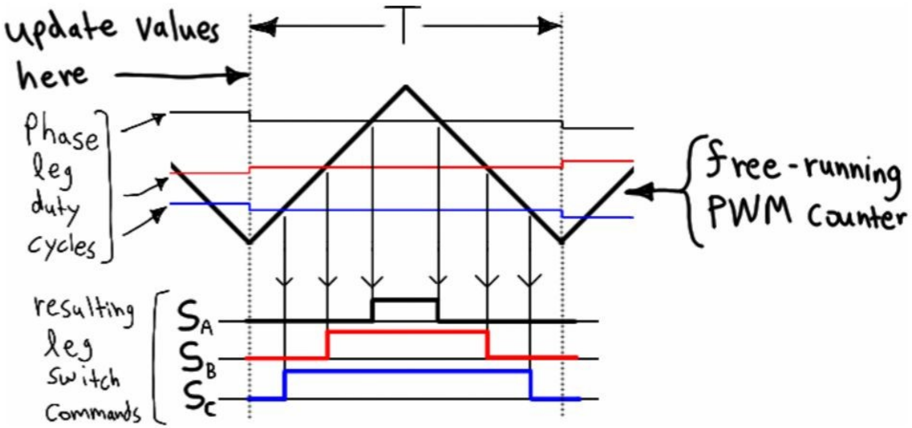
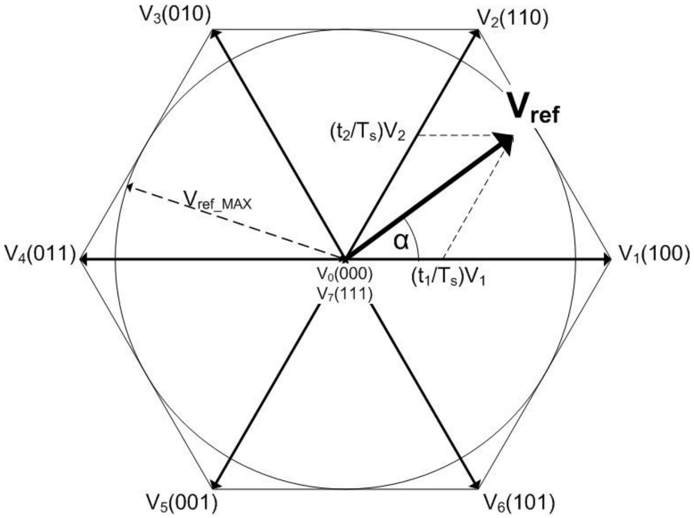
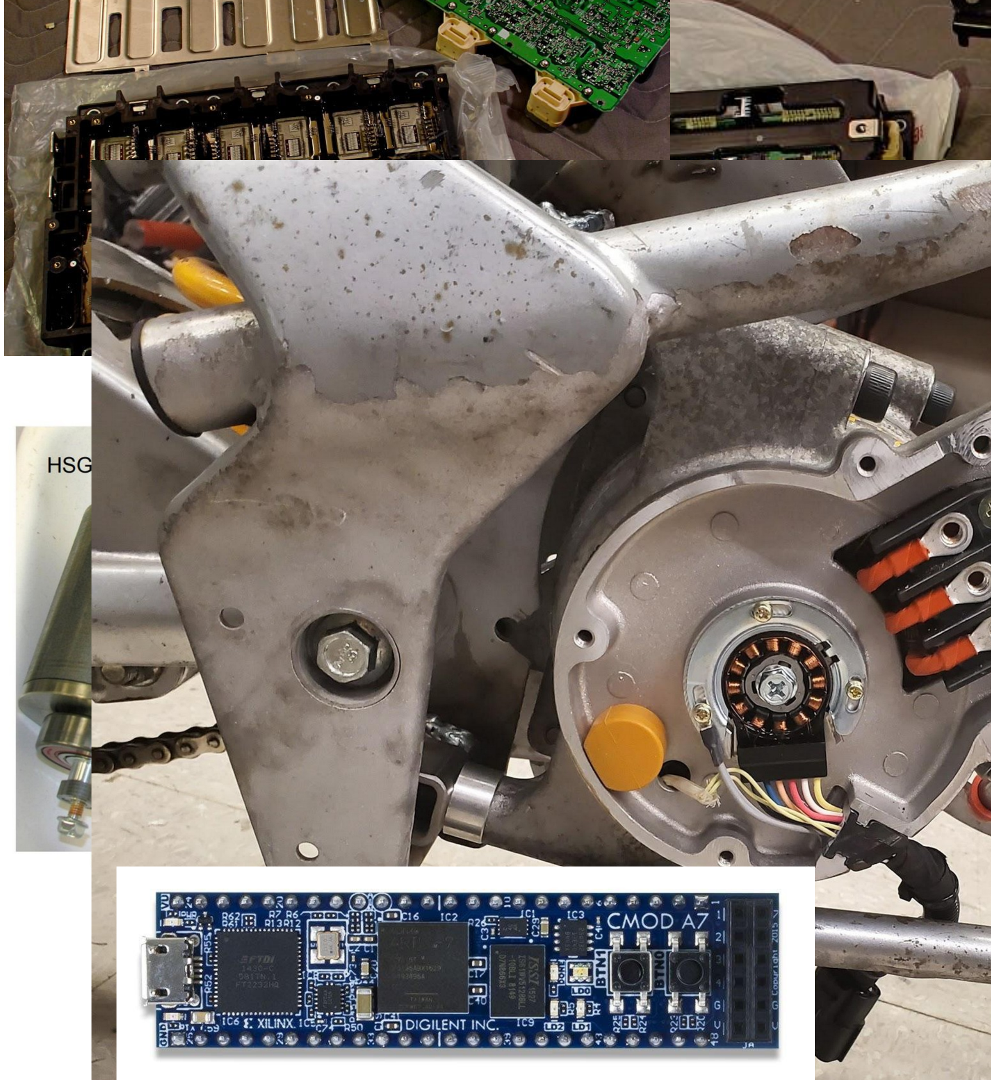


Figure 4.41 – Generating leg switch commands.

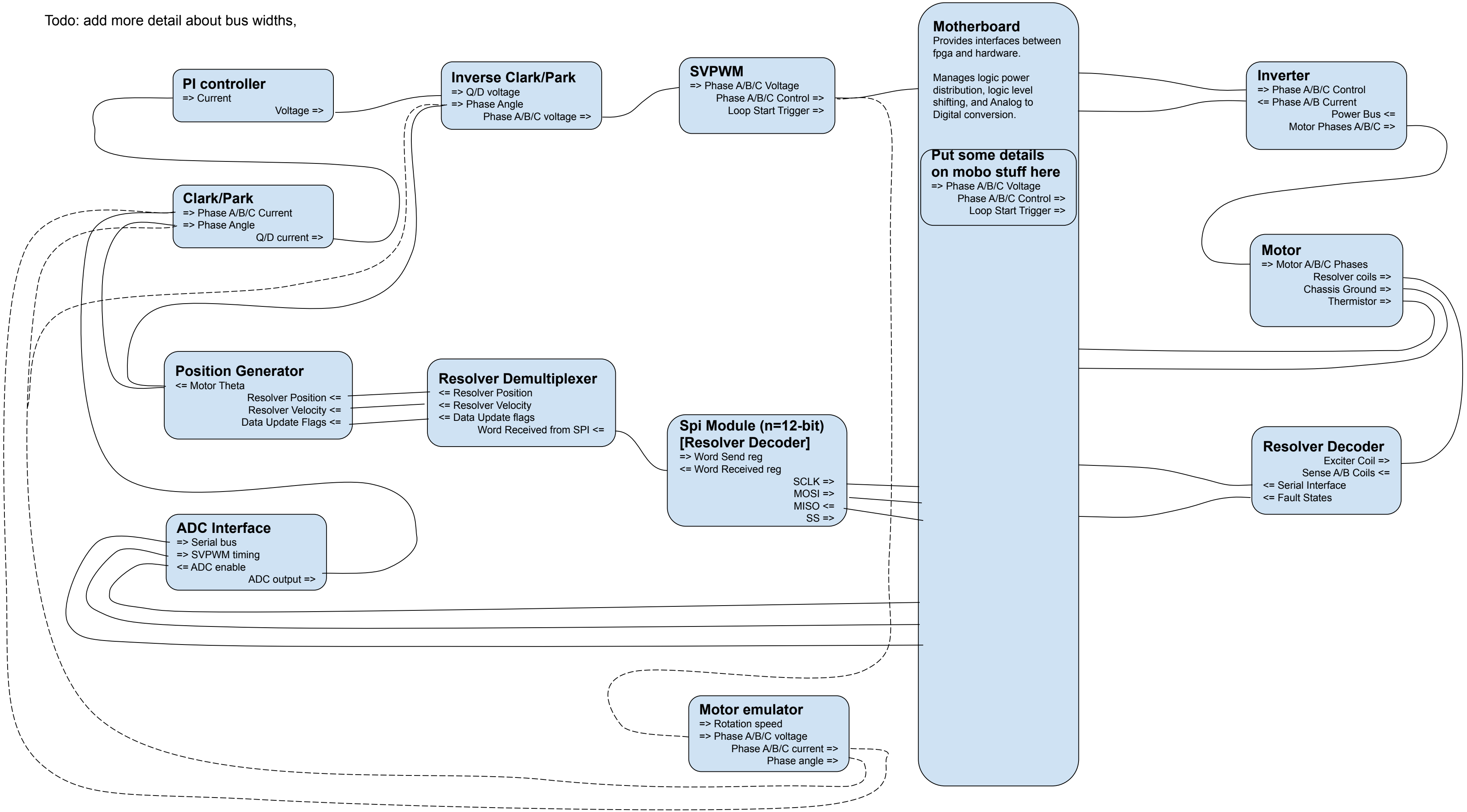


Hardware

- Inverter: Gen 2 Prius inverter module
  - 500V and 600 total phase amps
  - IGBT based
  - Fully isolated gate drive and current sense amplification
- Motor: Hyundai sonata Hybrid Starter-Generator (HSG) motor
  - 115 N-m, 40 kW
  - ~15,000 RPM
  - Integrated variable reluctance resolver
- FPGA: CMOD A7-35T
  - A miniature Artix-7 Dev board
- Our hardware:
  - Motherboard to interface the inverter with an FPGA.
  - Resolver decoder board, to excite and observe the resolver.



Todo: add more detail about bus widths,



Goals

- Base goal
  - Simulated motor model works as desired
- Minimum viable product
  - A motor controller that works with a real motor
- Stretch goal(s) [pick one or more]
  - %Max-Torque to D/Q mapping table
  - Sensorless control
  - Servo control
  - USB interface

Schedule

- 11/11-11/18
  - Implement FOC control loop blocks and testbenches
  - Finish motherboard PCB design
  - Finish characterizing Prius inverter block
- 11/18-11/25
  - Send out PCB’s and order parts
  - Write motor simulator testbench
  - Put blocks together and simulate the motor
  - Write serial interface blocks for ADCs/resolver
  - Assemble the motherboard and resolver boar test hardware.
  - Start testing with real motor
- 11/25-12/2
  - Test current loop, command Q or D currents o stationary motor.
  - Integrate resolver feedback; get the real moto spin.
  - Observe strange bugs and misbehaviors, atte to fix them.
- 12/2-12/7
  - Start implementing and testing Bonus feature
  - Attempt to tidy up the physical packaging, mo onto vehicle, achieve record for fastest 6.111 project.
  - Checkoffs.
  - Demos