

# Virtuelle Inbetriebnahme und Optimierung von Robotersystemen mit Simscape

## MATLAB EXPO 2017

## In this session

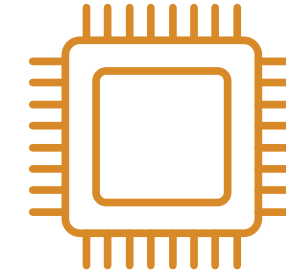
- Onshape and MATLAB enable engineers to combine CAD models with multidomain, dynamic simulation

**MATLAB**

## In this session

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

**MATLAB**



**Simulink**



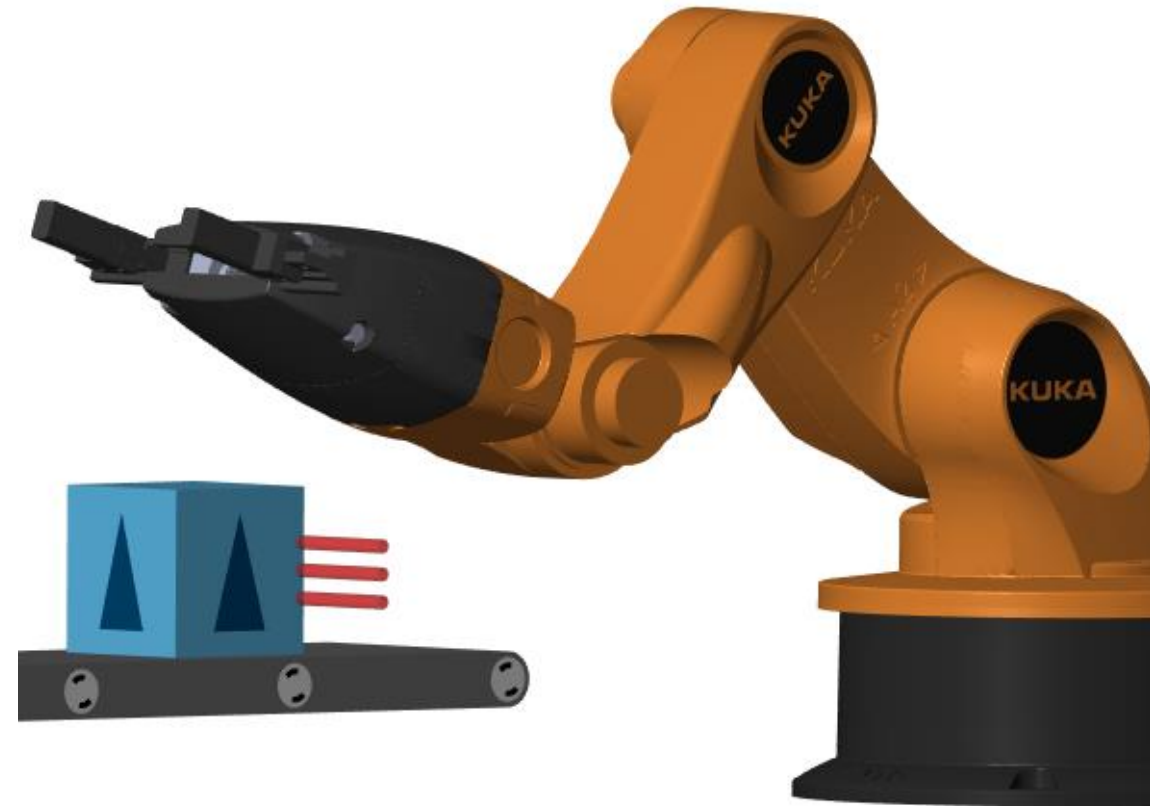
**Simscape**



**Stateflow**

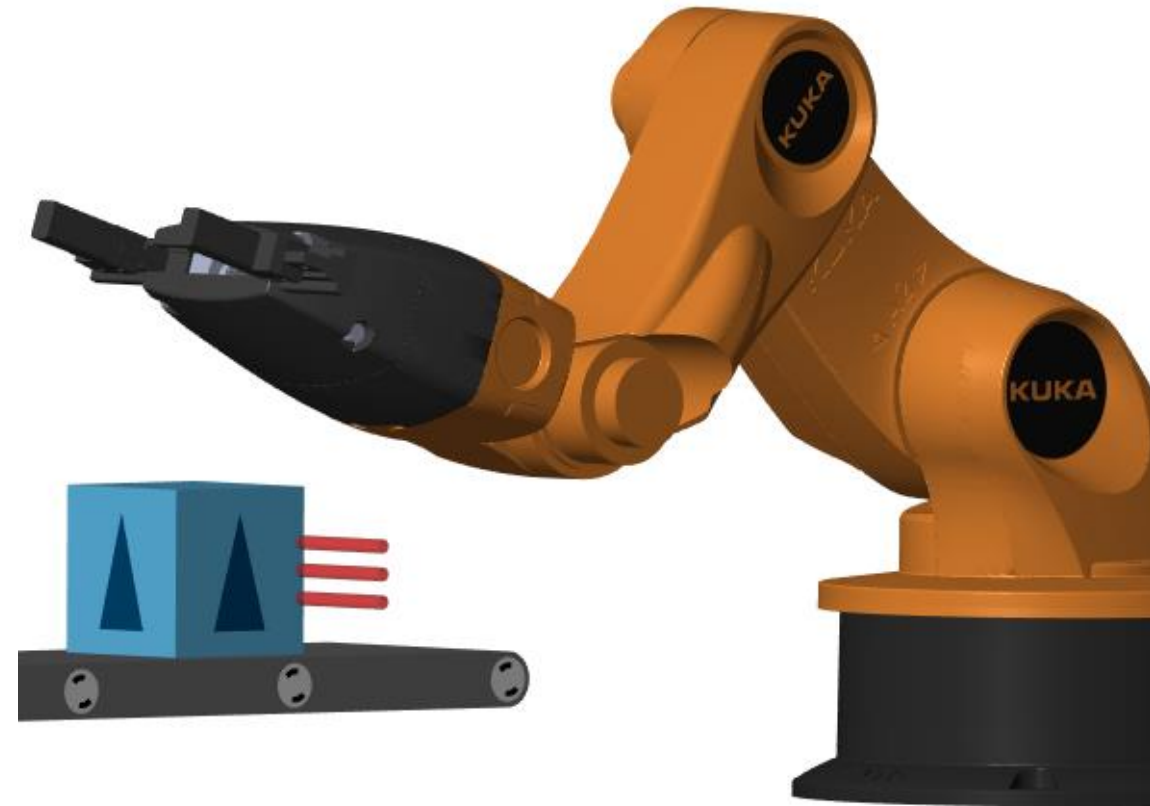
## In this session

- MathWorks enables 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



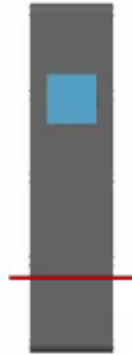
## Why Combine CAD and Multidomain, Dynamic Simulation?

- **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:**

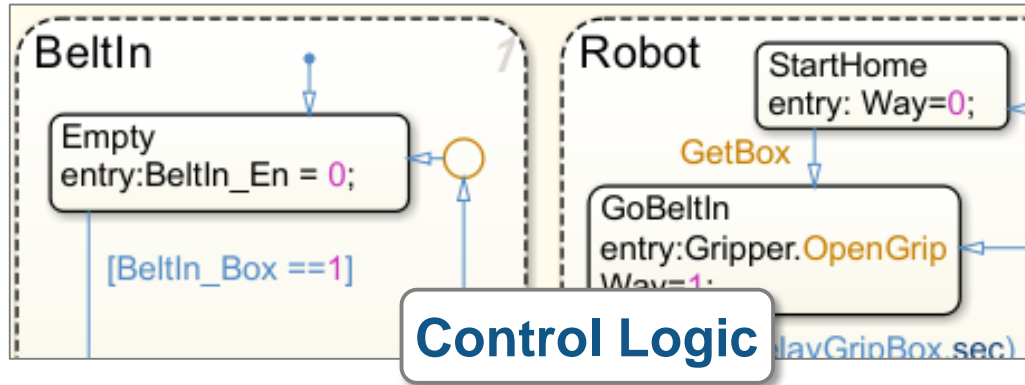


**Challenge:** Select motors and define controls for robot and conveyor belts.

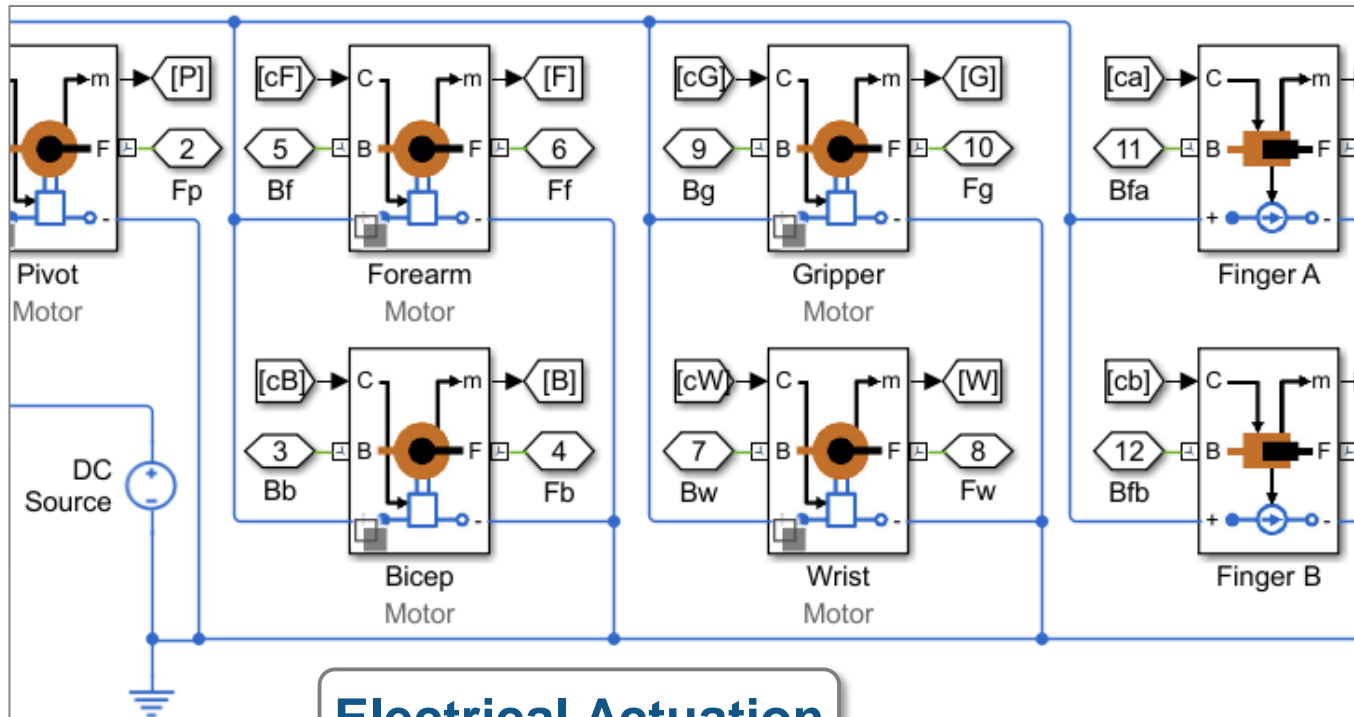
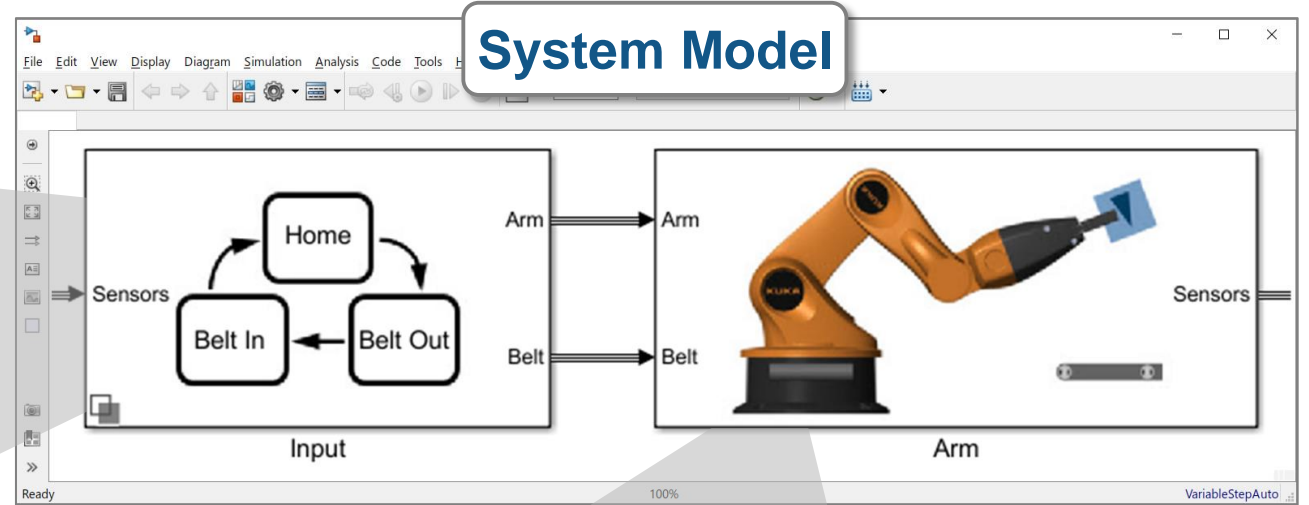
**Solution:** Import Onshape model into Simscape; use simulation to define actuator requirements and control logic

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

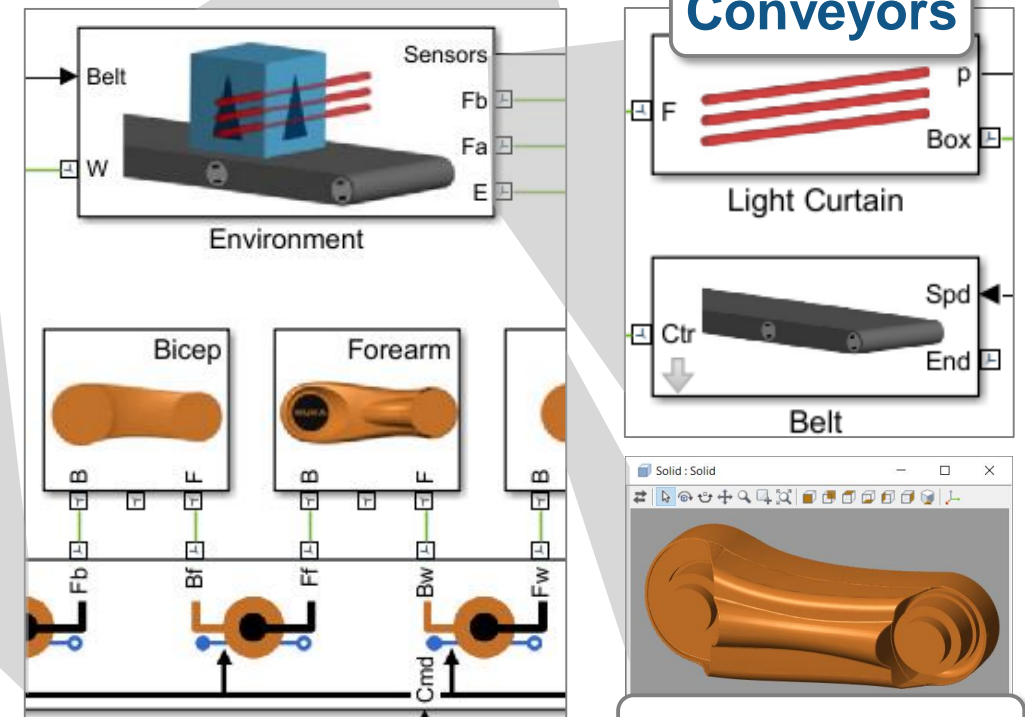
# System Model



Control Logic



Electrical Actuation

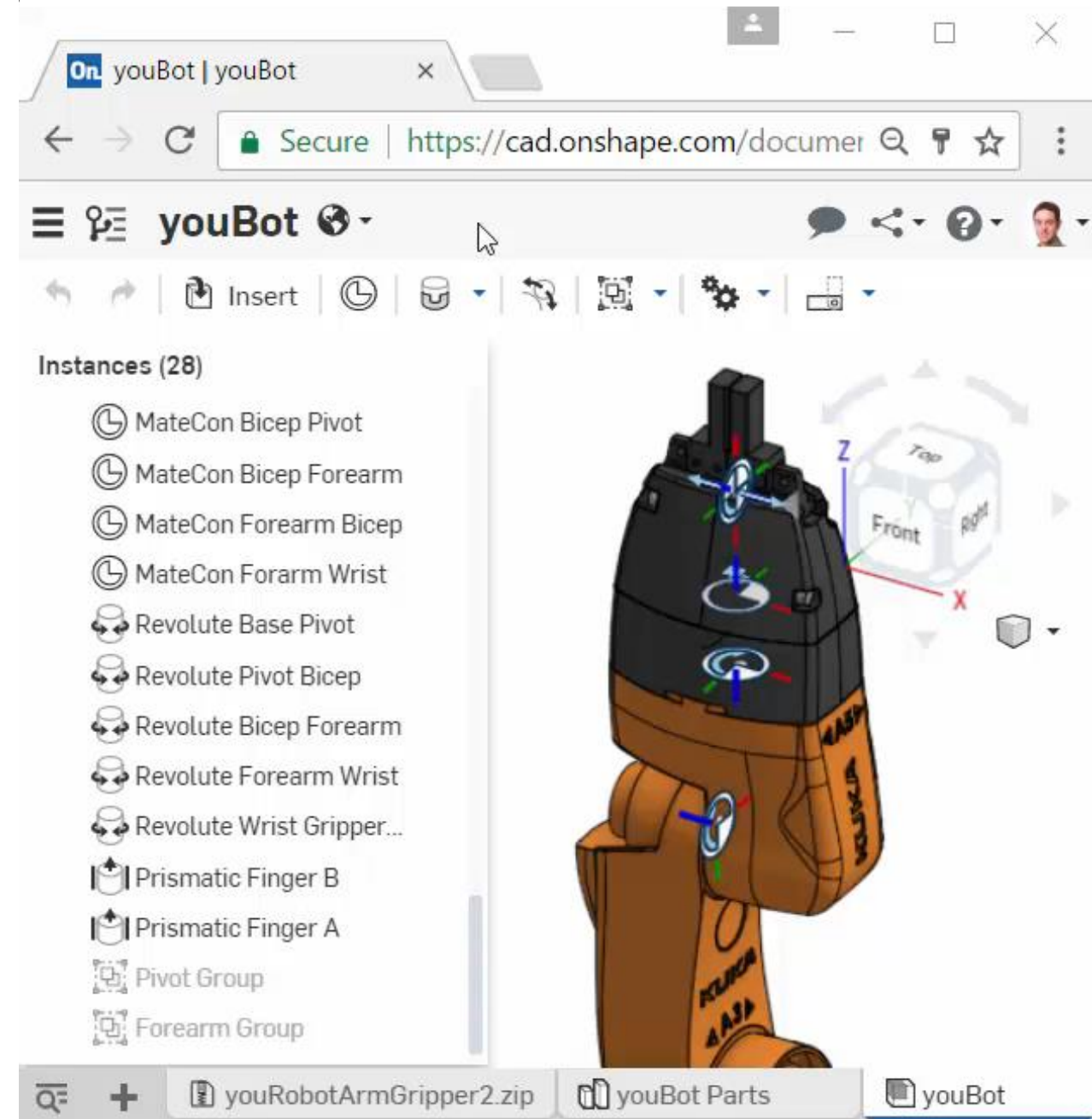


CAD Assembly



# Robot Mechanical Design

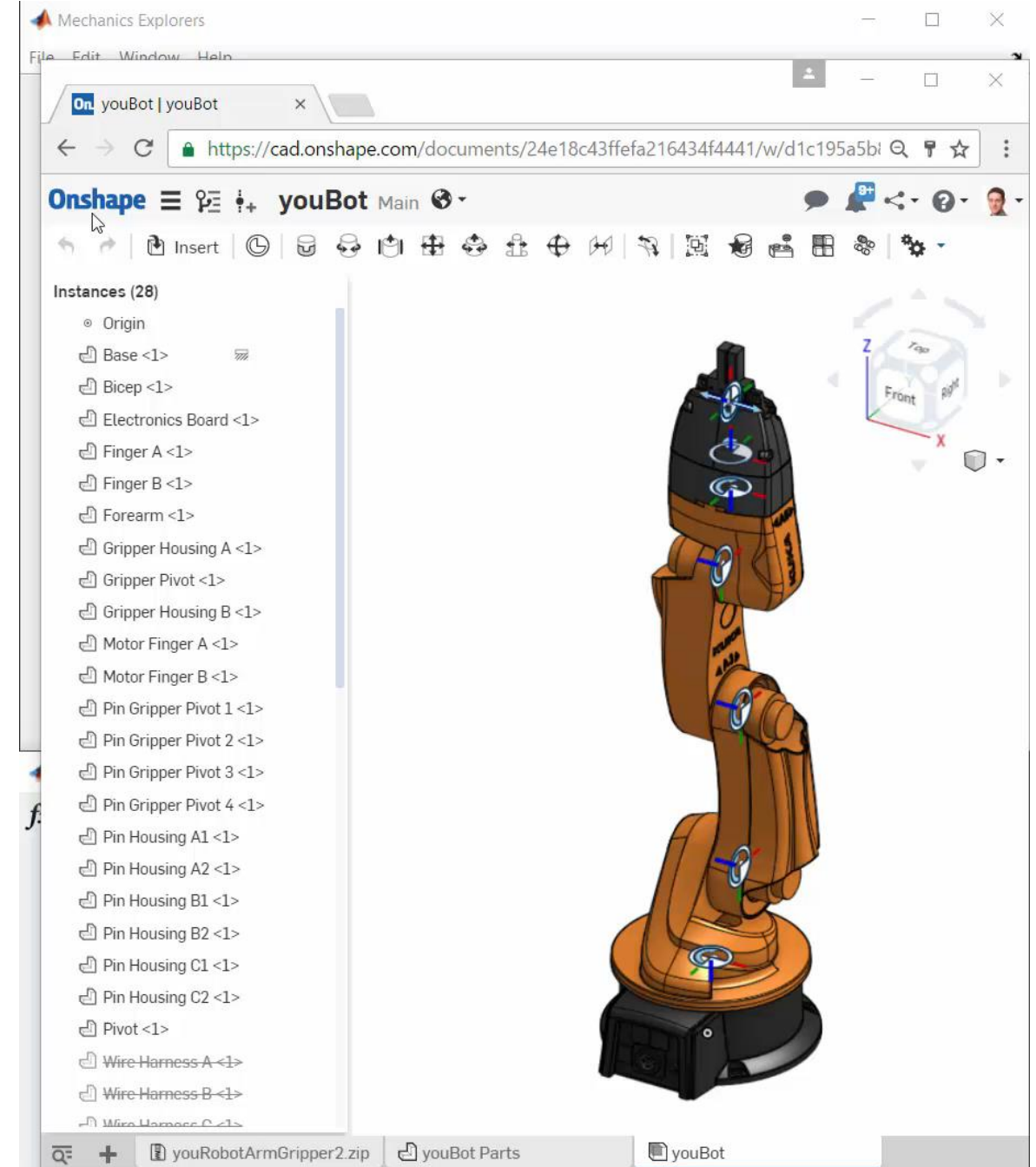
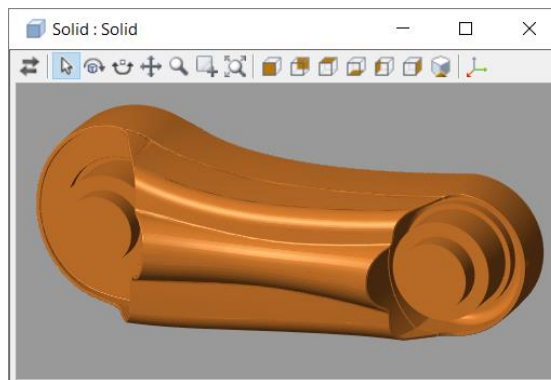
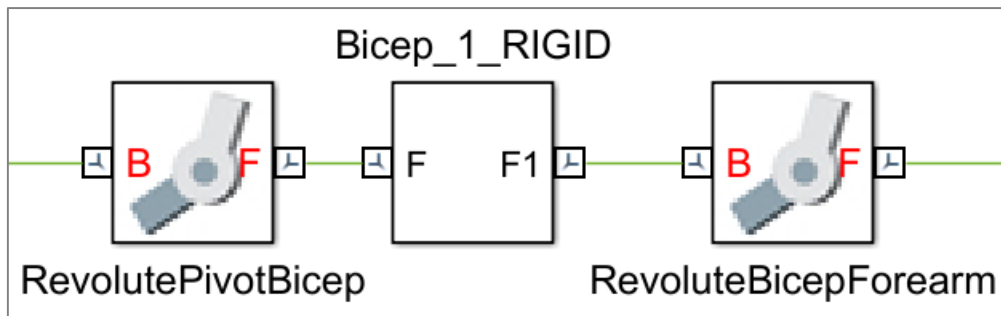
- 5 degrees of freedom, and a gripper
- Key advantage of Onshape: Ability to directly define joints
  - Exact mapping to constraints used in multibody simulation
- System engineer reuses mechanical design in dynamic simulation





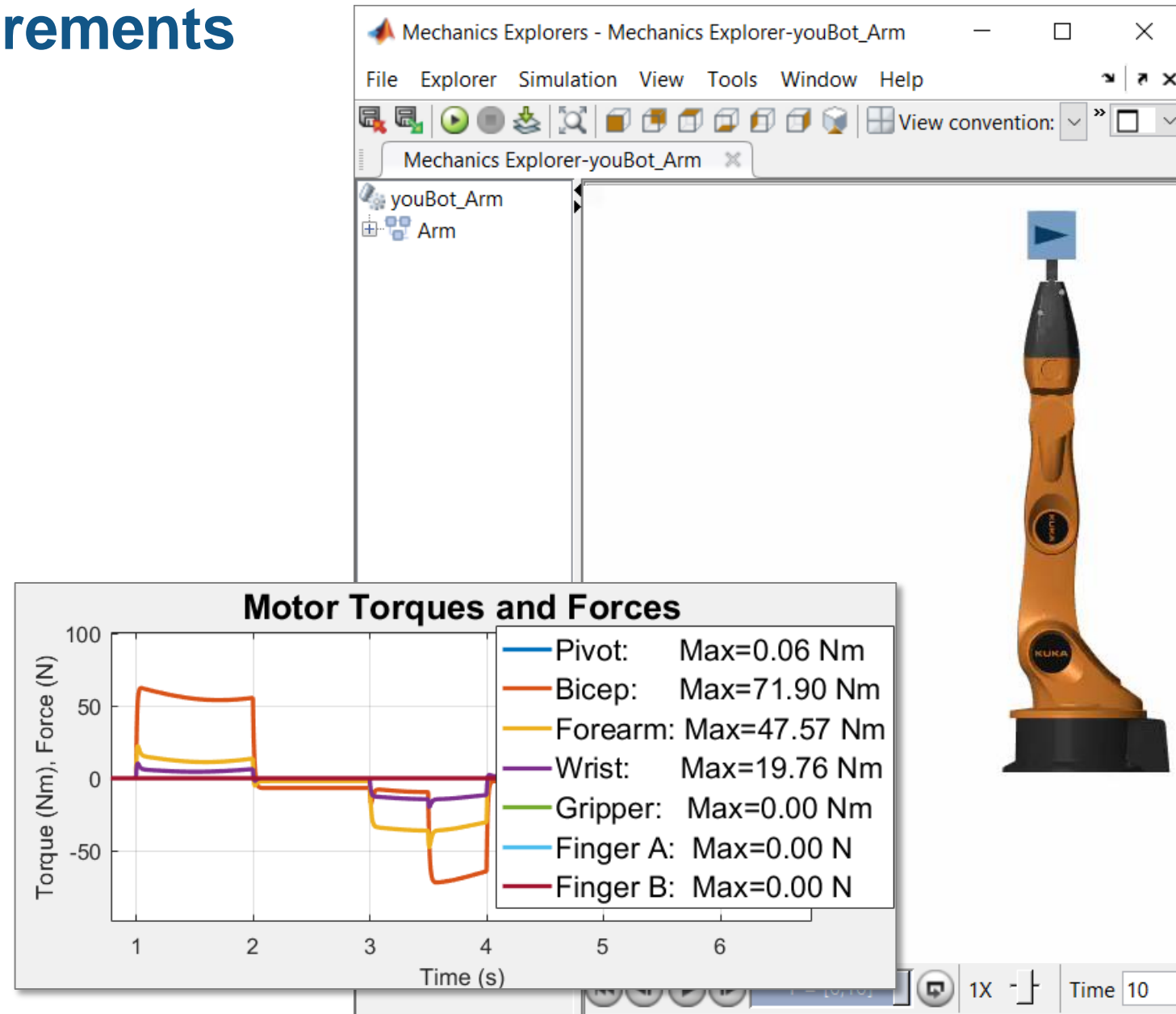
# 1. Import Model from Onshape

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



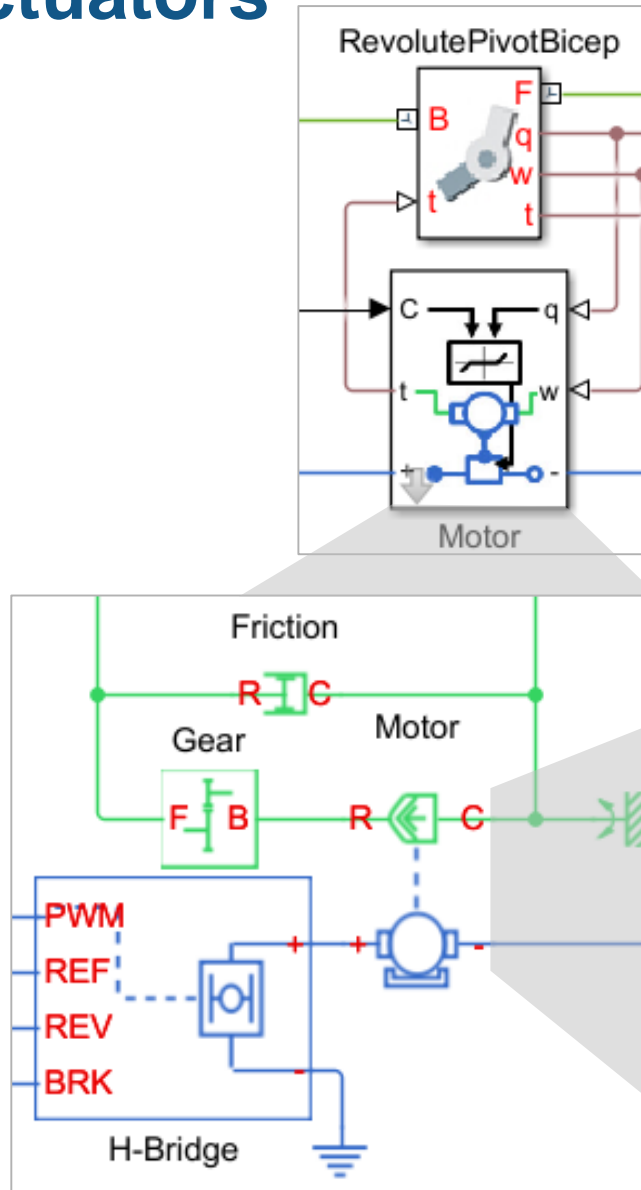
## 2. Determine Motor Requirements

- 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	$\Omega$	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 <sup>2</sup>	135

#### Electrical Torque

#### Mechanical

Model parameterization:

Armature resistance:

Armature inductance:

Torque constant:

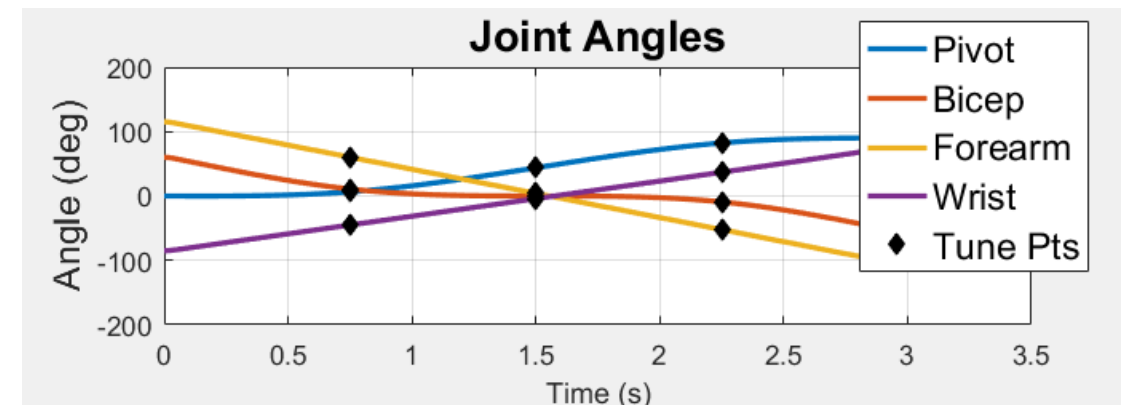
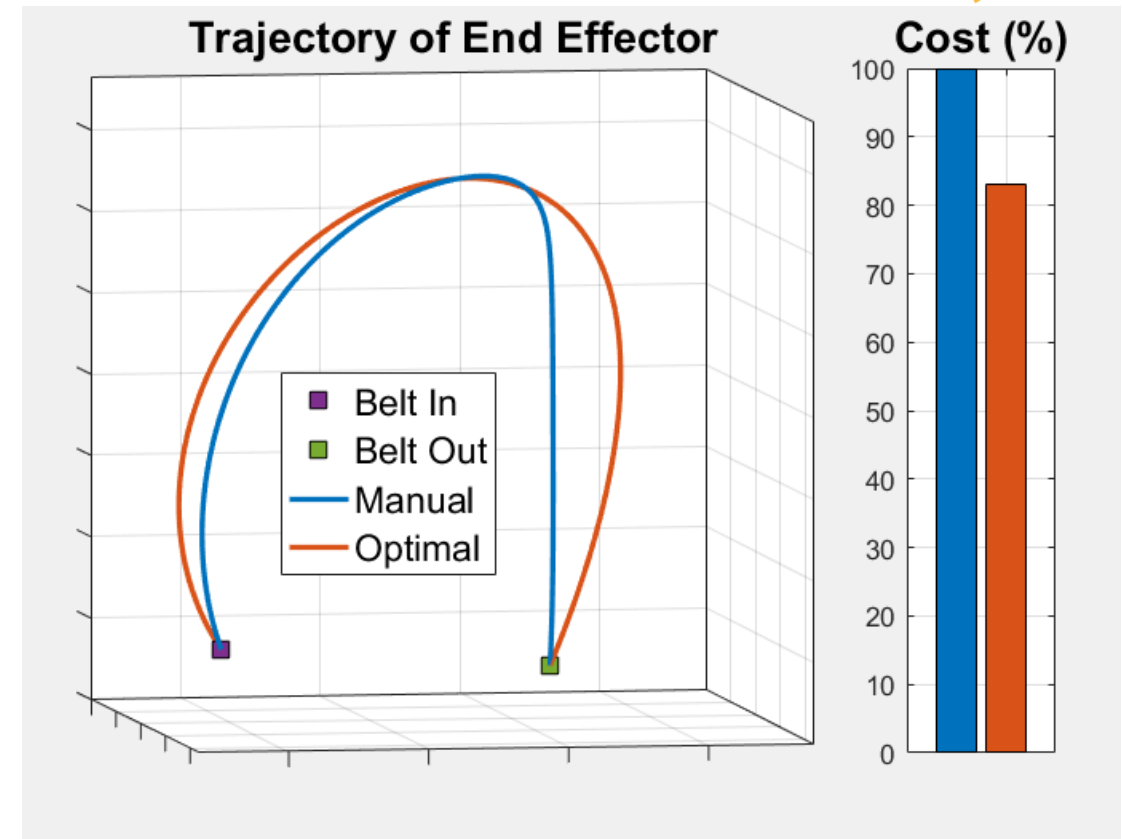
## 4. Minimize Power Consumption

### Model:

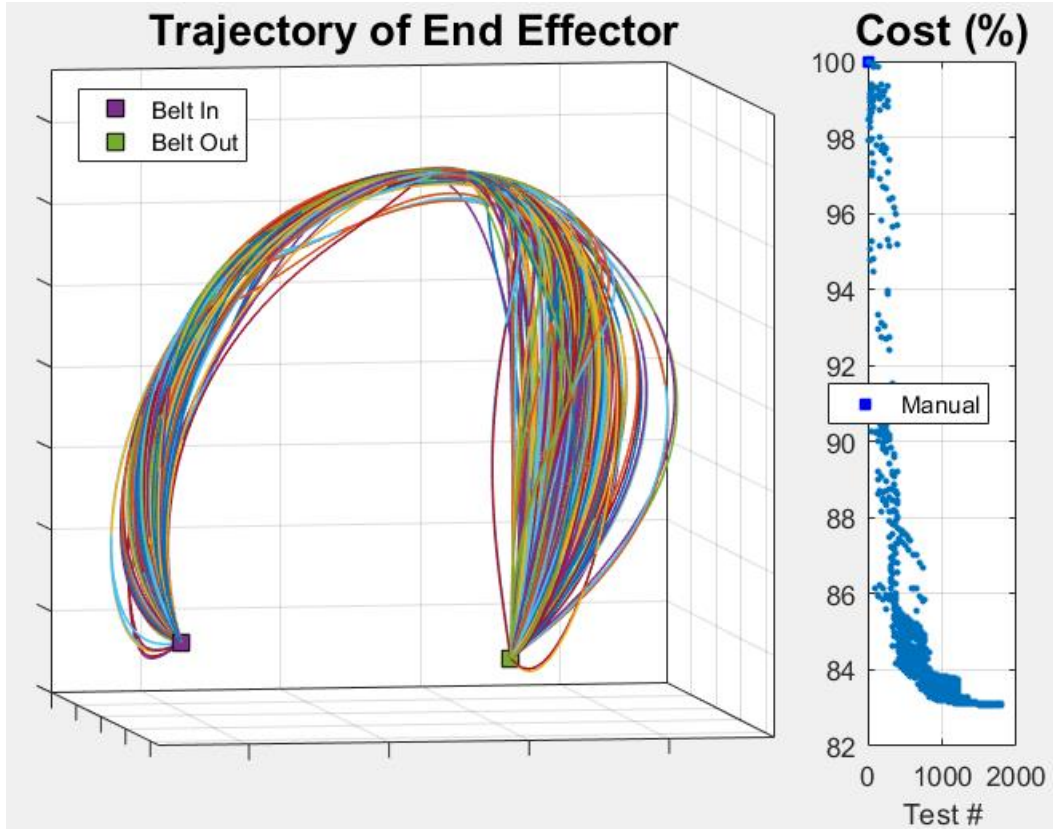


**Challenge:** Identify arm trajectory that minimizes power consumption.

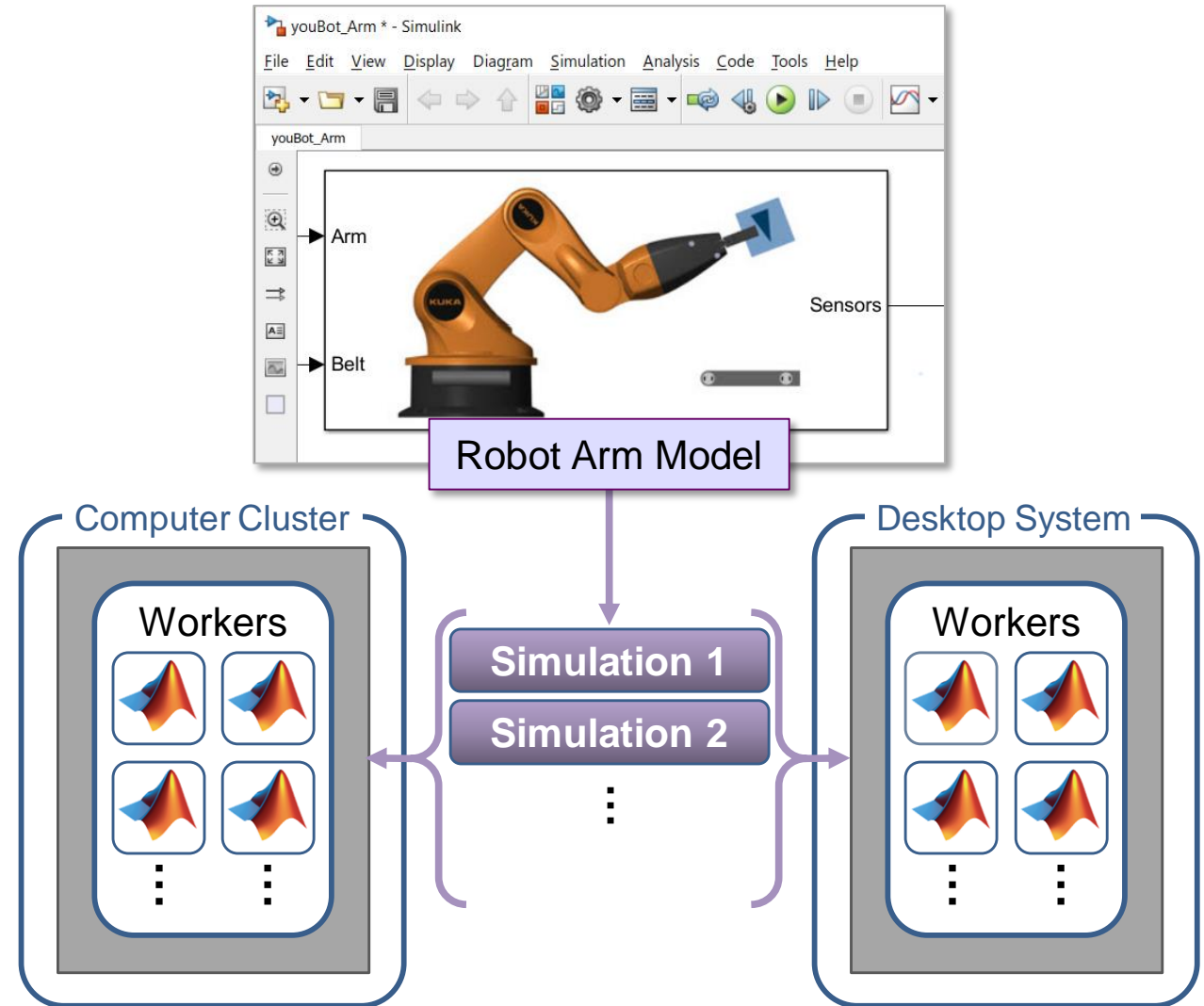
**Solution:** Use dynamic simulation to calculate power consumption, and use optimization algorithms to tune trajectory.



# Accelerate Design Iterations Using Parallel Computing



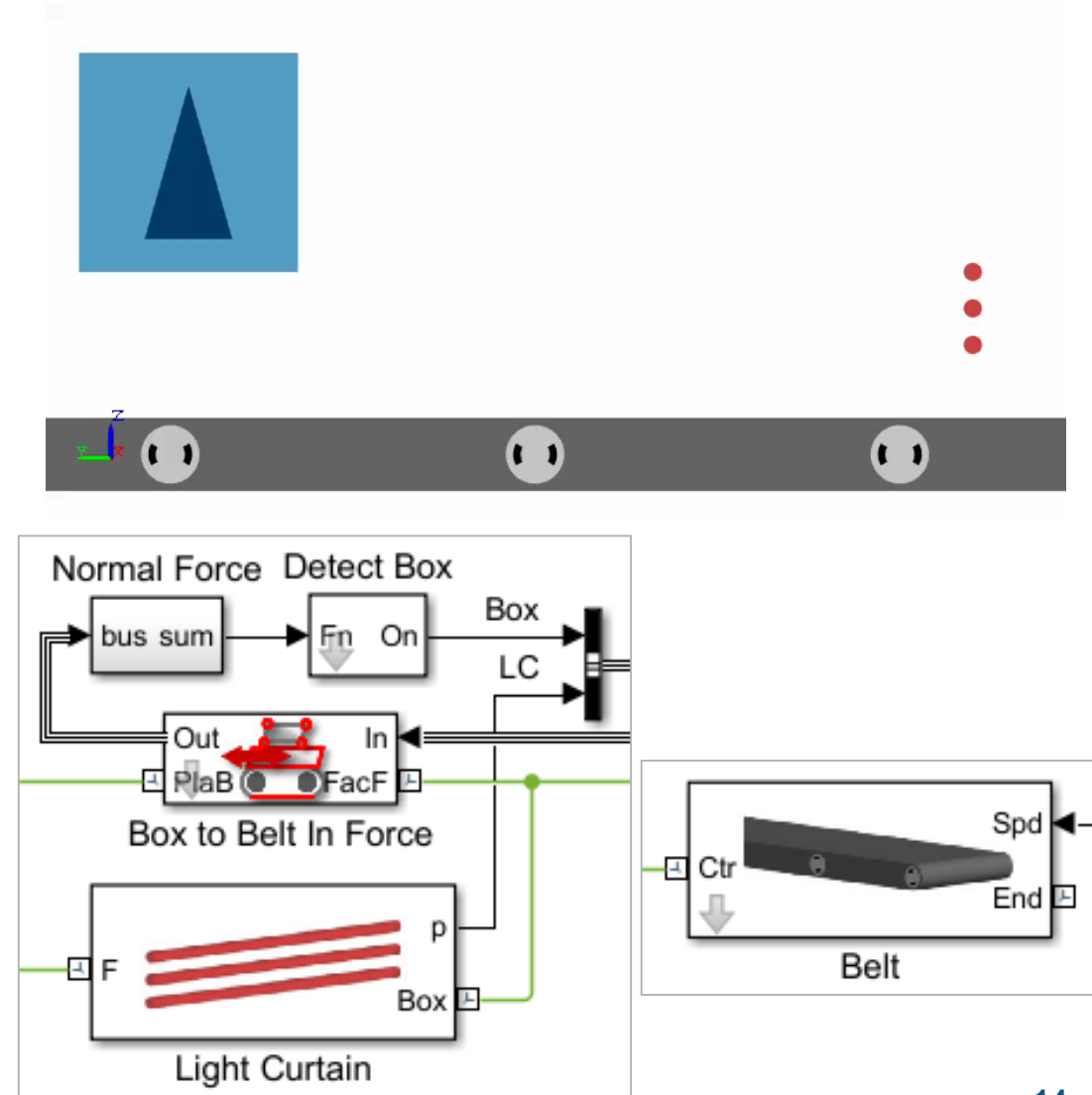
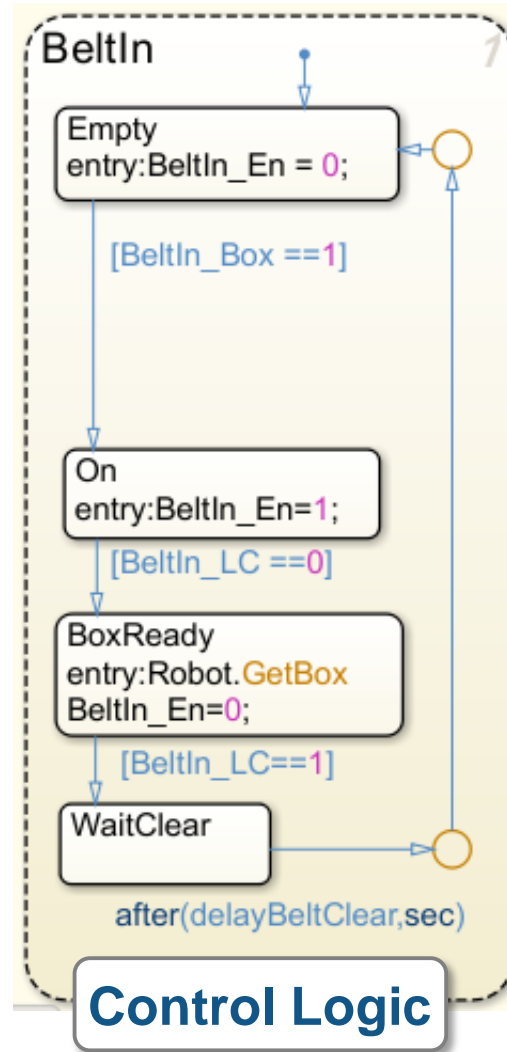
This optimization task required nearly 2000 simulations.



Running simulations in parallel speeds up your testing process.

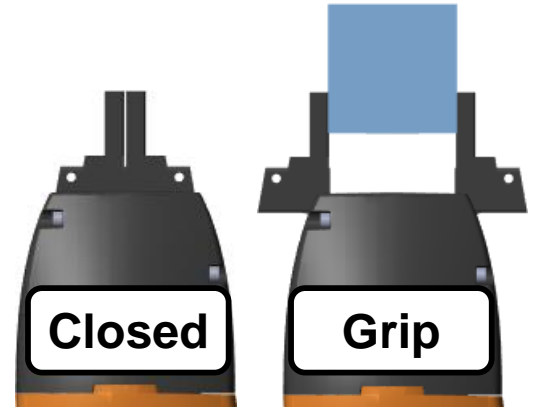
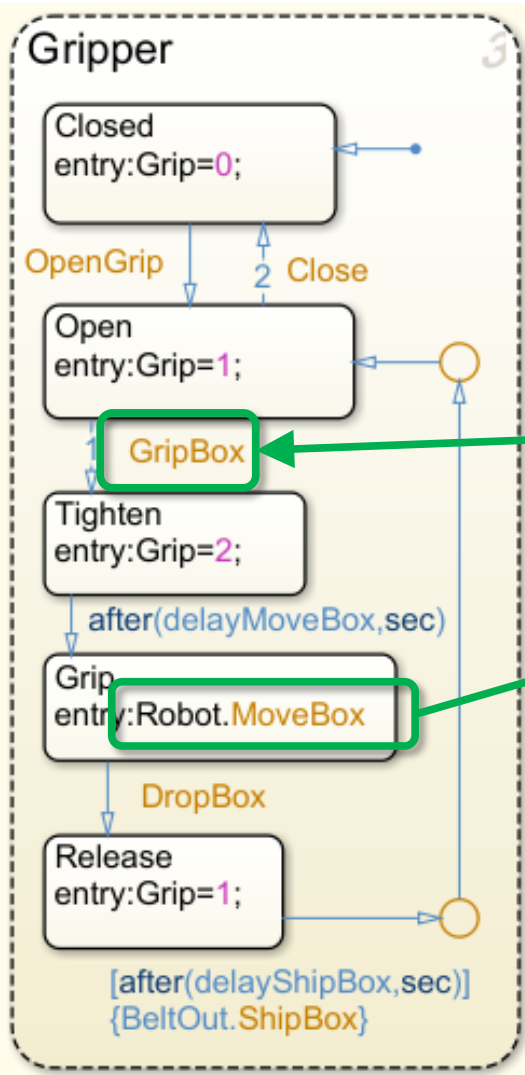
# 5. Design Control Logic for Arm and Conveyor Belts

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

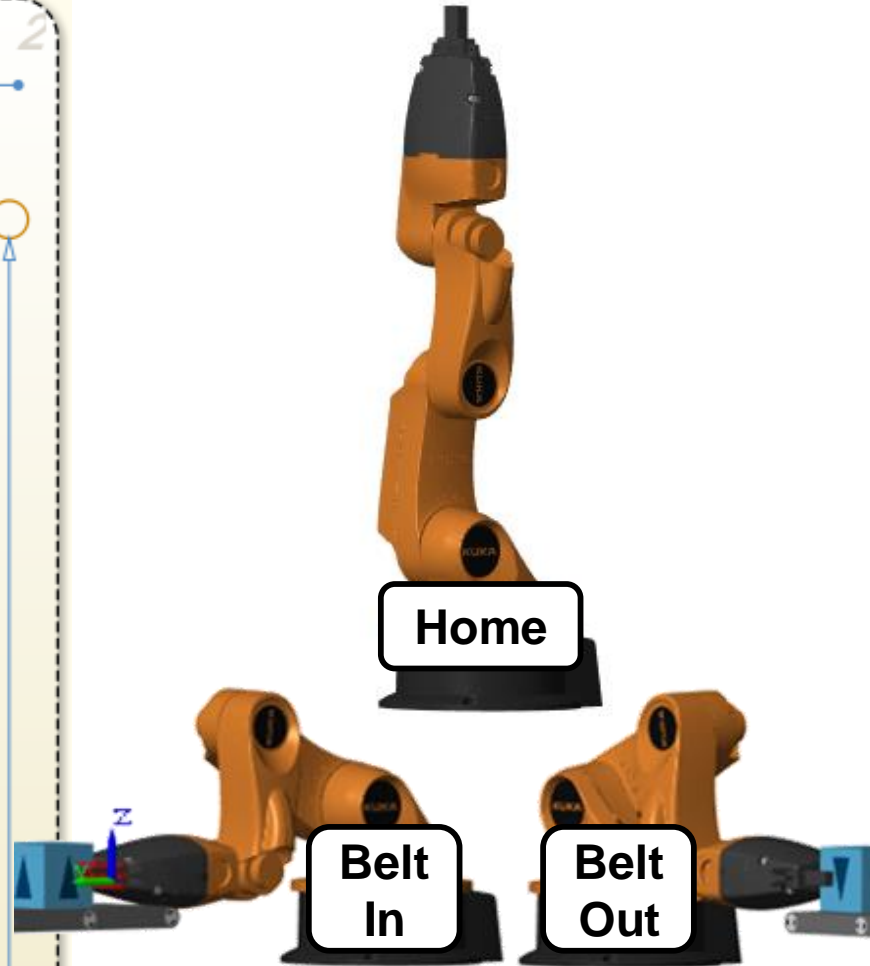
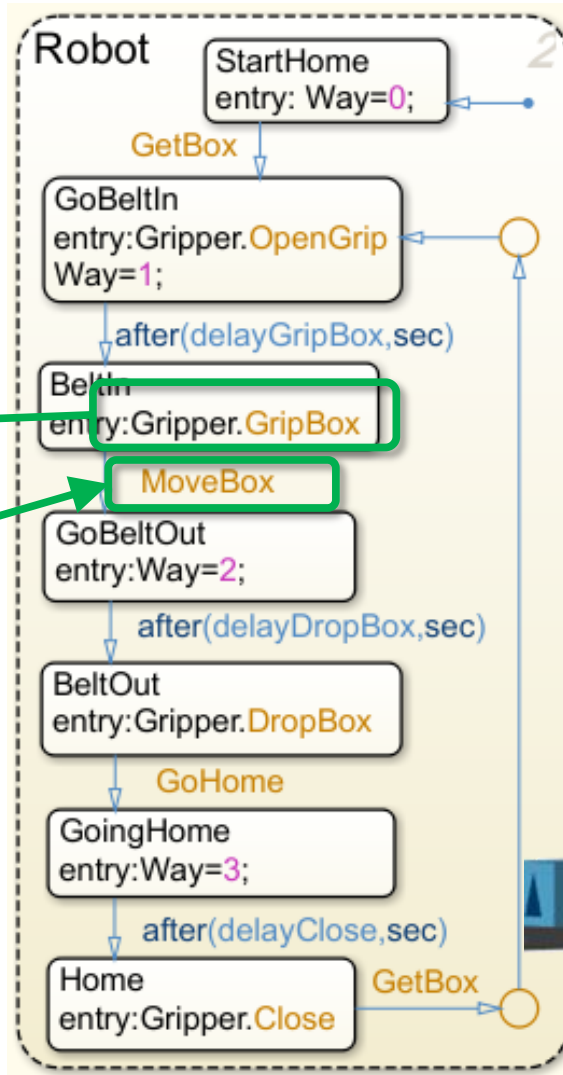




# 5. Design Control Logic for Arm and Conveyor Belts



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

The Stateflow chart is divided into four main sections:
 

- BeltIn (1):** Starts at an 'Empty' state (entry: BeltIn\_En = 0;). When a box is detected ([BeltIn\_Box == 1]), it transitions to an 'On' state (entry: BeltIn\_En = 1;). After a delay ([BeltIn\_LC == 0]), it moves to 'BoxReady' (entry: Robot.GetBox, BeltIn\_En = 0;). Once the robot is ready ([BeltIn\_LC == 1]), it transitions to 'WaitClear' (after: delayBeltClear, sec).
- Robot (2):** Starts at 'StartHome' (entry: Way = 0;). On 'GetBox' (entry: Gripper.OpenGrip, Way = 1;), it delays (after: delayGripBox, sec) and then moves to 'BeltIn' (entry: Gripper.GripBox). After 'MoveBox' (entry: Way = 2;), it delays (after: delayDropBox, sec) and moves to 'BeltOut' (entry: Gripper.DropBox). On 'GoHome' (entry: Way = 3;), it delays (after: delayClose, sec) and moves to 'Home' (entry: Gripper.Close). From 'Home', it transitions back to 'GetBox'.
- Gripper (3):** Starts at 'Closed' (entry: Grip = 0;). On 'OpenGrip' (entry: Grip = 1;), it delays (after: delayMoveBox, sec) and moves to 'Tighten' (entry: Grip = 2;). On 'DropBox' (entry: Robot.MoveBox), it delays (after: delayShipBox, sec) and moves to 'Release' (entry: Grip = 1;). From 'Release', it transitions back to 'Closed'.
- BeltOut (4):** Starts at 'Empty' (entry: BeltOut\_En = 0;). When a box is detected ([BeltOut\_Box == 1]), it transitions to 'WaitRelease'. On 'ShipBox' (entry: BeltOut\_En = 1;), it delays ([BeltOut\_LC == 0]) and moves to 'BoxReady' (entry: Robot.GoHome, BeltOut\_En = 0;). Once the robot is ready ([BeltOut\_LC == 1]), it transitions to 'WaitClear' (after: delayBeltClear, sec).

Running 100% ode15s

Mechanics Explorers - Mechanics Explorer-youBot\_Arm

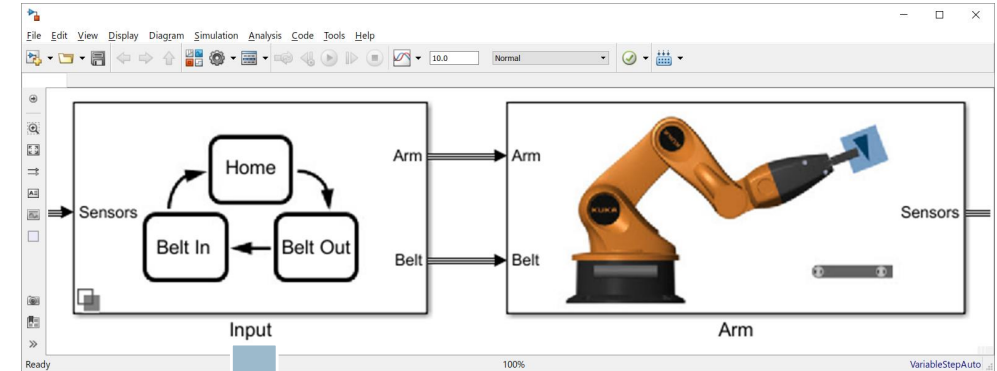
File Explorer Simulation View Tools Window Help

Mechanics Explorer-youBot\_Arm

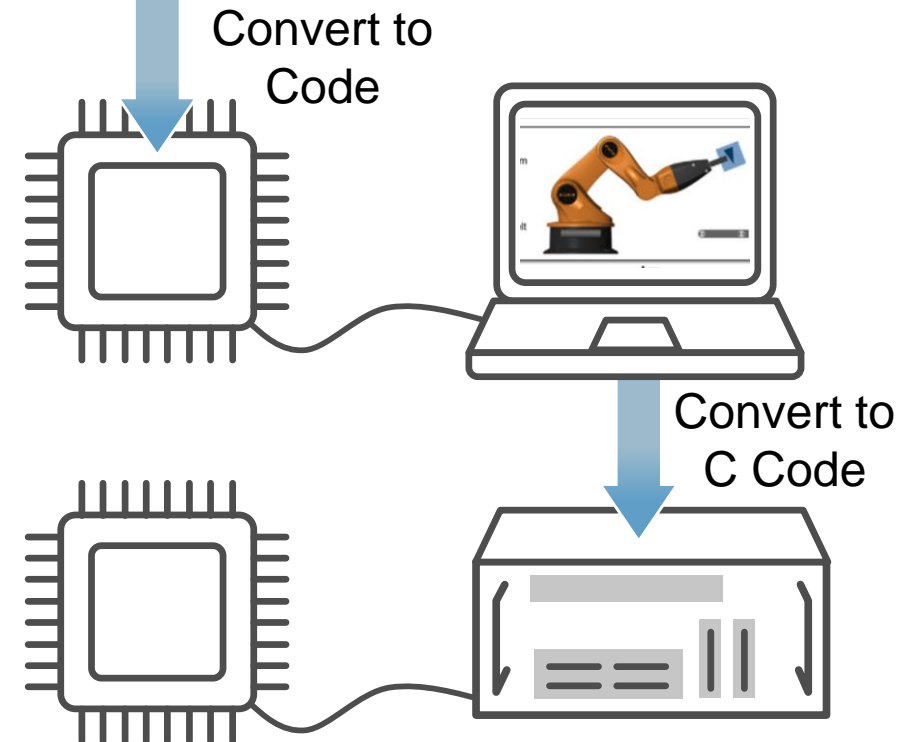
The simulation shows a 3D model of the youBot Arm and conveyor belt system. The arm is orange and black, and the conveyor belt is black with a red line. The simulation is running at 0% and 1X speed. The time is 0.

# Test Production Control Software

- Automatically convert algorithms to production code
  - C Code, IEC 61131-3 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)**

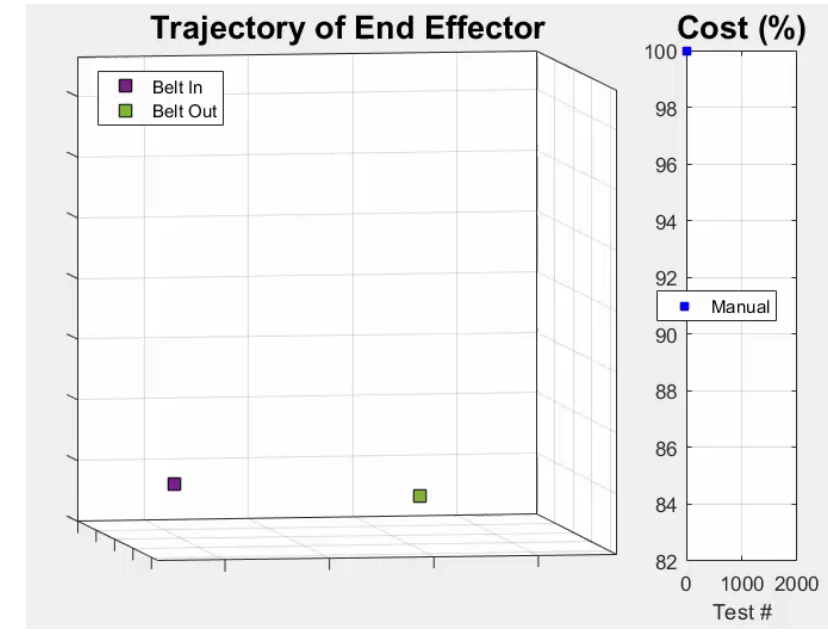
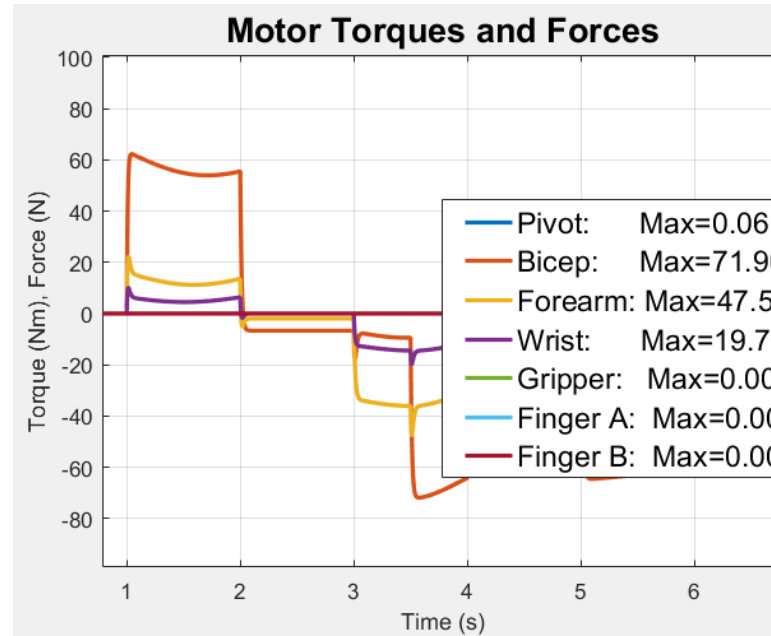


**Hardware-in-the-Loop (HIL)**



# What we have shown

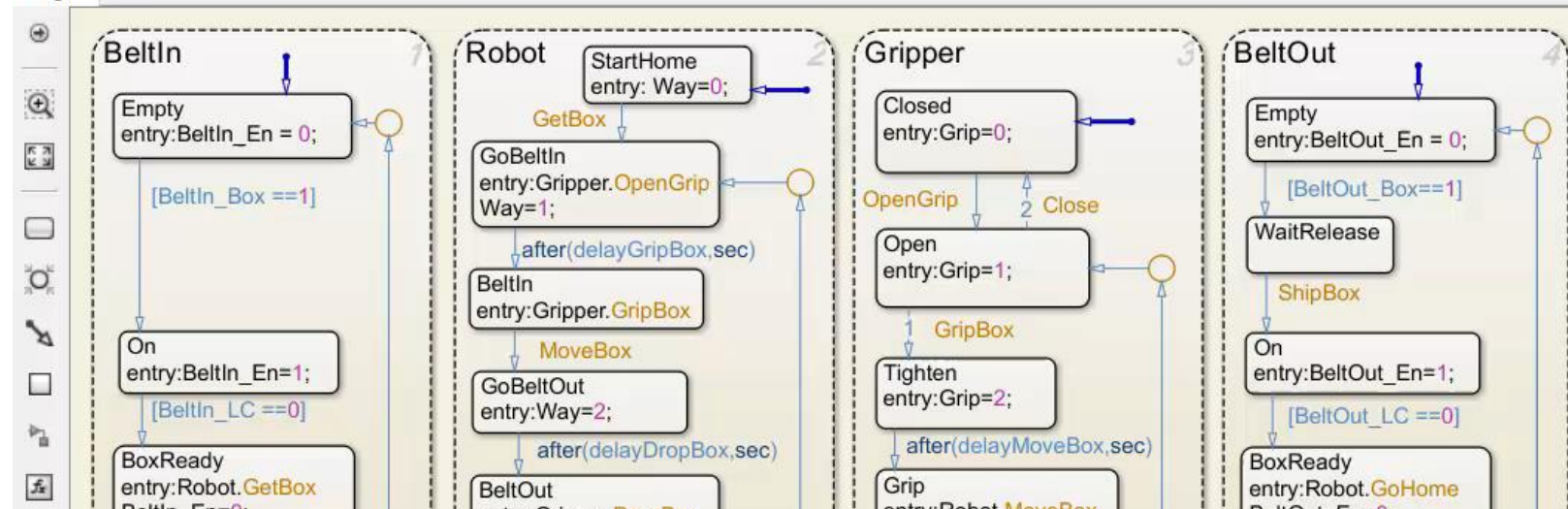
- Determine requirements for actuation system
- Minimize power consumption using optimization algorithms
- Design, test, and verify control logic behavior with dynamic simulation



File Edit View Display Chart Simulation Analysis Code Tools Help

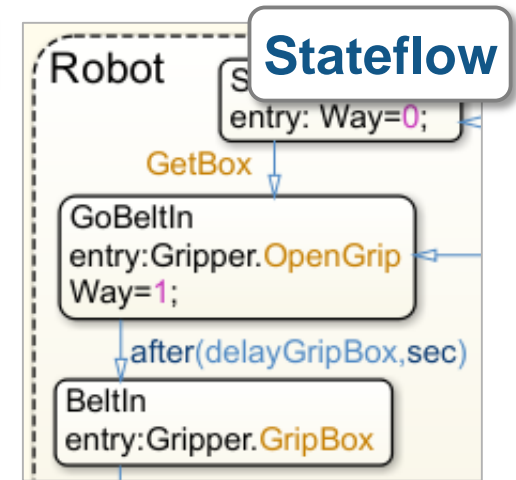
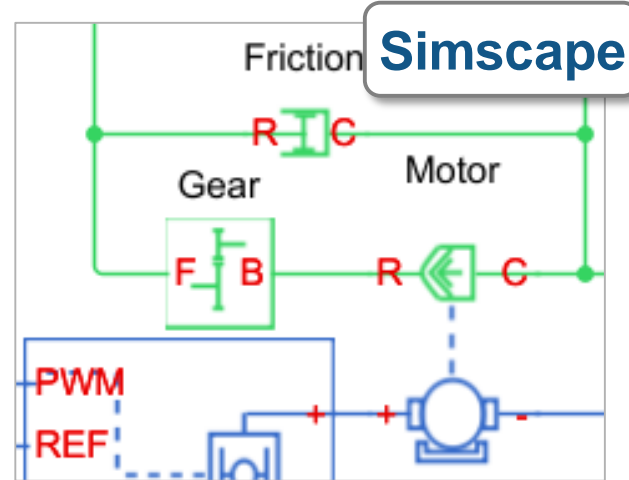
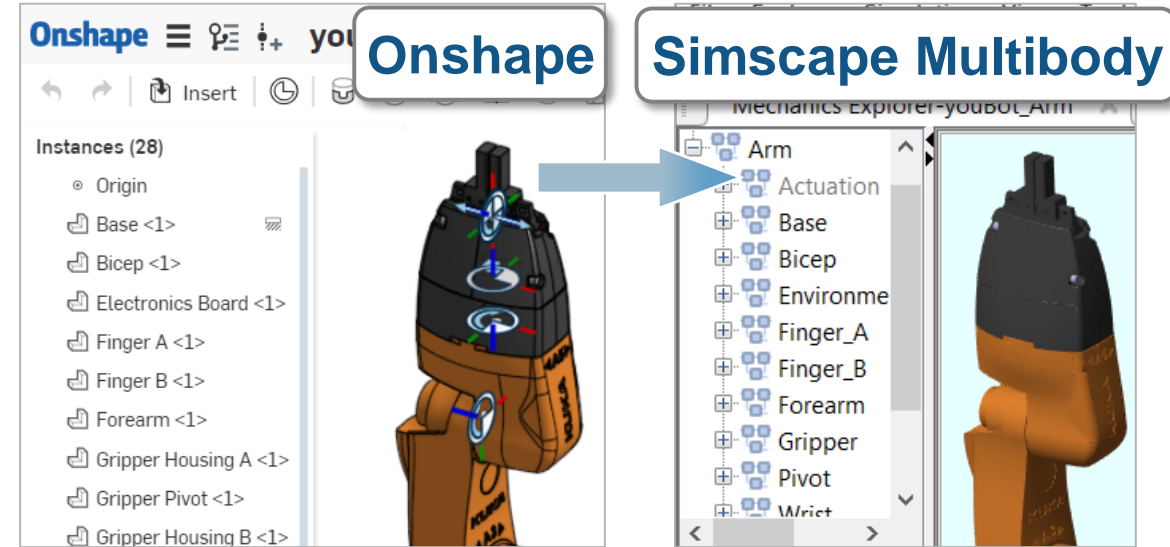


Logic



# 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

- MathWorks enables 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.mathworks.com](http://www.mathworks.com)

