An Embedded Scalable Linear Model Predictive Hardware-based Controller using ADMM

Pei Zhang, Joseph Zambreno and Phillip H. Jones

Presenter: Pei Zhang

Iowa State University

peizhang@iastate.edu

July 4, 2017

Overview

1. Related Work
2. Background
   - State Space Model
   - Model Predictive Optimal Control
   - Splitting Method
3. ADMM Hardware Architecture
   - Architecture Overview
   - Trajectory Setting During Runtime
   - Latency Analysis
4. Evaluation
   - Palnt on Chip
   - SW/HW Co-design
5. Conclusion

Quadratic Programming (QP) solutions

MPC can be posed as a Quadratic Programming problem.

QP problems can be solved reliably via various iterative methods.

- Interior-Point Method (IPM)
- Active Set Method (ASM)
- Splitting Method

FPGA-based QP solutions

Compare IPM and ASM in FPGA

- ASM gives lower computing complexity and converges faster when the number of decision variables and constraints are small.
- IPM is a better choice when considering scalability.

State Space Model

A discrete state-space model defines what state a system will be in one-time step into the future:

$$x_{k+1} = Ax_k + Bu_k \tag{1}$$

$$y_k = Cx_k + Du_k \tag{2}$$

- x_k represents the state of the system at time k
- u_k represents the input acting on the system at time k
- y_k represents outputs of the system at time k
- A is a matrix that defines the internal dynamics of the system
- B is a matrix that defines how the input acting upon the system impact its state
- C is a matrix that transforms states of the system into outputs (y_k)

Augmented Vector

$$U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+H_u} \end{bmatrix}, \quad \Delta U_k = \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+H_{u-1}} \end{bmatrix}, \quad X_k = \begin{bmatrix} x_k \\ x_{k+1} \\ \vdots \\ x_{k+H_p} \end{bmatrix} \tag{3}$$

Where:

- H_u: changeable future input horizon. We assume input u_k will be constant after H_u time steps.
- H_p: prediction horizon. Normally, $H_p \geq H_u$.
- $U_k \in \mathbb{R}^{M(H_u+1)}$, $\Delta U_k \in \mathbb{R}^{MH_u}$, $X_k \in \mathbb{R}^{N(H_p+1)}$.

Cost Function

$$\mathbb{C}(k) = \frac{1}{2}\Big(\sum_{i=k}^{k+H_p} (x_i^T q_i x_i - 2r_i^T q_i x_i) + \sum_{i=k}^{k+H_u} u_i^T p_i u_i + \sum_{i=k}^{k+H_{u-1}} \Delta u_i^T s_i \Delta u_i \Big) + Const \tag{4}$$

$$\mathbb{C}(k) = \frac{1}{2} \begin{bmatrix} X_k \\ U_k \\ \Delta U_k \end{bmatrix}^T \begin{bmatrix} Q & & \\ & P & \\ & & S \end{bmatrix} \begin{bmatrix} X_k \\ U_k \\ \Delta U_k \end{bmatrix} - R_k^T Q X_k \tag{5}$$

Box Constraints

Consensus Form

One technique for partitioning variables in ADMM is writing the convex QP problem into consensus form:

$$\begin{aligned} &\textit{minimize}: \quad \mathbb{1}_{\mathcal{D}}(\chi) + \phi(\chi) \; + \; \mathbb{1}_{\mathcal{C}}(\zeta) \\ &\textit{subject to}: \; \chi = \zeta \end{aligned}$$

Consensus Form

One technique for partitioning variables in ADMM is writing the convex QP problem into consensus form:

$$\begin{aligned} &\textit{minimize}: \quad \underbrace{\mathbb{1}_{\mathcal{D}}(\chi) + \phi(\chi)}_{g(\chi)} + \underbrace{\mathbb{1}_{\mathcal{C}}(\zeta)}_{f(\zeta)} \\ &\textit{subject to}: \quad \chi = \zeta \end{aligned}$$

Consensus Form

$$
\begin{aligned}
g(\chi) &= \mathbb{1}_{\mathcal{D}}(\chi) + \phi(\chi) \\
f(\zeta) &= \mathbb{1}_{\mathcal{C}}(\zeta)
\end{aligned} \tag{6}
$$

$$\chi^{i+1} := prox_{g,\rho}(\zeta^i + \upsilon^i) \tag{7}$$

$$\zeta^{i+1} := prox_{f,\rho}(\chi^{i+1} + \upsilon^i) \tag{8}$$

$$\upsilon^{i+1} := \upsilon^i + \rho(\chi^{i+1} - \zeta^{i+1}) \tag{9}$$

Here, i is the iteration counter, $prox_{f,\rho}(\chi)$ is the proximal mapping (or proximal operator) of a convex function f:

$$prox_{f,\rho}(\chi) = arg\ \min_u (f(u) + \frac{\rho}{2}\|\chi - u\|_2^2)$$

$\rho > 0$ is the dual update step length.

Solve χ^{i+1}

Matrix-vector Multiply (MvM)

KKT Condition

$$\begin{aligned} &minimize: \ \frac{1}{2}(\chi^{i+1})^T E \chi^{i+1} + l^T \chi^{i+1} \\ &subject\ to: \ G\chi^{i+1} = h \end{aligned} \tag{10}$$

Solve ζ^{i+1}

Saturation Function

Solve v^{i+1}

Vector Plus Vector

ADMM Algorithm

Algorithm 1: ADMM algorithm

Start from $i = 0$ with arbitrary ζ^0 and υ^0.

do

$$I := \begin{bmatrix} Q * R_k \\ \mathbf{0} \end{bmatrix} - \rho(\zeta^i + \upsilon^i)$$

$$\chi^{i+1} := M_{11} * \begin{bmatrix} -I & x_k \end{bmatrix}^T$$

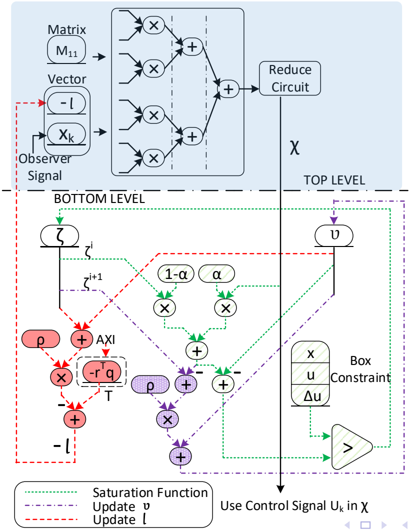
$$\zeta^{i+1} := sat(\chi^{i+1} - \upsilon^i, \mathbf{dom}\, \mathcal{C})$$

$$\upsilon^{i+1} := \upsilon^i + \rho(\zeta^{i+1} - \chi^{i+1}) \; i := i + 1$$

until *stopping criterion is satisfied*;

Hardware Architecture

Hardware Architecture for ADMM with Relaxation Parameter α.



Hardware Architecture

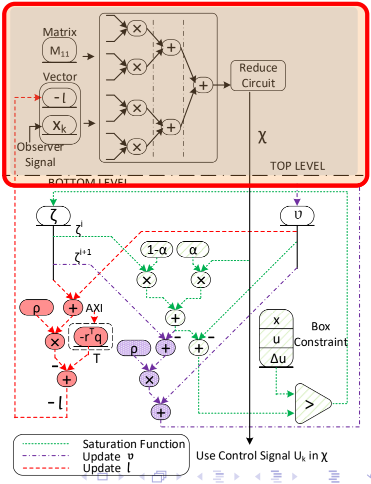
Processing Flow

Step 1
Solve KKT

Step 2
Saturation Function

Step 3
Update υ

Step 4
Update l



Hardware Architecture

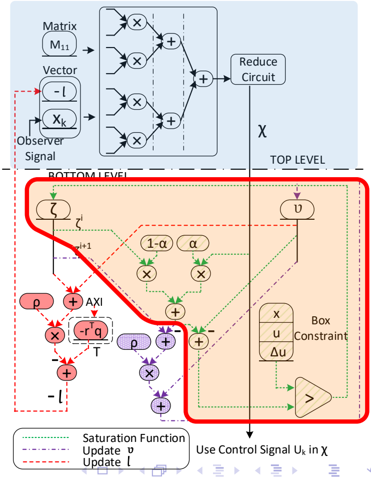
Processing Flow

Step 1
Solve KKT

Step 2
Saturation Function

Step 3
Update υ

Step 4
Update l



Hardware Architecture

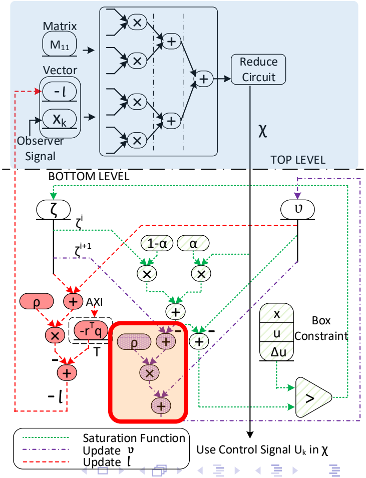
Processing Flow

Step 1
Solve KKT

Step 2
Saturation Function

Step 3
Update υ

Step 4
Update l



Hardware Architecture

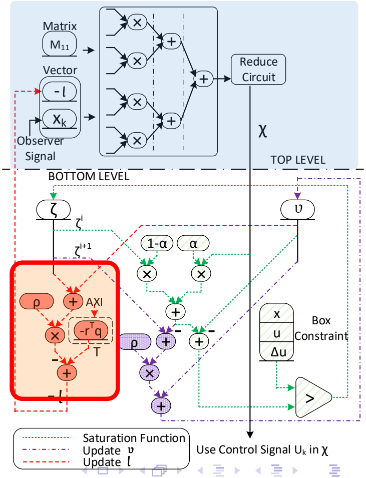
Processing Flow

Step 1
Solve KKT

Step 2
Saturation Function

Step 3
Update υ

Step 4
Update l



Hardware Architecture

Reduce Circuit

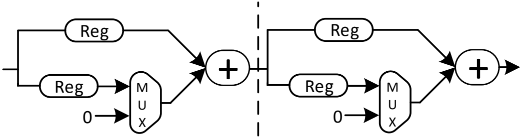


Figure: Reduce Circuit Architecture with Two Cascaded Adders

Hardware Architecture

Runtime Trajectory Planning

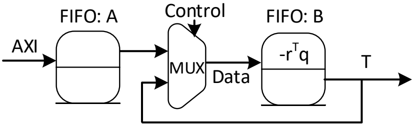


Figure: Runtime Trajectory Planning

Mass-spring System

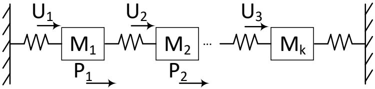


Figure: Mass-spring System

Emulation using Plant on Chip

Mass Position

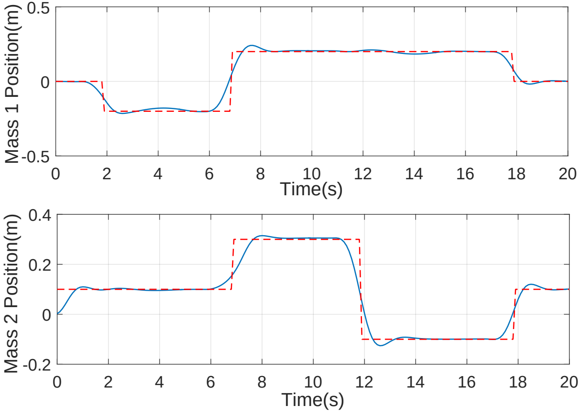


Figure: Mass Position Change with respect to Planned Trajectory. Red dashed line is the planned trajectory, and the blue line is the actual trajectory.

Emulation using Plant on Chip

Constraints on Force and its Rate of Change

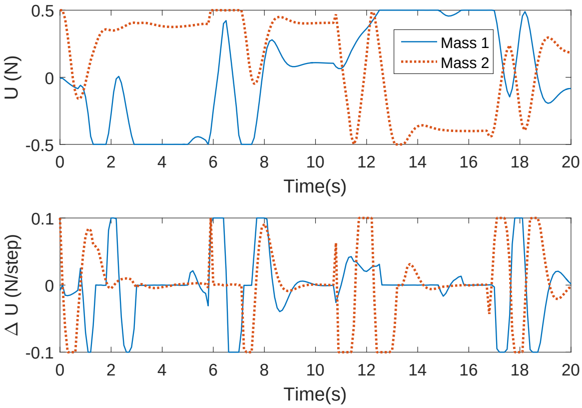


Figure: Control Signal U and ΔU. Blue line is the input force and the force rate of change for M_1, red dashed line is for M_2.

SW/HW Co-design

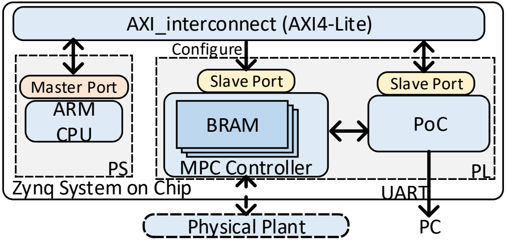


Figure: Top Level System Overview

Computation Speed Versus Hardware Resources

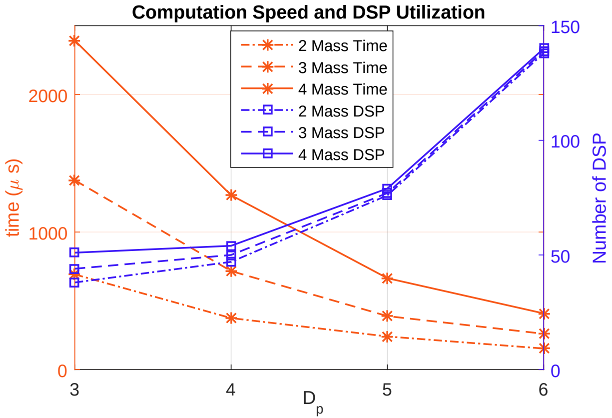


Figure: Computation time of 40 converge iteration loops and DSP usage for different system configurations from simulation. Computation time is marked by $*$, number of DSPs is marked by □. Hardware speed is 100MHz.

Resource Utilization

Table: Zynq-7020 Hardware Resource Usage

MVM Size D_p	Flip-Flops (106400 total)	LUTs (53200 total)	18Kb BRAM (280 total)	DSP48E (220 total)	Maximum Frequency
3	18147	12746	55	38	151.149MHz
4	21058	15103	87	47	144.885MHz
5	32425	23391	151	76	143.699MHz
6	57167	41273	279	138	133.298MHz

Timing Summary

Table: Hardware Computation Time per Iteration between Related Work.

	Method	Data Format	Chip Series	f_{clk}	#Multipliers	Iteration	#Opt Var	Running Time
This Paper	ADMM	floating-point	Zynq-7020	130MHz	72 (D_p=6, K=1)	40	204	314.2 μs
							350*	717.2 μs
					80 (D_p=5, K=2)		204	291.4 μs
			ZU9EG (Zynq UltraScale+)	340MHz	264 (D_p=8, K=1)			46.1 μs
					792 (D_p=8, K=3)			30.1 μs
HW[?]	ADMM	fixed-point	Virtex-6 (LX75)	400MHz	216 (K=1)	40	216	23.4μs
			Virtex-6 (SX475)		1512 (K=7)			4.90μs
HW[?]	IPM	floating-point	Virtex-7 (XC7VX485T)	200MHz	448	10	240	2,650 μs
SW[?]	ADMM	floating-point	Quad-core Intel Xeon	3.4GHz	n/a	35.1	525	3,400 μs

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References

John Smith (2012)
Title of the publication
Journal Name 12(3), 45 – 678.

The End