Communication Support for Task-Based Runtime Reconfiguration in FPGAs

Shannon Koh

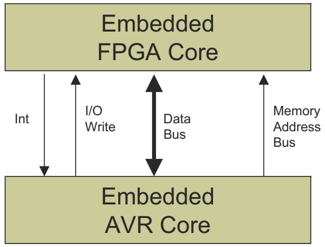
COMP4211 Advanced Computer Architectures Seminar
1 June 2005

Overview

- Motivation
- Model Definition
  - System architecture
  - Task model
  - Implementation model
- Research Direction

Motivation

- Today's FPGA-based systems run hardware/software partitioned applications
- Hardware modules sharing the reconfigurable resource
- Conceivably have dynamism during runtime



Embedded FPGA Core

Int

I/O Write

Data Bus

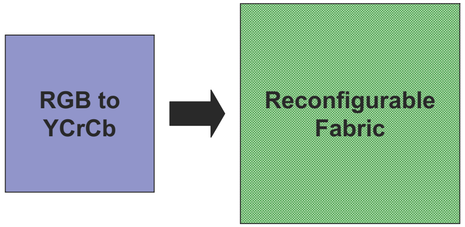
Memory Address Bus

Embedded AVR Core

Real Application: UAV Control System
Hardware: Repetitious, long time-scale functions e.g. command pulse
Software: Altitude and heading control

Dynamism – Single Task

- Hardware Virtualisation
  - Image Processing

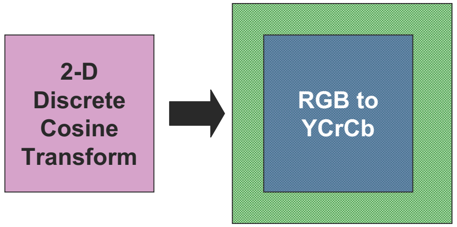


RGB to YCrCb

Reconfigurable Fabric

Dynamism – Single Task

- Hardware Virtualisation
  - Image Processing

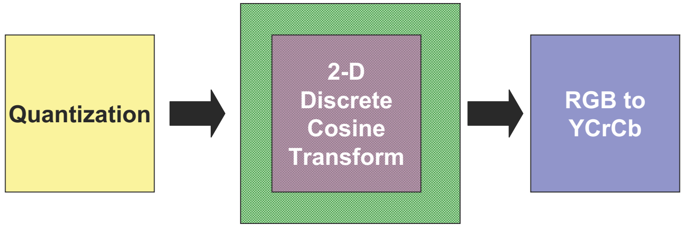


2-D Discrete Cosine Transform

RGB to YCrCb

Dynamism – Single Task

- Hardware Virtualisation
  - Image Processing



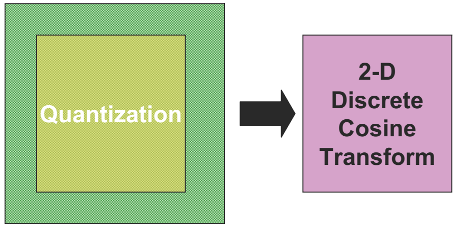
Quantization

2-D Discrete Cosine Transform

RGB to YCrCb

Dynamism – Single Task

- Hardware Virtualisation
- Power/QoS Tradeoffs
  - E.g. 16/32 bit realisation

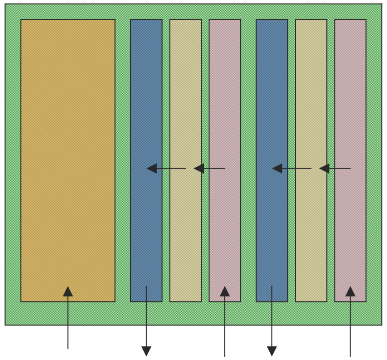


Quantization

2-D Discrete Cosine Transform

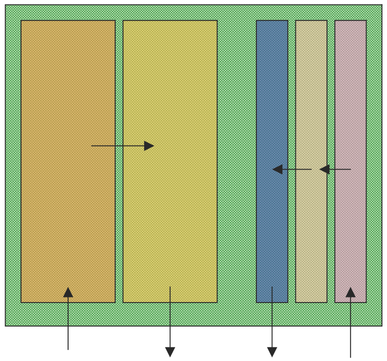
Dynamism – Multiple Tasks

- Pipelining/ Dataflow
- Multiple independent applications



Dynamism – Multiple Tasks

- Pipelining/ Dataflow
- Multiple independent applications

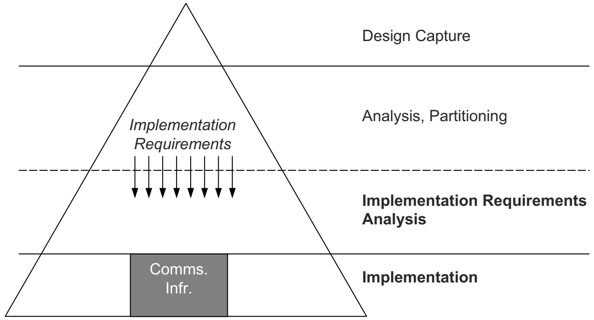


Challenge (Focus of my study)

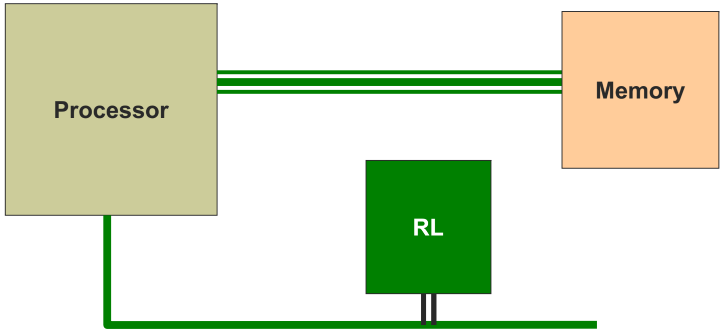
- How to support communication
  - Processor and reconfigurable logic tasks
    - HW/SW partitioning
  - RL tasks and other fixed components
    - Memory
    - I/O
  - RL tasks and other RL tasks
    - Pipelining

Overall Goal

- Design Flow Framework

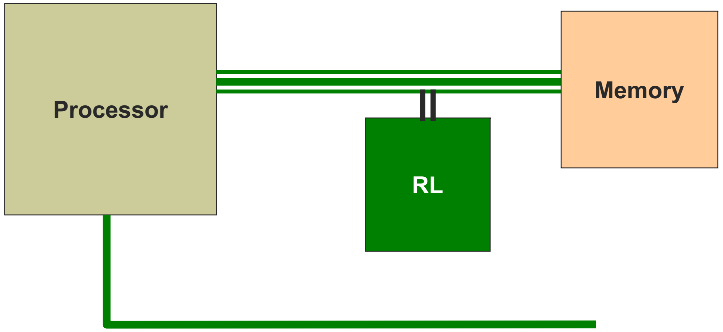

Design Capture
Analysis, Partitioning
Implementation Requirements
Implementation Requirements Analysis
Comms. Infr.
Implementation

Model Definition: System Model



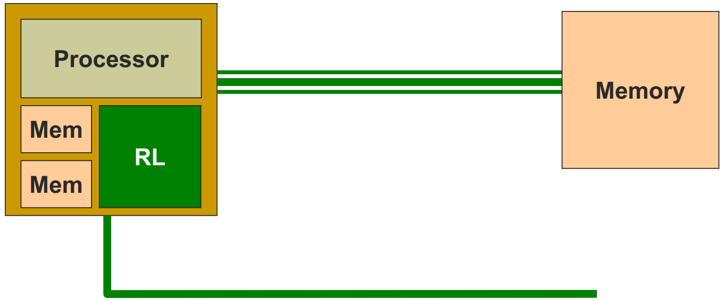
- Loose Coupling
- RL on IO/Peripheral Bus

System Model



- “Medium-Tightness” Coupling
- RL on Processor Bus

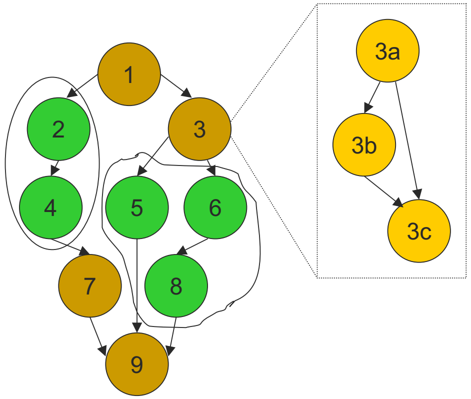
System Model



- Tight Coupling
  - Reconfigurable Systems-on-Chip
  - Platform FPGAs
- Model I will be considering

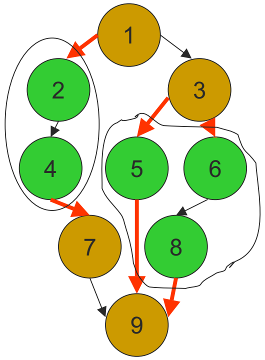
Model Definition: Tasks

- Task Flow Partitioning
  - Pipeline for 2 and 4
  - Parallel processing at 5 and 6



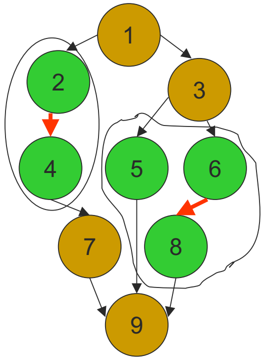
Communication Model

- SW to RL task communication
- Characterised by
  - Bit widths
  - Frequency
  - Control



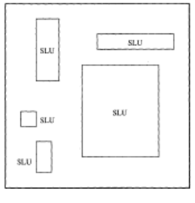
Communication Model

- RL to RL task
- Same characteristics apply?
  - Have to cater for possibility of task not being present later

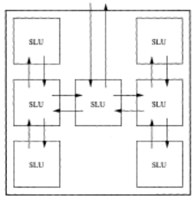


Implementation Model

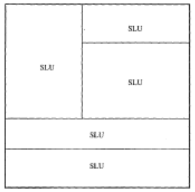
- Swappable Logic Units
- Advantages:
  - SOA: Dedicated routing
  - Parallel Harness: Routing and placement issues do not need to be considered
- Disavantages
  - SOA: Very difficult to realise dynamic routing, fragmentation
  - Parallel Harness: Less flexibility



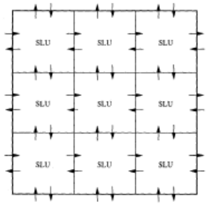
(a) Sea with five accelerators present, placed fairly randomly



(a) Parallel harness with extra H-tree wiring supplied



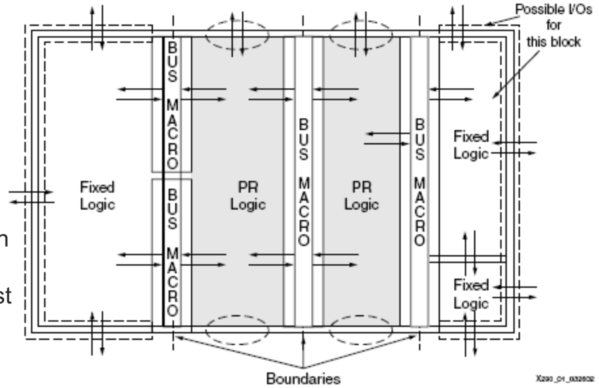
(b) Sea with five accelerators present, densely packed



(b) Parallel harness with no extra wiring needed

Xilinx Task-Based Reconfiguration

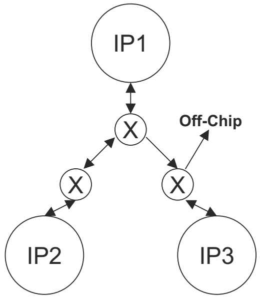
- Advantages:
  - Commercially available model
  - Realisable
- Disavantages:
  - No dynamic sizing and placement
  - Size and location in multiples of 4
  - Bus macros must be used
  - No parallel communication on same row
    - Passthroughs required

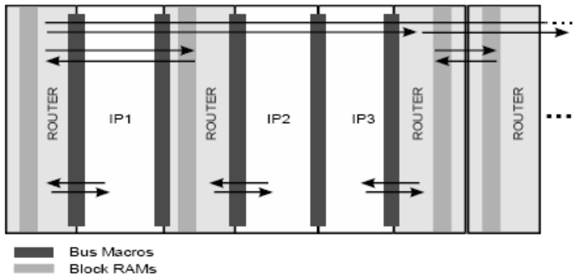


[Kalte04] Recent Study: Min 1 (S), 6 (M), 55(L)
Large XCV2000E: 80 columns (max 20 possible modules, about half are bus macros)

Network-on-Chip

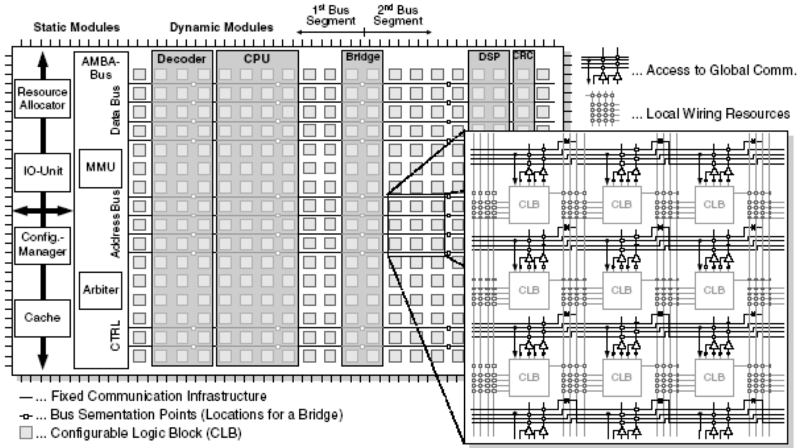
- Task wrappers and bus macros provide interfacing





Marescaux, T., Bartic, A., Verkest, D., Vernalde, S. and Lauwereins, R. (2002).
Interconnection Networks Enabled Fine-Grain Dynamic Multi-tasking on FPGAs.
In proceedings of the 2002 International Conference on Field-Programmable Logic.

1 Dimensional Task Model

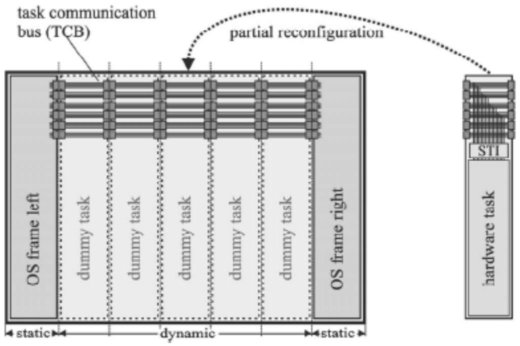


Static Modules
Dynamic Modules
1st Bus Segment
2nd Bus Segment
AMBA-Bus
Decoder
CPU
Bridge
DSP
CRC
Resource Allocator
Data Bus
IO-Unit
MMU
Address Bus
Config.-Manager
Arbiter
Cache
CTRL
CLB
... Access to Global Comm.
... Local Wiring Resources
... Fixed Communication Infrastructure
... Bus Sementation Points (Locations for a Bridge)
... Configurable Logic Block (CLB)

Kalte, H., Porrmann, M. and Rückert, U. (2004). ***System-on-Programmable-Chip Approach Enabling Online Fine-Grained 1D-Placement.*** *In proceedings of the 11th Reconfigurable Architectures Workshop 2004.*

Hardware Operating Systems

- Fixed-size pages, an example of parallel wiring harness



Steiger, C., Walder, H., and Platzner, M. (2004). ***Operating Systems for Reconfigurable Embedded Platforms: Online Scheduling of Real-Time Tasks.*** *IEEE Transactions on Computers, Vol. 53, No. 11, November 2004.*

Research Focus

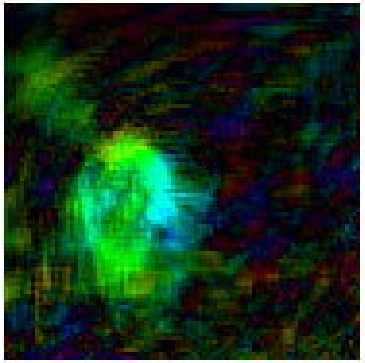
- Embedded systems
  - SoC with reconfigurable logic
- Applications (Partitions & Schedules)
  - Multiple hardware modules
  - Run-time dynamism
- Specific Applications
  - Optical flow algorithm
  - JPEG

Problem to solve:

- Given a partition and (dynamic) schedule, with known flows between components, how do we satisfy the communication requirements?

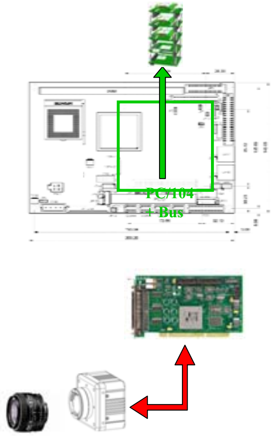
Optical Flow

- Determines velocity of pixels from frame to frame
  - Closer objects have higher relative velocity



System architecture

- Camera
  - 567x378 @ 27.4 fps
- Framegrabber
- Motherboard
  - P4-M 2.6GHz
  - 1024MB DDR 266
- BenNUEY board
- VirtexII XC2V6000

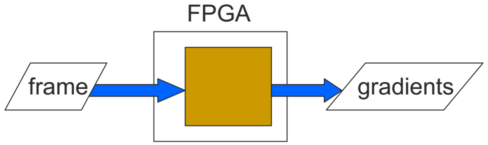


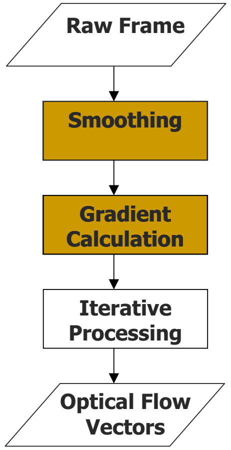
Optical Flow

Core iterative operation:

$$u_i^{k+1} = \frac{\sum\limits_{j\in N(i)} u_j^k - \frac{1}{\alpha}(I_{x,y,i}.v_i^k + I_{x,t,i})}{|N(i)| + \frac{1}{\alpha} I_{y,y,i}}$$

$$v_i^{k+1} = \ldots$$



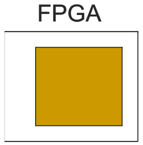


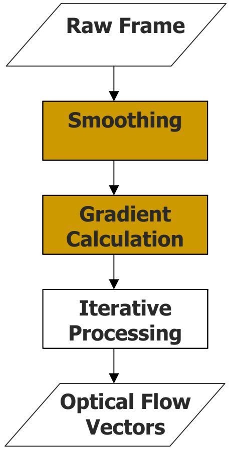
Optical Flow

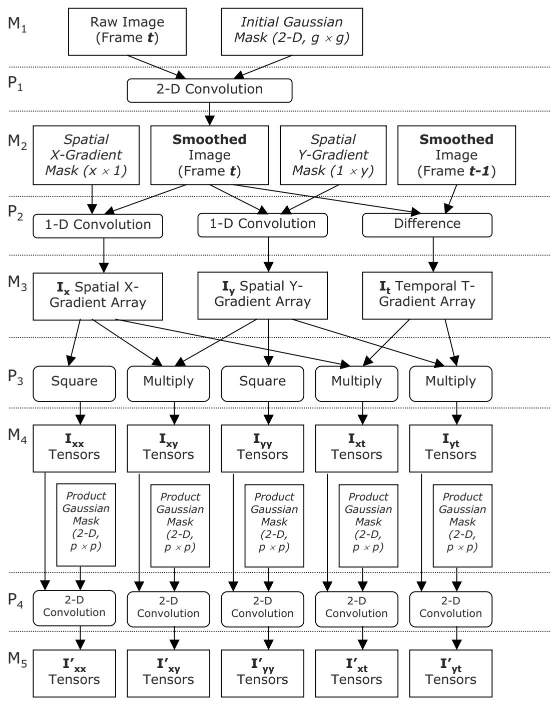
Core iterative operation:

$$u_i^{k+1} = \frac{\sum\limits_{j\in N(i)} u_j^k - \frac{1}{\alpha}(I_{x,y,i}.v_i^k + I_{x,t,i})}{|N(i)| + \frac{1}{\alpha} I_{y,y,i}}$$

$$v_i^{k+1} = \ldots$$





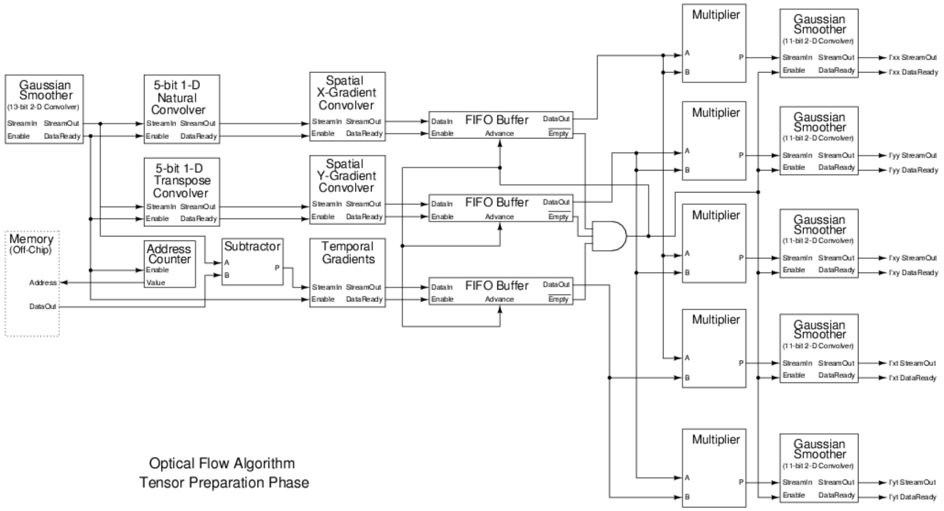

M1
Raw Image (Frame t)
Initial Gaussian Mask (2-D, g × g)
P1
2-D Convolution
M2
Spatial X-Gradient Mask (x × 1)
Smoothed Image (Frame t)
Spatial Y-Gradient Mask (1 × y)
Smoothed Image (Frame t-1)
P2
1-D Convolution
1-D Convolution
Difference
M3
Ix Spatial X-Gradient Array
Iy Spatial Y-Gradient Array
It Temporal T-Gradient Array
P3
Square
Multiply
Square
Multiply
Multiply
M4
Ixx Tensors
Ixy Tensors
Iyy Tensors
Ixt Tensors
Iyt Tensors
Product Gaussian Mask (2-D, p × p)
Product Gaussian Mask (2-D, p × p)
Product Gaussian Mask (2-D, p × p)
Product Gaussian Mask (2-D, p × p)
Product Gaussian Mask (2-D, p × p)
P4
2-D Convolution
2-D Convolution
2-D Convolution
2-D Convolution
2-D Convolution
M5
I′xx Tensors
I′xy Tensors
I′yy Tensors
I′xt Tensors
I′yt Tensors

Partitioning

- Module Partitioning
  - ○ Convolution Units
    - Horizontal (Row) Unit
    - Vertical (Column) Unit
  - ○ Multipliers
  - ○ Modularised Arithmetic

$$u_i^{k+1} = \frac{\left[\sum_{j \in N(i)} u_j^k\right] - \frac{1}{\alpha}\left[I_{x,y,i} . v_i^k + I_{x,t,i}\right]}{\left[\,|N(i)| + \frac{1}{\alpha} I_{y,y,i}\right]}$$

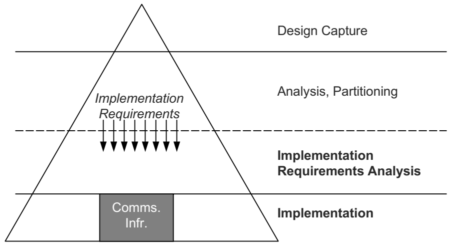
Implementation Requirements



Optical Flow Algorithm
Tensor Preparation Phase

Implementation Requirements

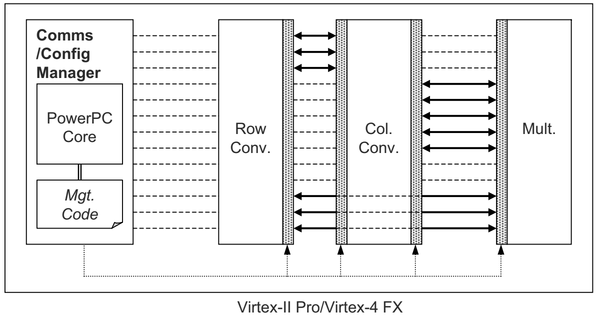
- Convolution Inputs/Outputs
  - ○ Bitwidths
  - ○ Module Timing


Design Capture
Analysis, Partitioning
Implementation Requirements
Implementation Requirements Analysis
Comms. Infr.
Implementation

Requirements Analysis

- Other Inputs e.g. device size
  - Number of communication lines per module
  - Time allowed per module
- Implementation Plan
  - Scheduling rules (not explicit schedule)
  - Communication modules

Implementation



Further Work

- Formal Definitions
- More Applications
- Dynamic Framework