

MATLAB EXPO 2017

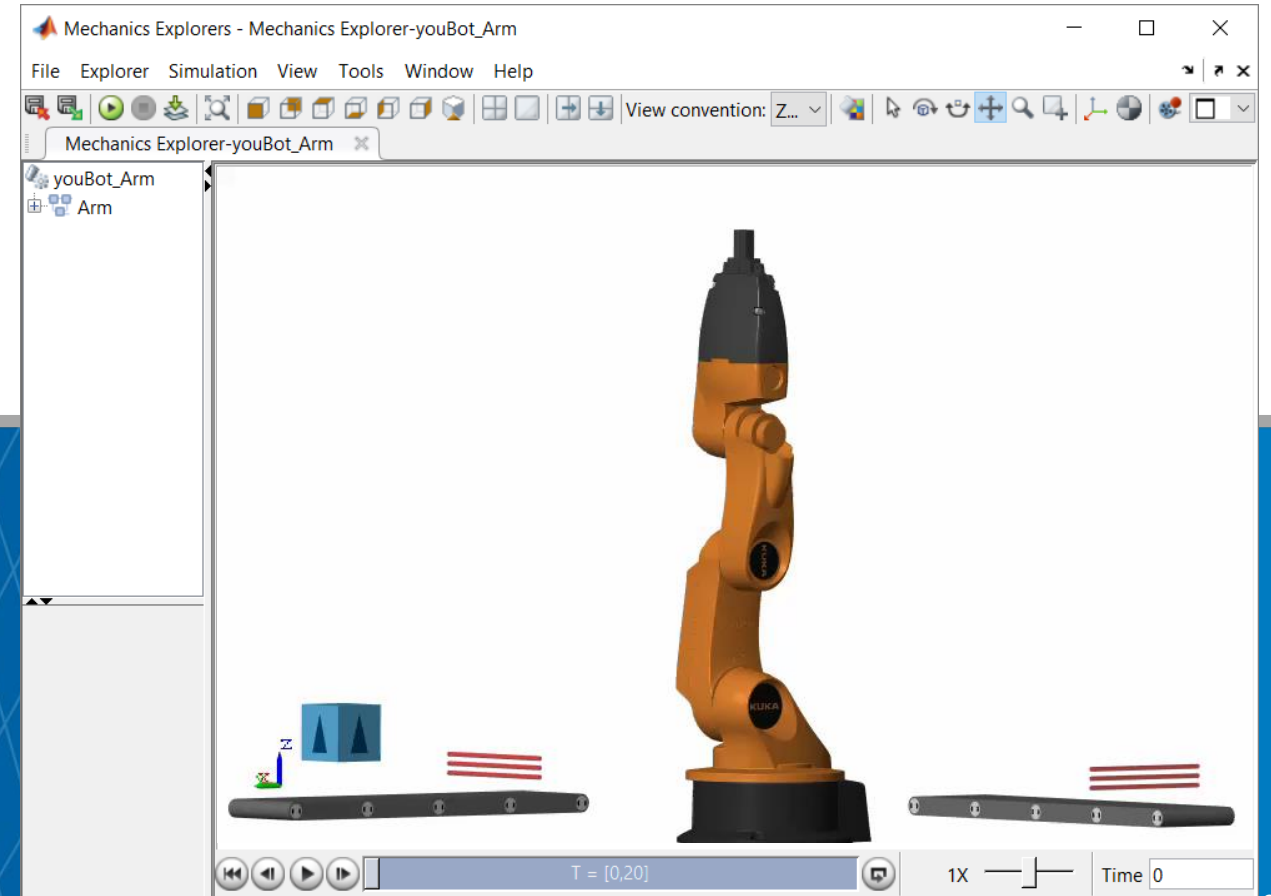
KOREA

4월 27일, 서울

등록 하기 matlabexpo.co.kr

Integrating Mechanical Design and Multidomain Simulation with Simscape

강효석 과장 / Ph. D.
 Application Engineer
 MathWorks Korea



In this session

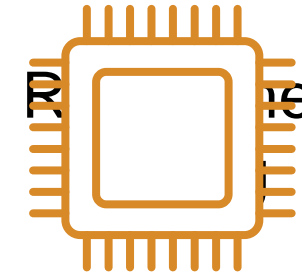
- Onshape and MATLAB enable engineers to combine CAD models with multidomain, dynamic simulation

MATLAB

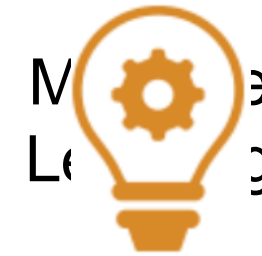
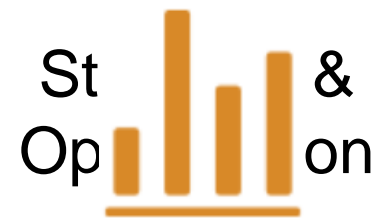
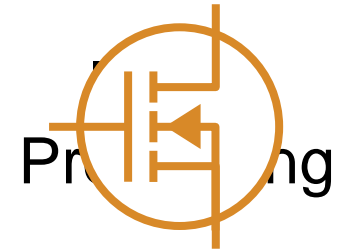
In this session

- Onshape and MATLAB enable engineers to combine CAD models with multidomain, dynamic simulation

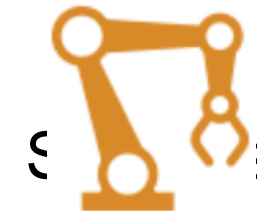
MATLAB



Simulink



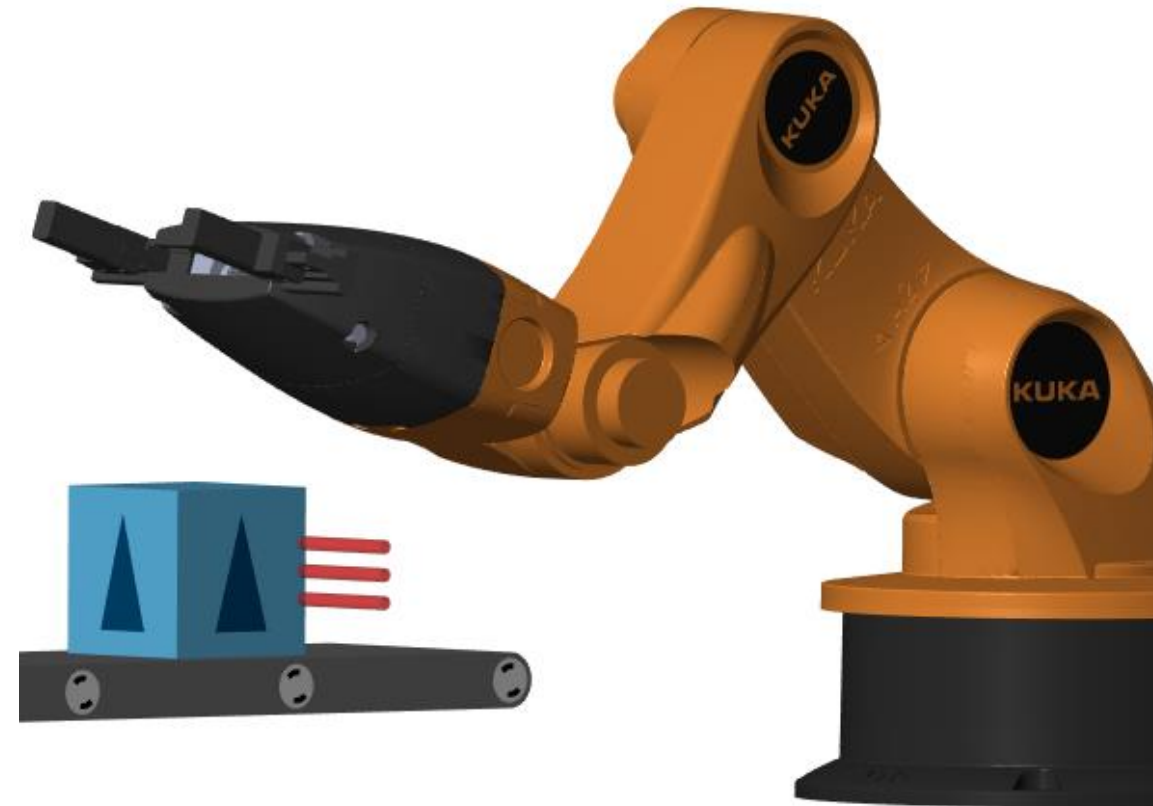
Simscape



Stateflow

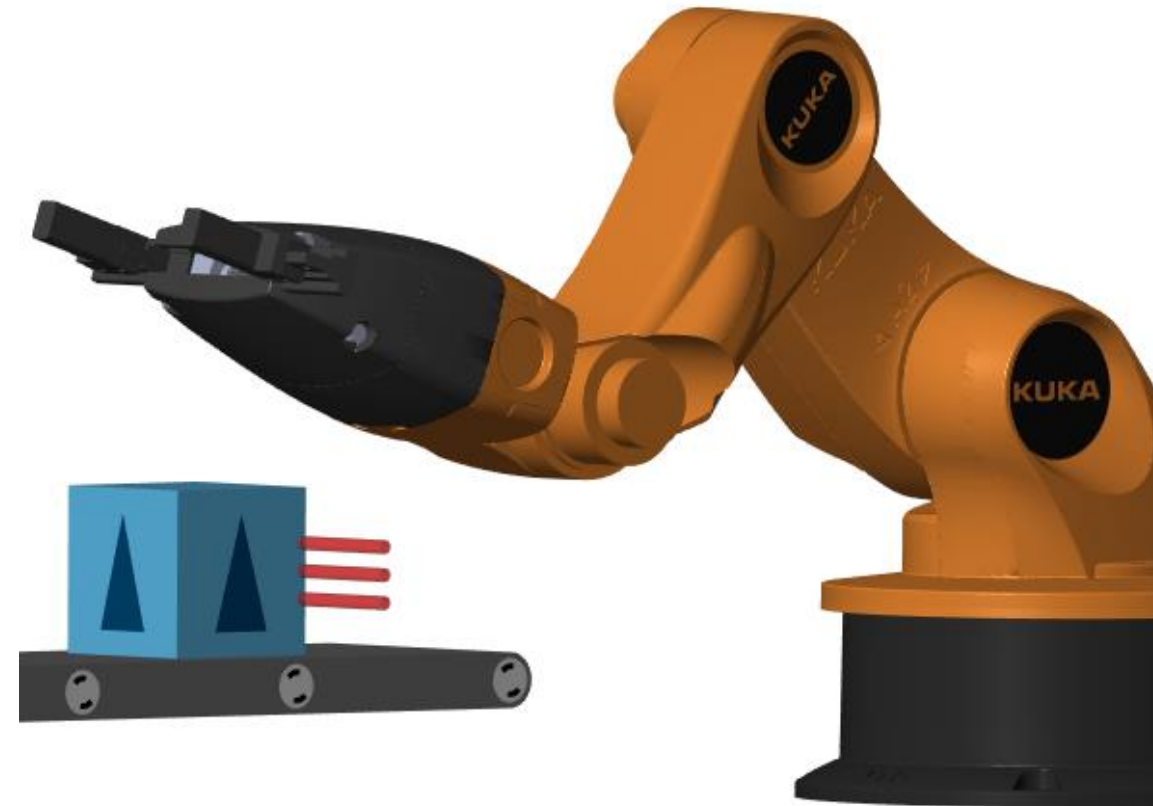
In this session:

- Onshape and MATLAB enable engineers to combine CAD models with multidomain, dynamic simulation
- Results you can achieve:
 1. Optimized mechatronic systems
 2. Improved quality of overall system
 3. Shortened development cycle



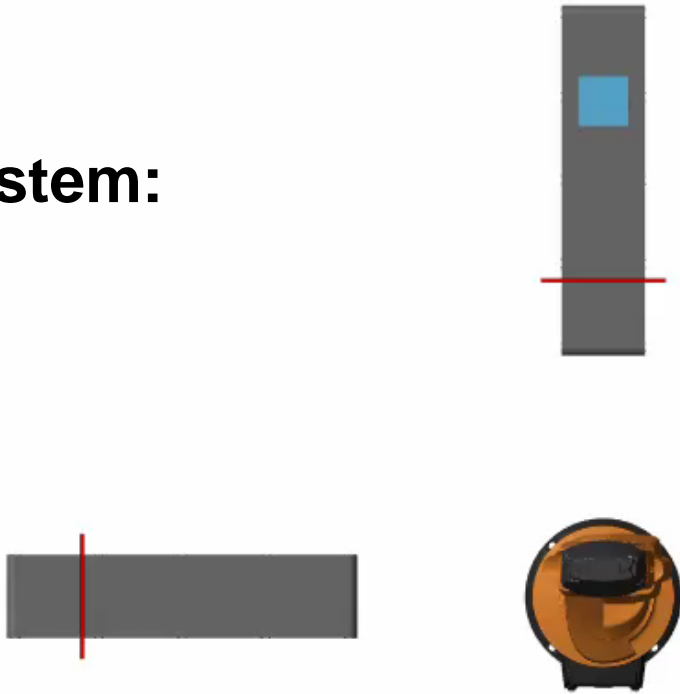
Why Combine CAD and Multidomain, Dynamic Simulation?

- **Fewer iterations** on mechanical design because requirements are refined
- **Fewer mechanical prototypes** because mistakes are caught earlier
- **Reduced system cost** because components are not oversized
- **Less system downtime** because system is debugged using virtual commissioning



Design Challenge

System:

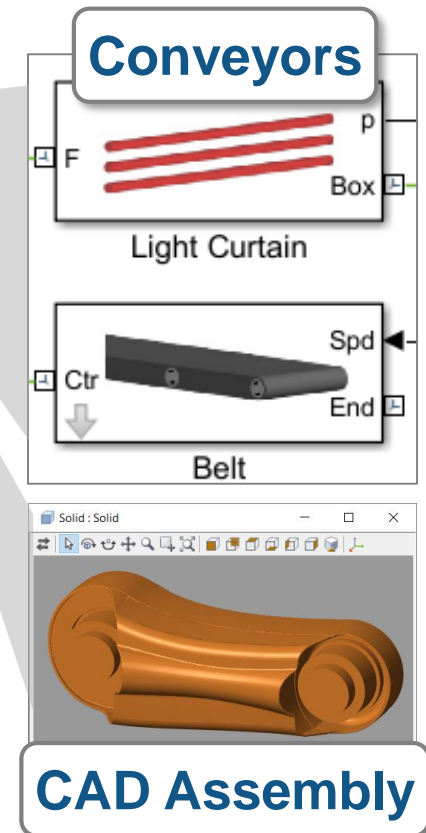
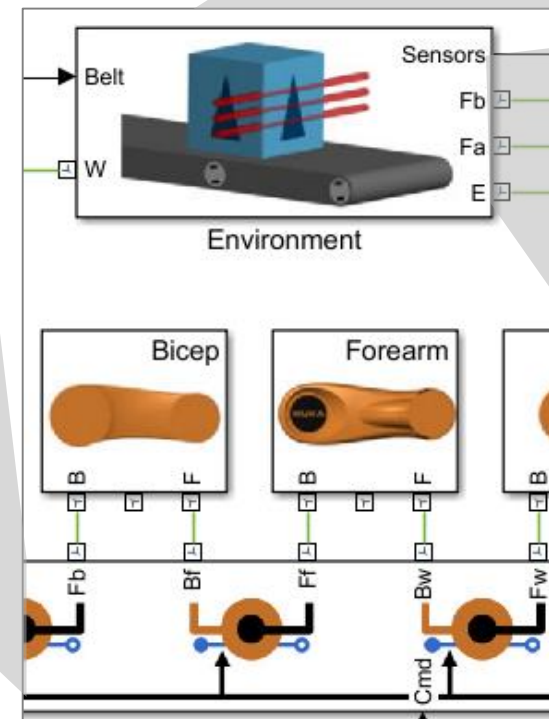
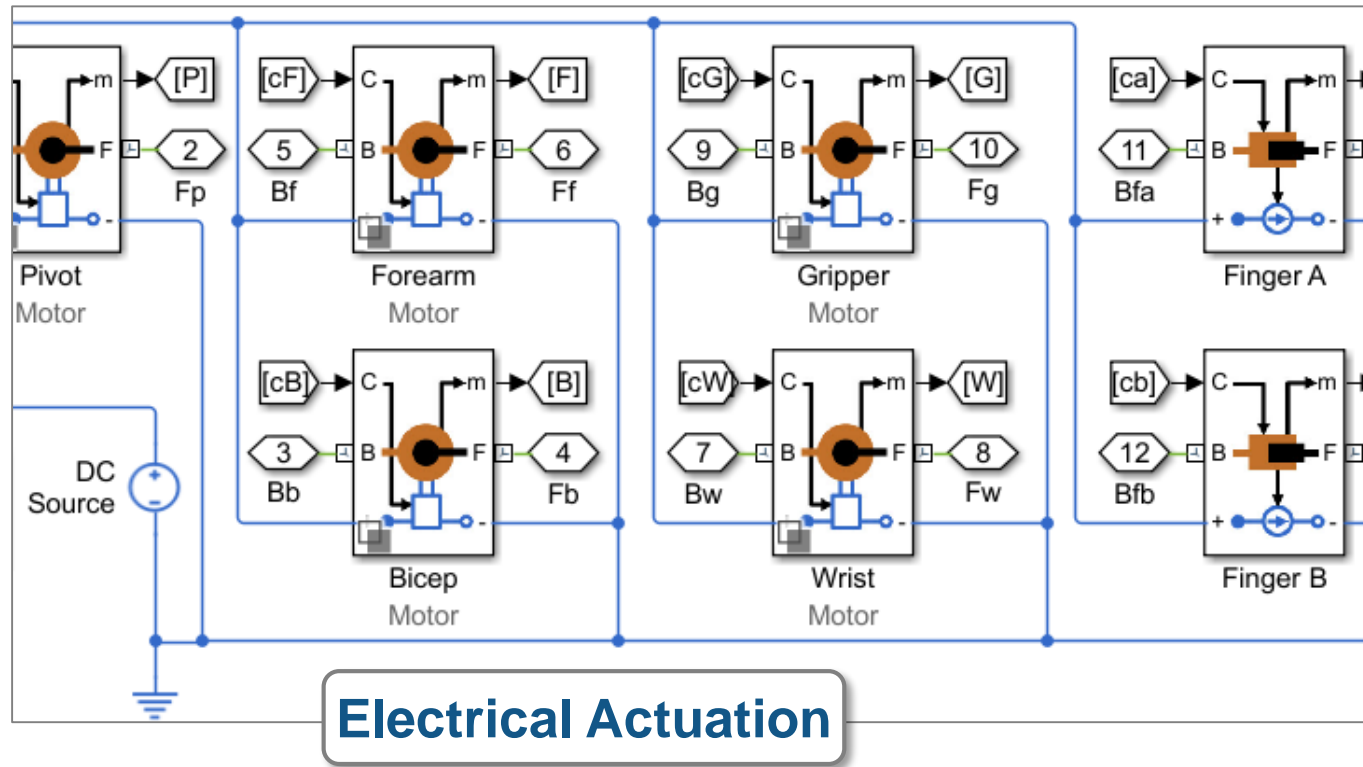
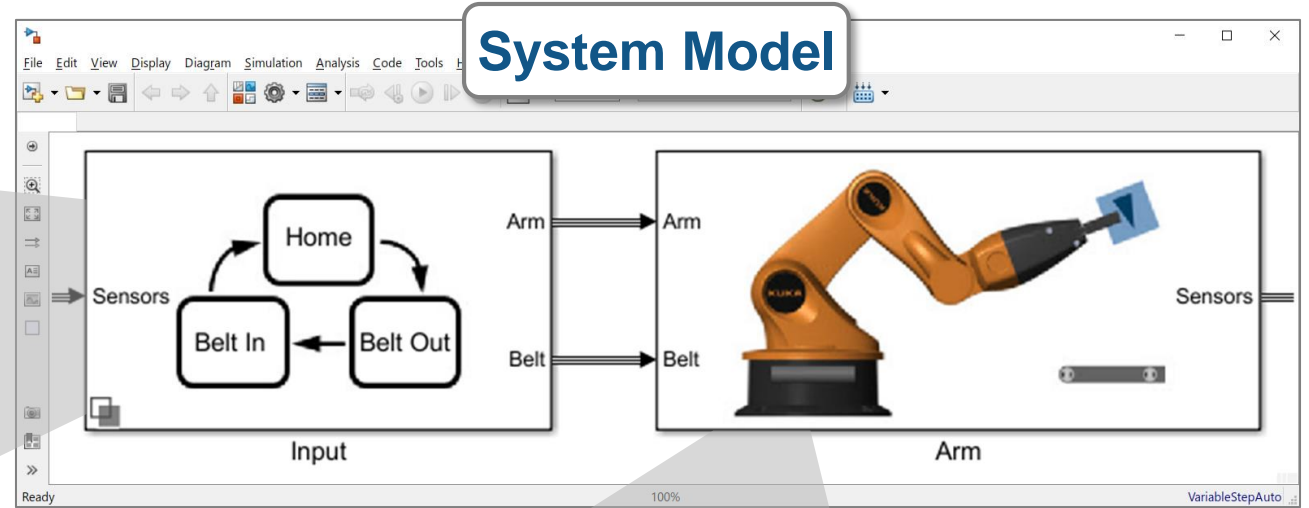
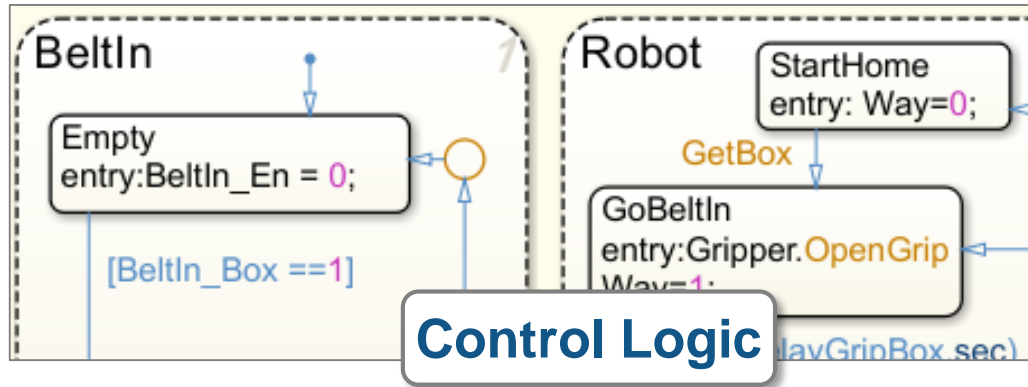


1. Import Onshape Model
2. Determine Motor Requirements
3. Integrate Electrical Actuators
4. Minimize Power Consumption
5. Develop Control Logic

Challenge: Select motors and define controls for robot and conveyor belts.

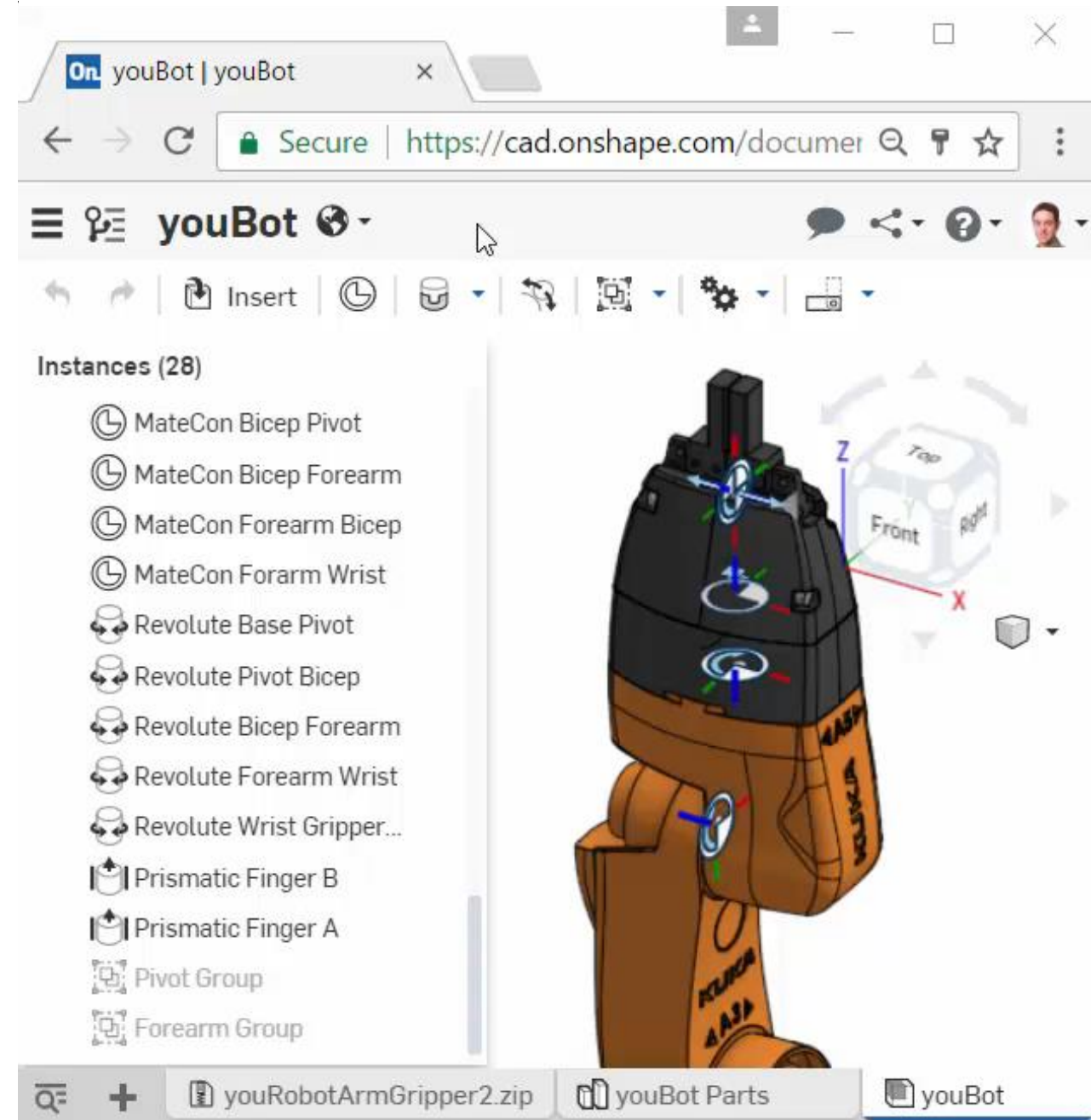
Solution: Import Onshape model into Simscape; use simulation to define actuator requirements and control logic

System Model



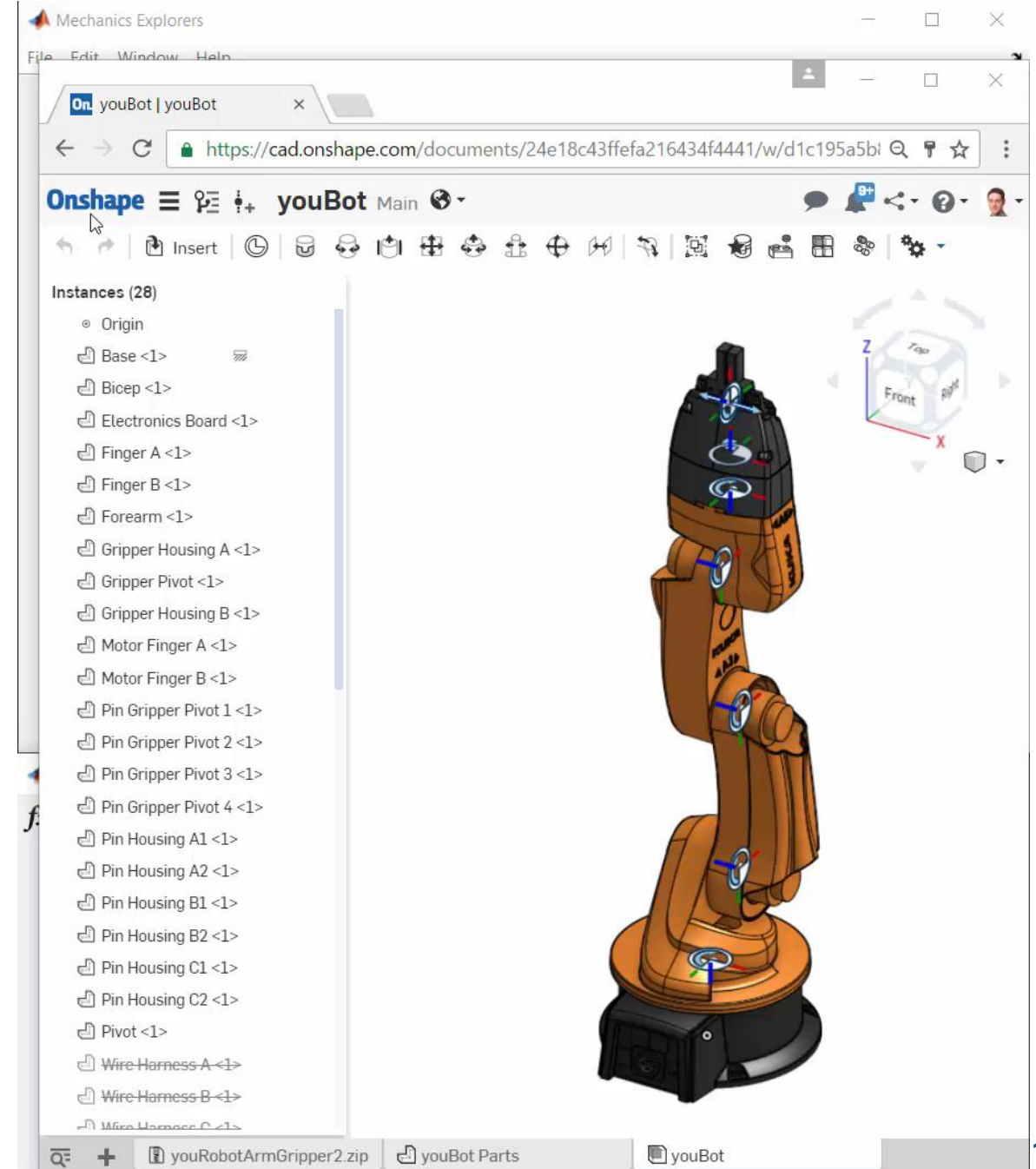
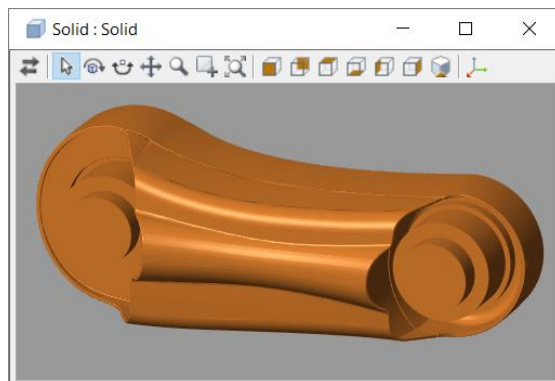
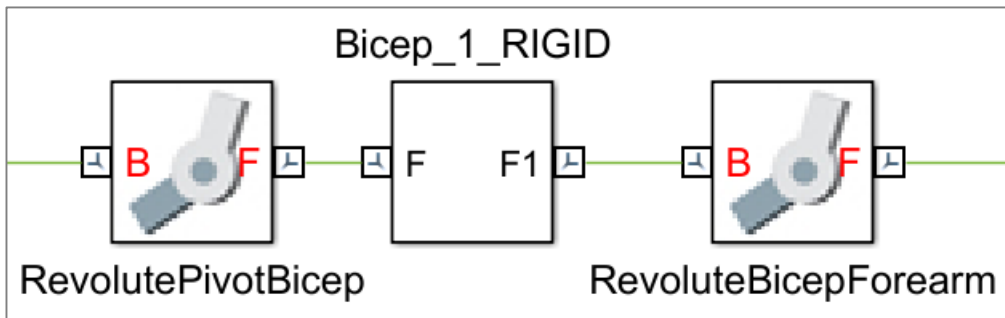
Kuka Robot

- 5 degrees of freedom, and a gripper
- Key advantage of Onshape: Ability to directly define joints
 - Exact mapping to constraints used in multibody simulation
- System engineer reuses mechanical design in dynamic simulation



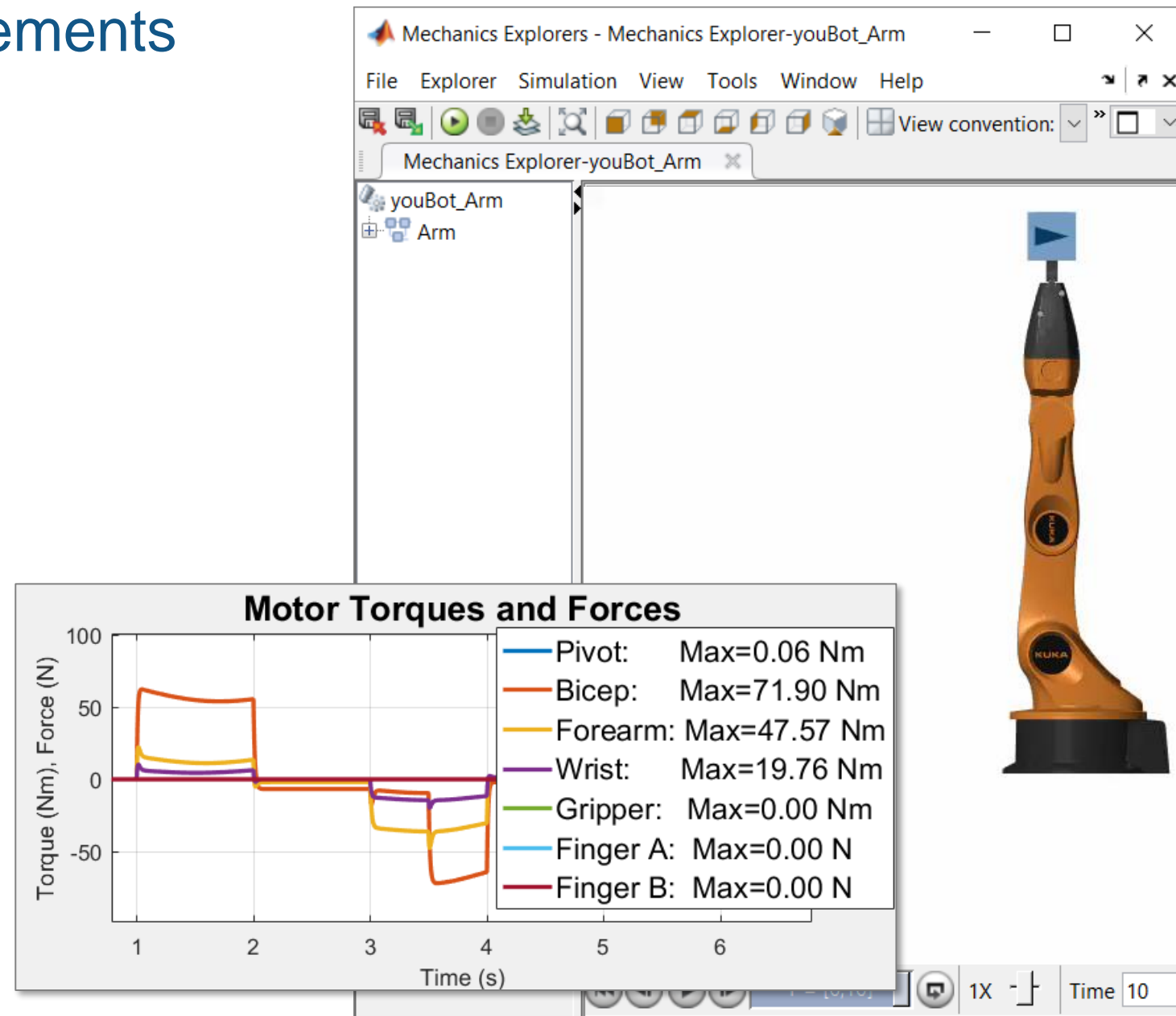
1. Import Model from Onshape

- Convert CAD assembly to dynamic simulation model for use within Simulink
 - Mass, inertia, geometry, and joints



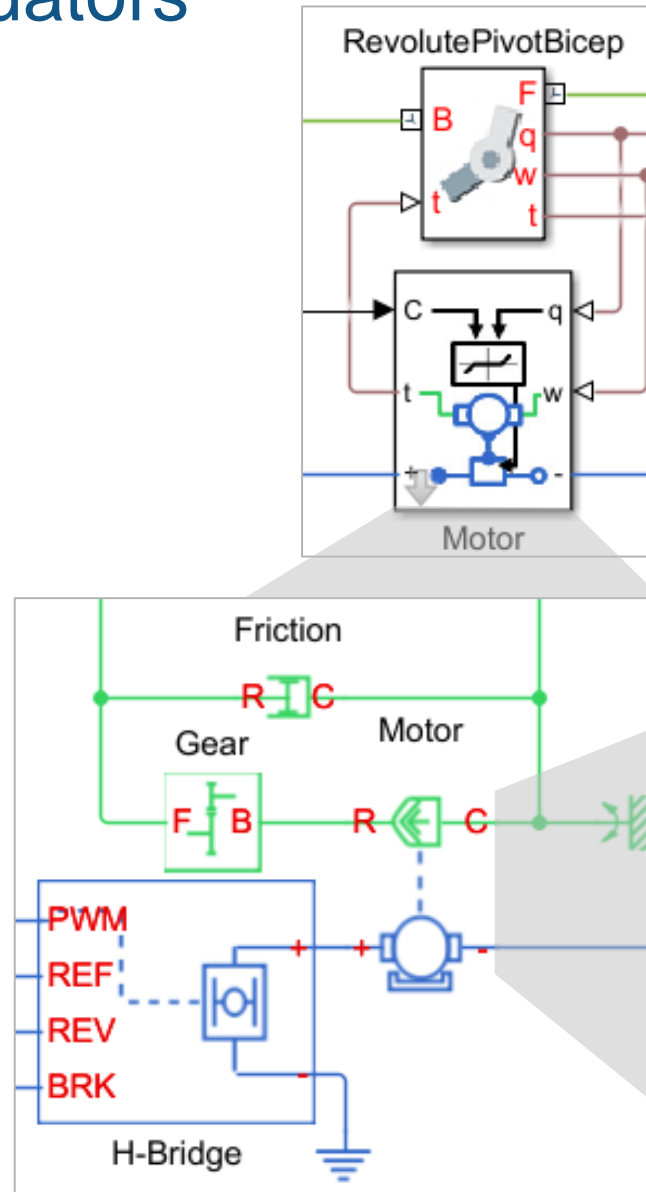
2. Determine Motor Requirements

- Define and run a set of tests
 - Maximum payload, speed
 - Worst case friction levels
 - Full range of movement
- Use dynamic simulations to calculate required torque and bearing forces
- If design changes, automatically rerun tests and re-evaluate results



3. Integrate Electrical Actuators

- Add motors, drive circuitry, gears, and friction
- Choose motors based on torque requirements
- Assign parameters directly from data sheets



Motor Data

251601

Characteristics

Terminal resistance	Ω	0.978
Terminal inductance	mH	0.573
Torque constant	mNm / A	33.5
Speed constant	rpm / V	285
Speed / torque gradient	rpm / mNm	8.32
Mechanical time constant	ms	11.8
Rotor inertia	gcm ²	135

Electrical Torque

Mechanical

Model parameterization: Circuit parameters

Armature resistance: 0.978 Ohm

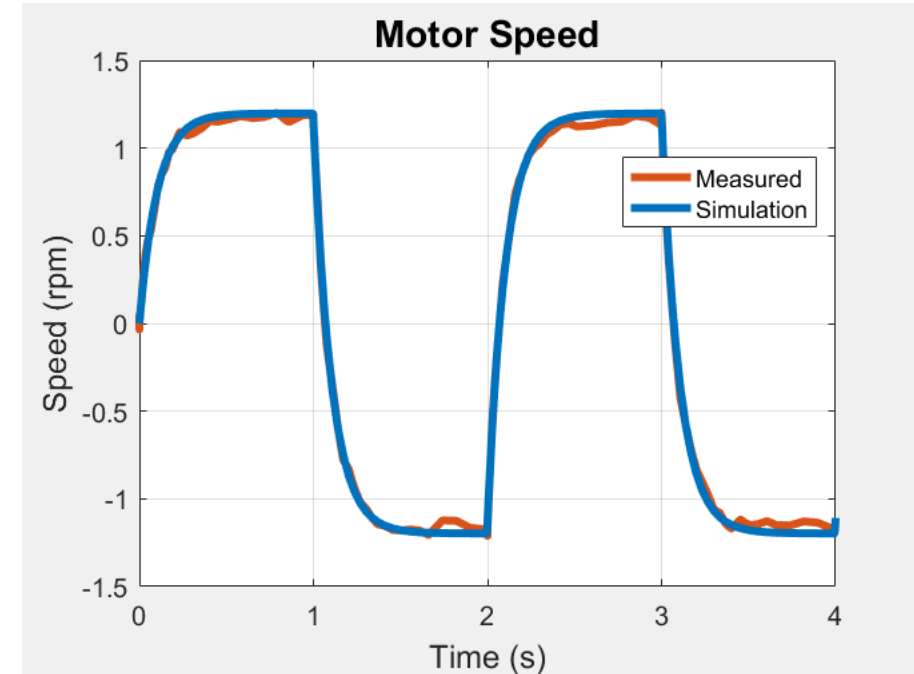
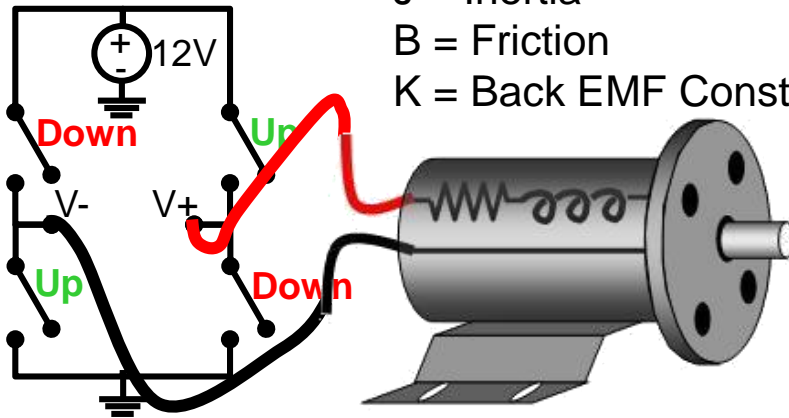
Armature inductance: 0.573 mH

Torque constant: 33.5 mN*m/A

Estimating Parameters Using Measured Data

Model:

R = Resistance
 L = Inductance
 J = Inertia
 B = Friction
 K = Back EMF Constant



Problem: Simulation results do not match measured data because model parameters are incorrect

Solution: Use [Simulink Design Optimization](#) to automatically tune model parameters

R	L	J	K	B
4.03	1e-4	0.11	0.45	1.07

Estimating Parameters Using Measured Data

- Steps to Estimating Parameters

1. Import measurement data and select estimation data
2. Identify parameters and their ranges
3. Perform parameter estimation

Edit Experiment: MeasuredData

Outputs
Specify measured output signals for this experiment.
[../PS-Simulink Converter:1\)](#)
 <1x1 Signal, 291 points>

Select Measured Output Signals

Inputs
Optionally specify input signals for this experiment.
[../Input Signal \(V\):1\)](#)
 <2x1 Signal, 525 points>

Select Inputs

Edit: Estimated Parameters

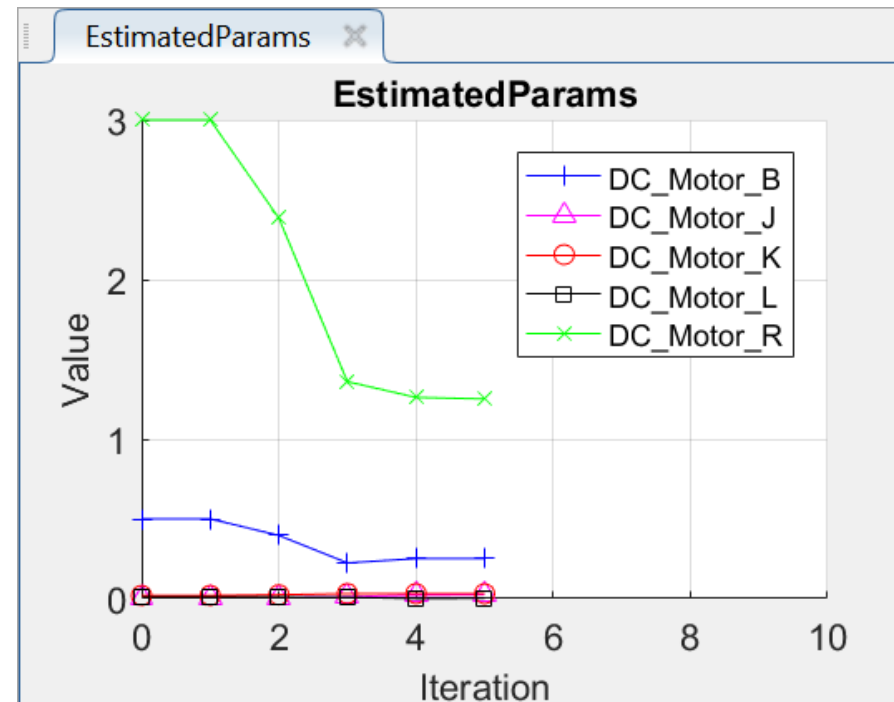
Parameters Tuned for all Experiments

[DC_Motor_B](#)
 0.5
 Minimum: 0.01
 Maximum: Inf
 Scale: 0.5

[DC_Motor_J](#)
 0.01

[DC_Motor_K](#)
 0.02

[DC_Motor_L](#)
 0.01



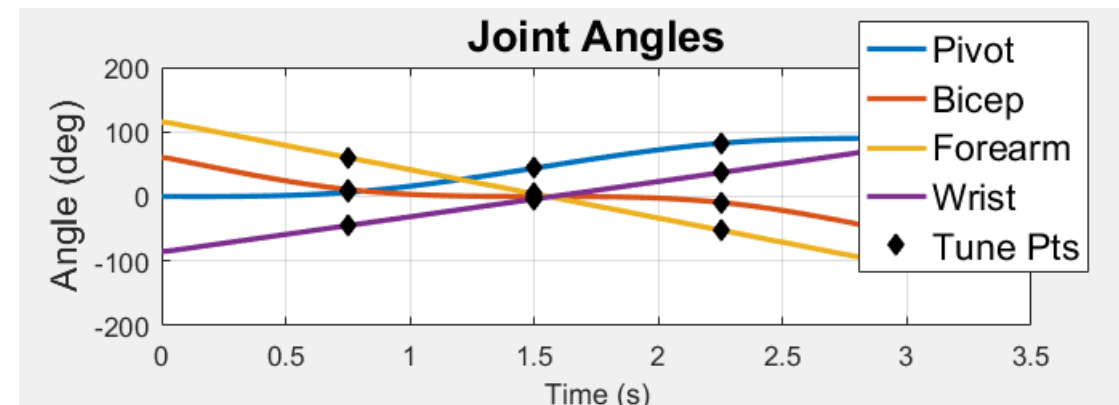
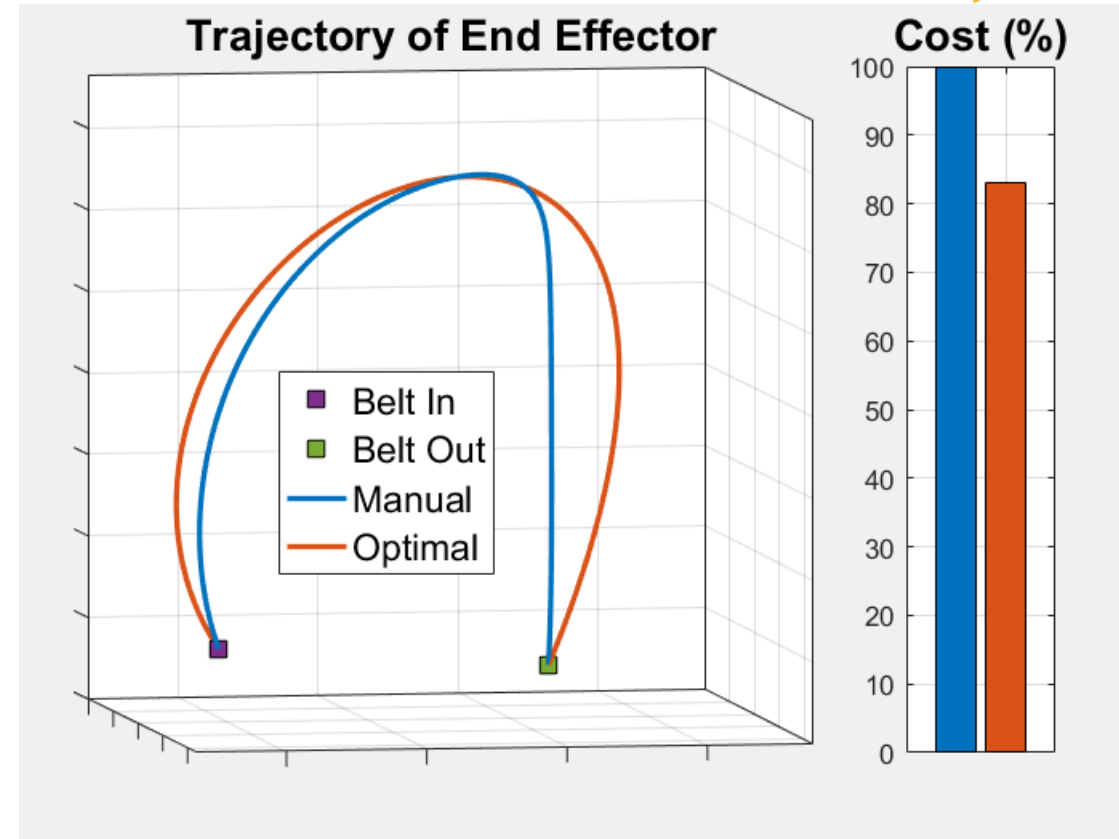
4. Minimize Power Consumption

Model:

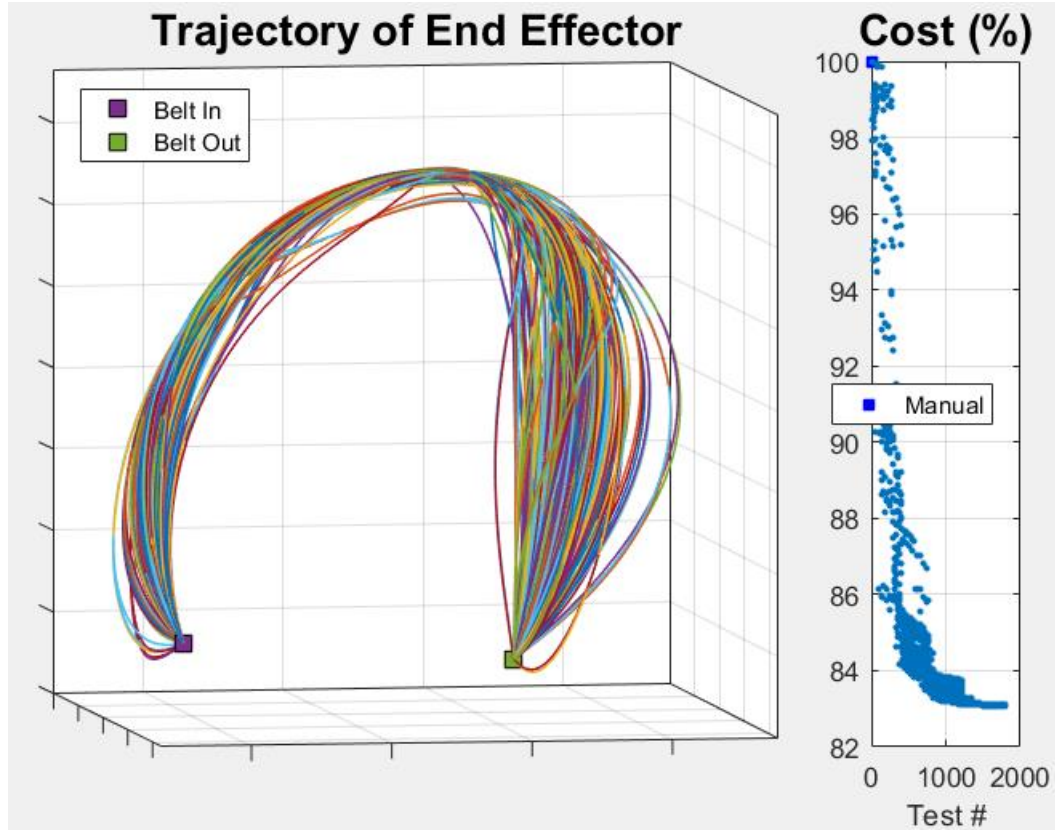


Challenge: Identify arm trajectory that minimizes power consumption.

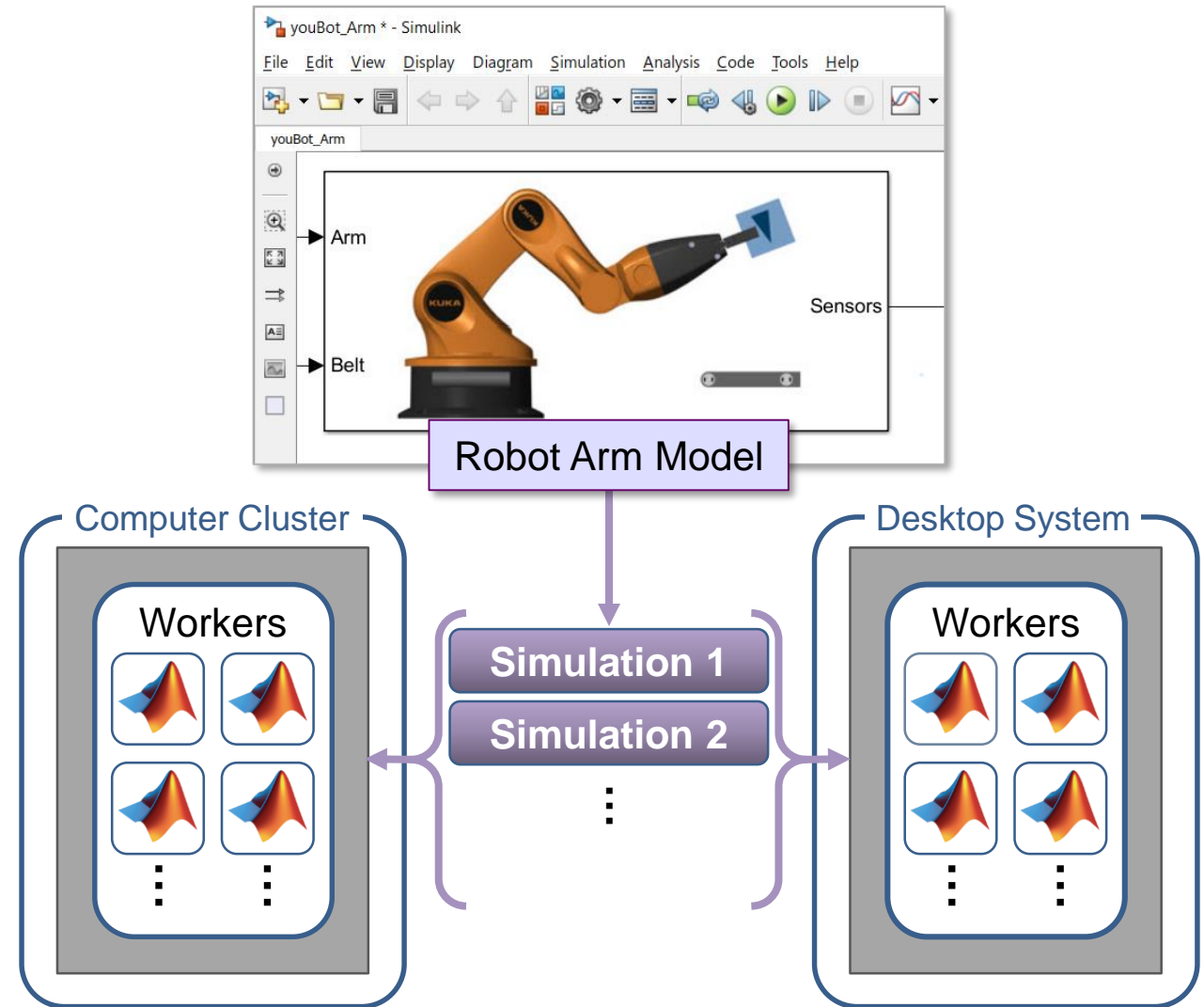
Solution: Use dynamic simulation to calculate power consumption, and use optimization algorithms to tune trajectory.



Accelerate Design Iterations Using Parallel Computing



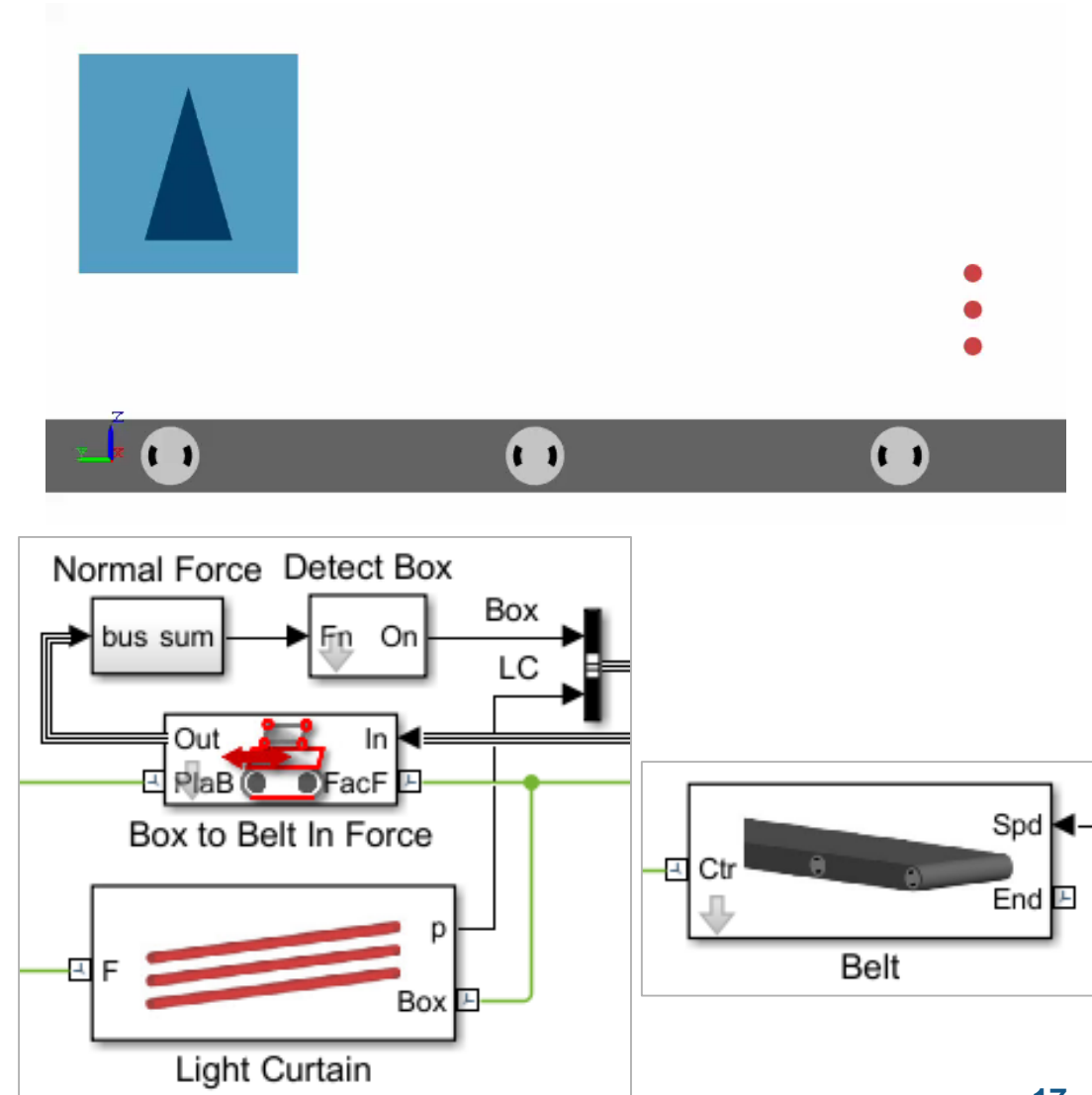
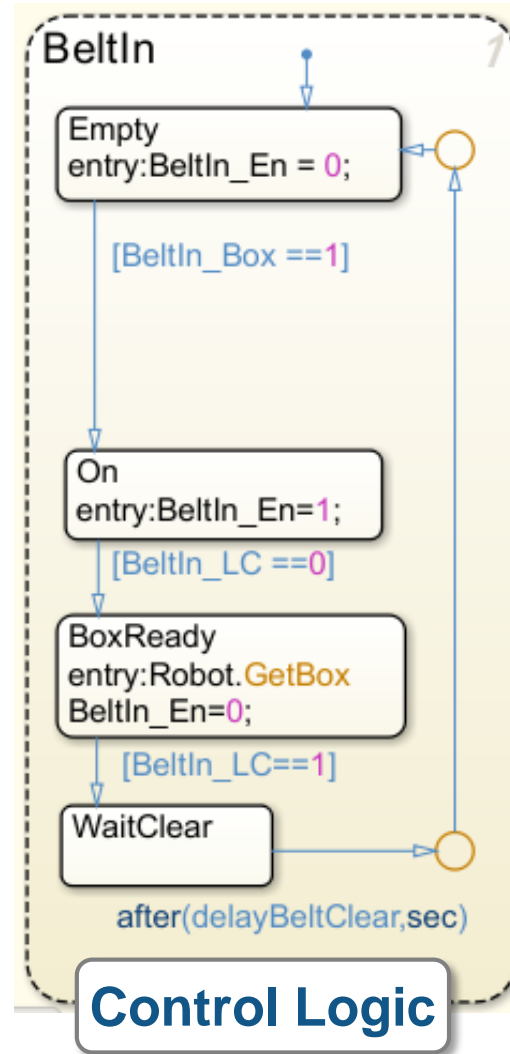
This optimization task required nearly 2000 simulations.



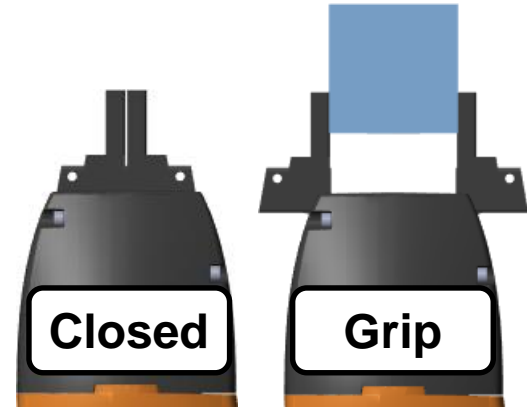
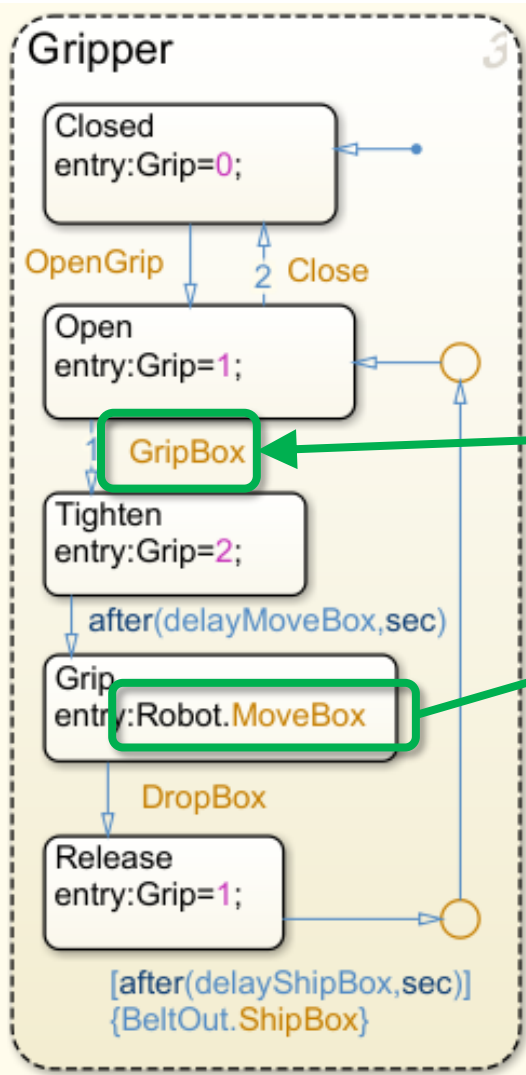
Running simulations in parallel speeds up your testing process.

5. Design Control Logic for Arm and Conveyor Belts

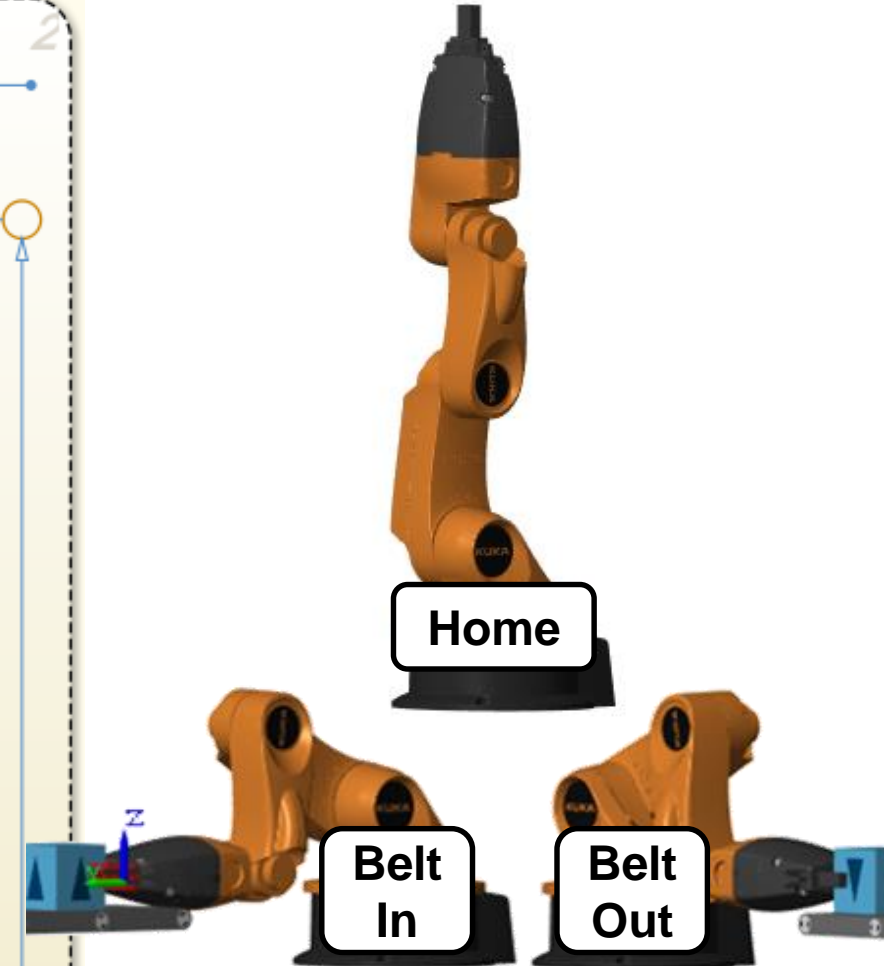
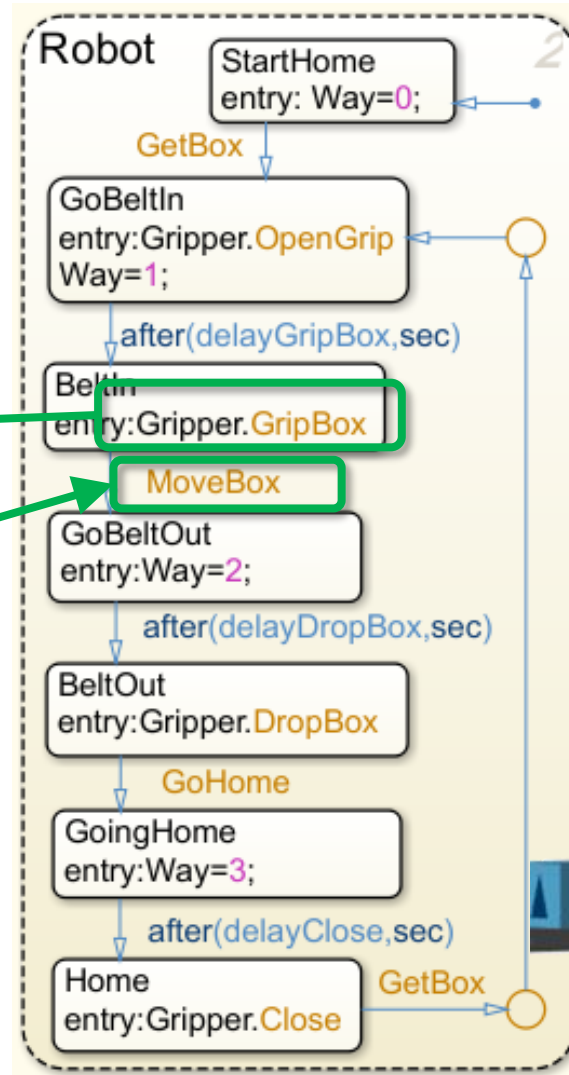
- Sense quantities within model that govern system events
- Design logic using a state chart
- Use outputs of logic to control models of system components



5. Design Control Logic for Arm and Conveyor Belts



State charts depend on each other



5. Design Control Logic for Arm and Conveyor Belts

Stateflow (chart) youBot_Arm/Input/Control/Logic* - Simulink

File Edit View Display Chart Simulation Analysis Code Tools Help

Logic

BeltIn

- Empty entry: BeltIn_En = 0;
- [BeltIn_Box == 1]
- On entry: BeltIn_En = 1;
- [BeltIn_LC == 0]
- BoxReady entry: Robot.GetBox; BeltIn_En = 0;
- [BeltIn_LC == 1]
- WaitClear after(delayBeltClear, sec)

Robot

- StartHome entry: Way = 0;
- GetBox
- GoBeltIn entry: Gripper.OpenGrip; Way = 1;
- after(delayGripBox, sec)
- BeltIn entry: Gripper.GripBox
- MoveBox
- GoBeltOut entry: Way = 2;
- after(delayDropBox, sec)
- BeltOut entry: Gripper.DropBox
- GoHome
- GoingHome entry: Way = 3;
- after(delayClose, sec)
- Home entry: Gripper.Close
- GetBox

Gripper

- Closed entry: Grip = 0;
- OpenGrip
- Open entry: Grip = 1;
- Close
- GripBox
- Tighten entry: Grip = 2;
- after(delayMoveBox, sec)
- Grip entry: Robot.MoveBox
- DropBox
- Release entry: Grip = 1;
- [after(delayShipBox, sec)] {BeltOut.ShipBox}

BeltOut

- Empty entry: BeltOut_En = 0;
- [BeltOut_Box == 1]
- WaitRelease
- ShipBox
- On entry: BeltOut_En = 1;
- [BeltOut_LC == 0]
- BoxReady entry: Robot.GoHome; BeltOut_En = 0;
- [BeltOut_LC == 1]
- WaitClear after(delayBeltClear, sec)

Running 100% ode15s

Mechanics Explorers - Mechanics Explorer-youBot_Arm

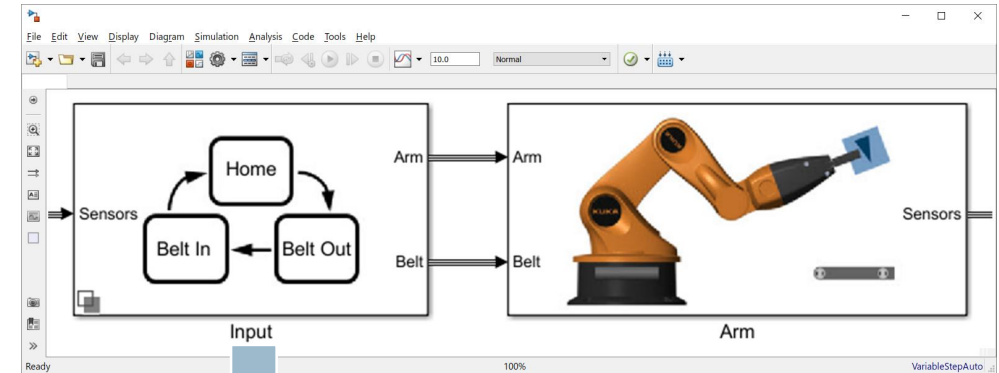
File Explorer Simulation View Tools Window Help

Mechanics Explorer-youBot_Arm

0% 1X Time 0

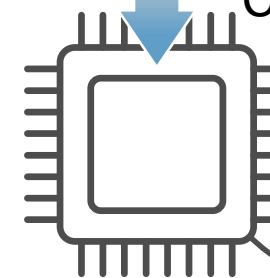
Test Production Control Software

- Automatically convert algorithms to production code
 - C Code, PLC Code
- Incrementally test the effect of each conversion step
 - Fixed-point math
 - Latency on production controller
- Use the same plant model
 - Test without expensive hardware prototypes



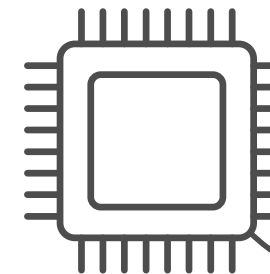
Processor-in-the-Loop (PIL)

Convert to
C Code



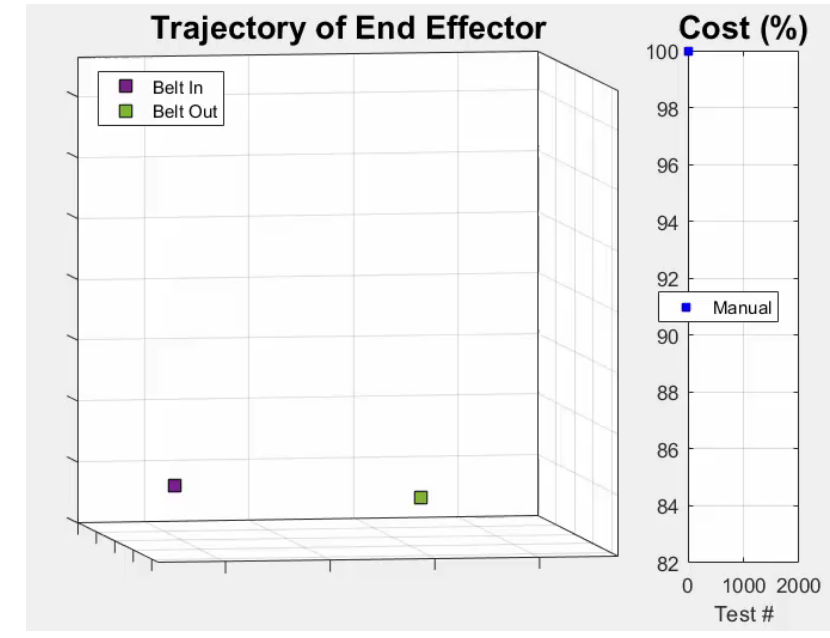
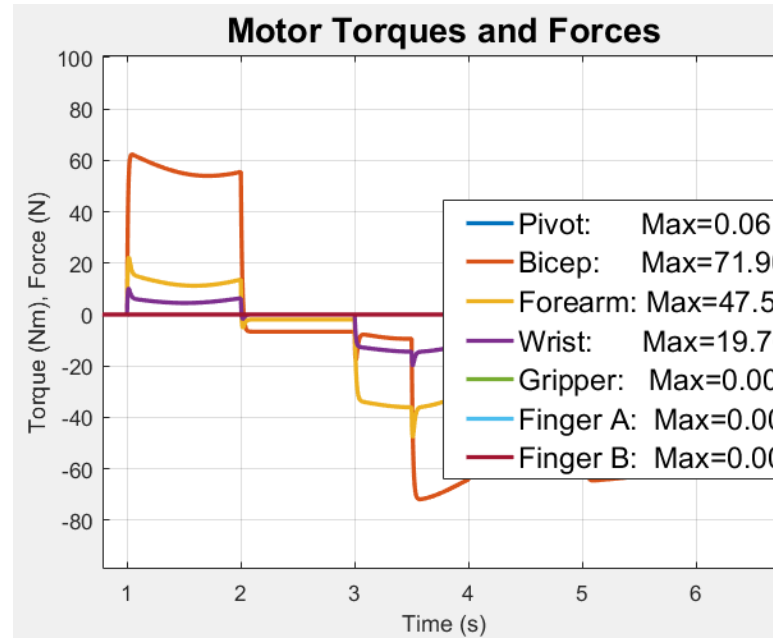
Hardware-in-the-Loop (HIL)

Convert to
C Code



What we have shown

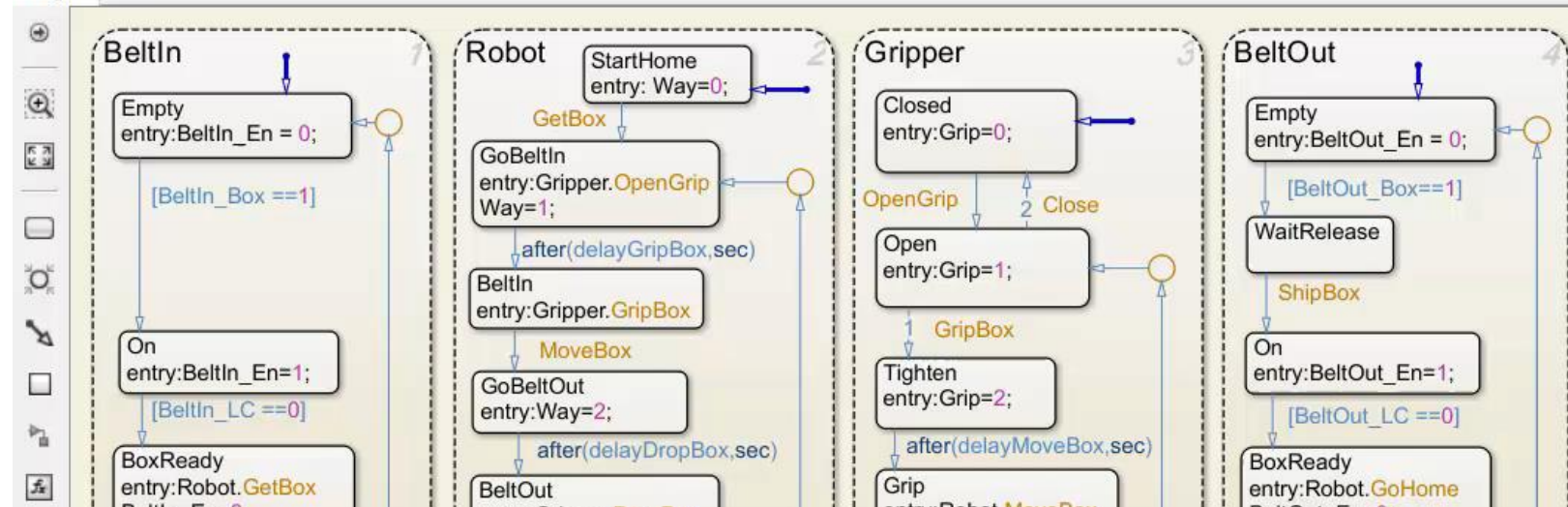
- Determine requirements for actuation system
- Minimize power consumption using optimization algorithms
- Design, test, and verify control logic behavior with dynamic simulation



File Edit View Display Chart Simulation Analysis Code Tools Help

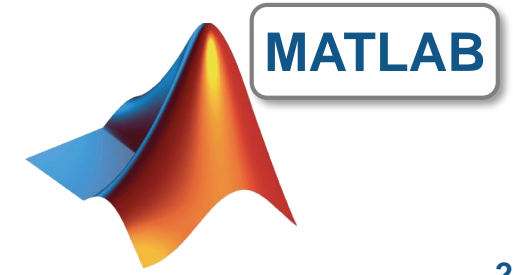
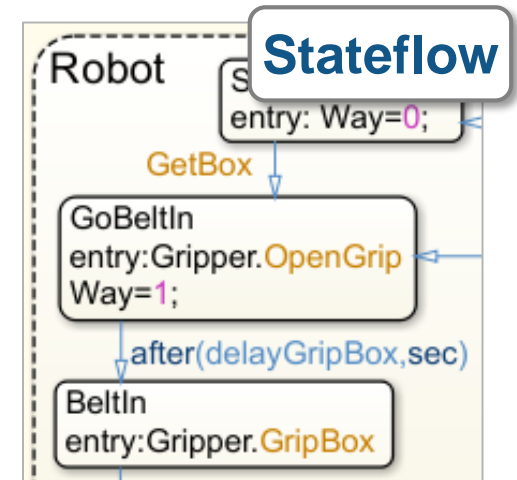
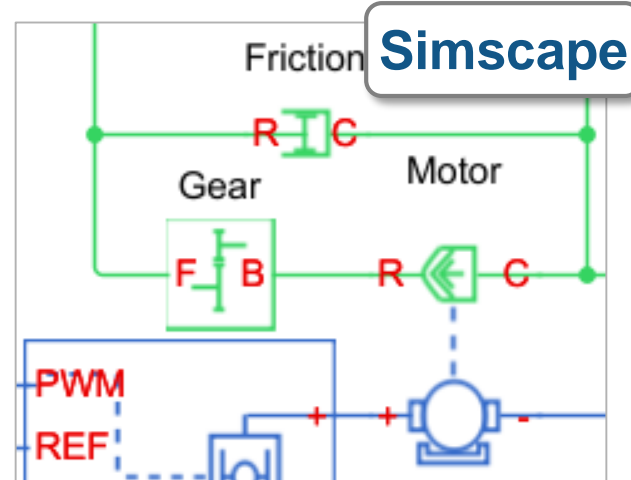
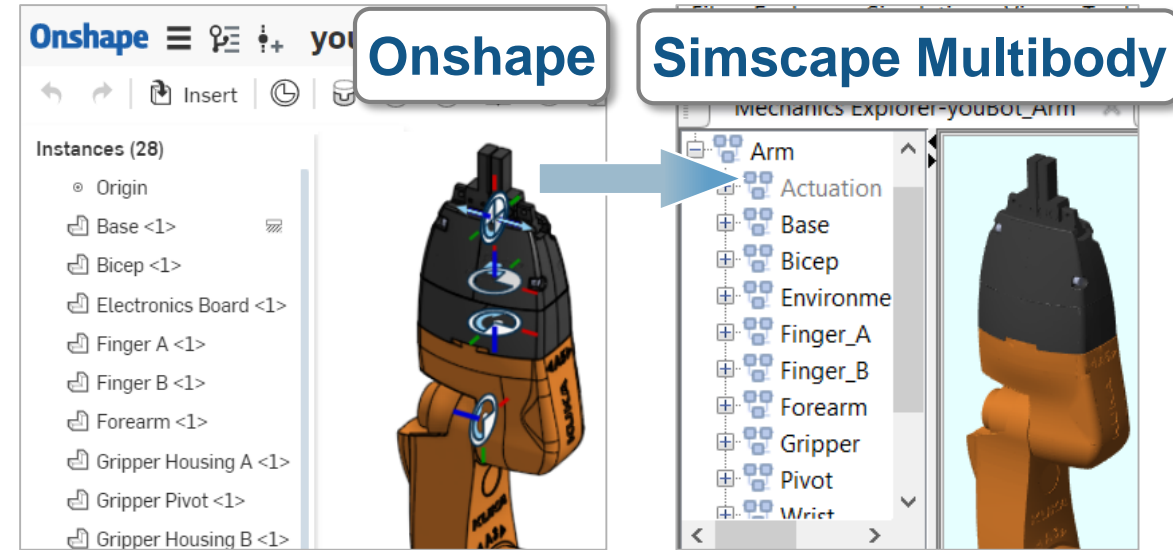


Logic



How we did it

- Convert **Onshape** CAD assemblies into dynamic simulation models with **Simscape Multibody**
- Add electric actuators with **Simscape** and control logic using **Stateflow**
- Perform dynamic simulation in **Simulink**
- Optimize system using **MATLAB**



Summary

- Onshape and MathWorks enable engineers to combine CAD models with multidomain, dynamic simulation
- Results:
 1. Optimized mechatronic systems
 2. Improved quality of overall system
 3. Shortened development cycle
- Visit us at our section of this booth and see web pages for more information
www.onshape.com, www.mathworks.com

