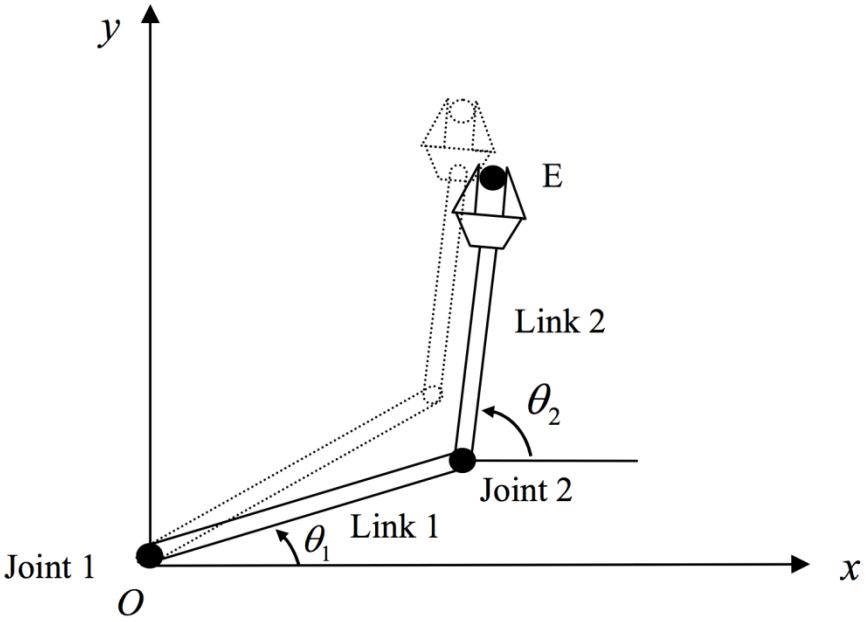
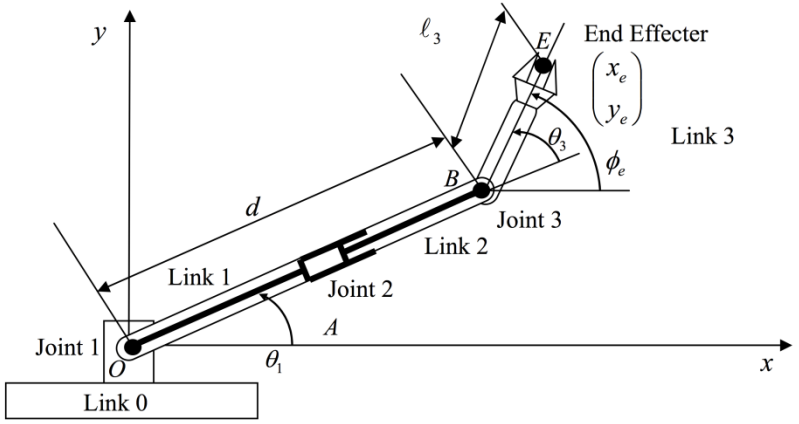
A Core Robot Algorithm: Inverse Kinematics





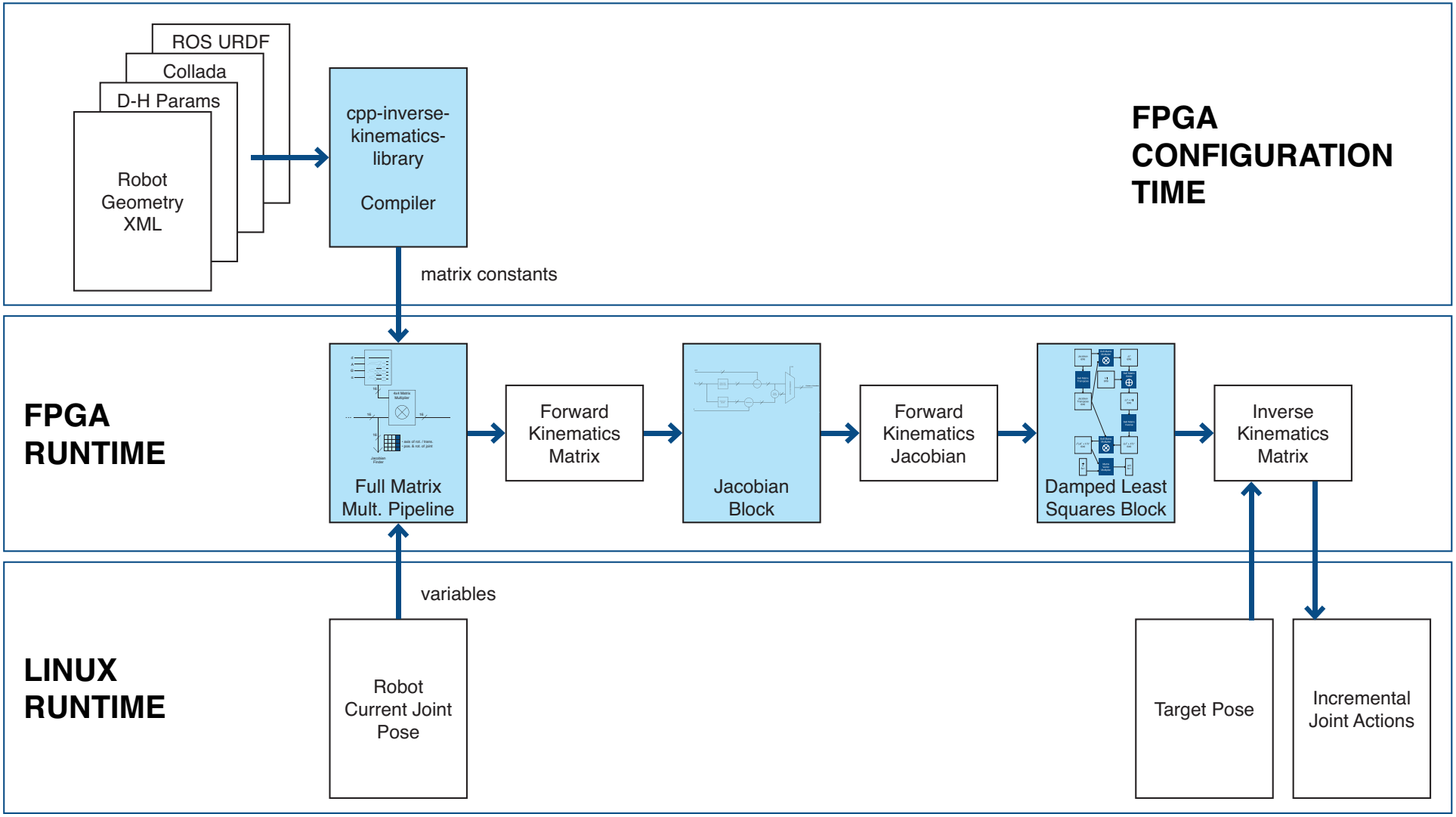
- Setting a robot's joints so end effector reaches a target
  - Input: current robot geometry
  - Output: required joint increments
- Computationally intensive problem all limbed robots must solve
- Beyond controlling single arms and legs, many larger problems rely on inverse kinematics
  - redundant manipulators
  - multiple end effectors
  - inverse dynamics

A Digital Accelerator for Inverse Kinematics

- Inverse kinematics not well suited for normal digital architectures
  - Entirely floating point array, matrix operations
  - 40% of cycles in inverting matrices
  - 15% of cycles in sine, cosine operations

- We solve IK via damped least squares
  - Dedicated sine, cosine function generators
  - Parallel, fixed-point functional units
  - Solves IK problem in 100μs: compare against 10ms for general algorithm on CPU

Architecture and Toolchain



Architecture and Timing Design

- Architectural Choices
  - Pipelining sine/cosine and array multiply
  - Parallelized matrix multiply and matrix inversion
  - Fixed point representations throughout system

- Timing Choices
  - Single array of multipliers shared amongst modules
  - Aggregate individual enable and done signals into global state machine

Experiences and Issues

- Deciding on the algorithm to use
- Determining what implementation would fit on the board
- Convincing ourselves the algorithm works
- Extensively tested the core hardware, but not the top-level interface (until yesterday)

Lessons Learned

- Test the whole stack earlier
- Plan before trying to implement
- Use timing diagrams and area estimates before touching hardware
- Leave no ambiguity in the design