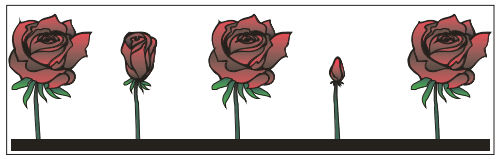
Automatic Configuration of Distributed Embedded Systems

A First Step Towards a Customization Manager

William Nace

Electrical & Computer ENGINEERING

RoSES



Motivation

- Embedded systems increasingly distributed
- Sensors/actuators gaining computing resources
- Substantial design time tradeoffs
  - # of Processing Elements?
  - How big are the PEs?
  - S/W task -> PE mapping?

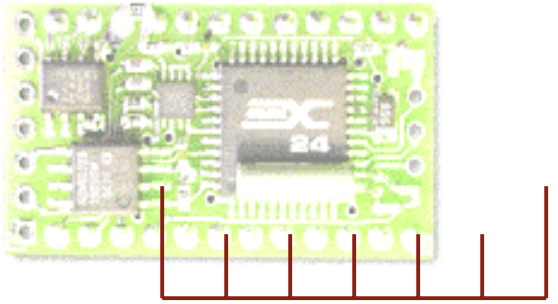
- Good first step for RoSES

The Challenge

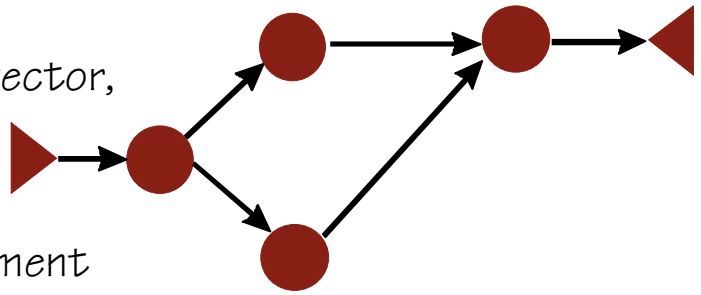
- Develop automated tool to
  - Map S/W tasks to the PEs
  - Specify size of PEs
- At all times
  - Respect network bandwidth constraints
  - Attempt to minimize cost
- Additional constraint
  - Sensor / actuator placement

Sub-Problem1: Allocating Tasks to PEs

- PE model
  - Vector of available resources
    - CPU Cycles, ROM, RAM, …
    - Resource levels are discrete
      - i.e. RAM can be 1K, 2K, 4K, …

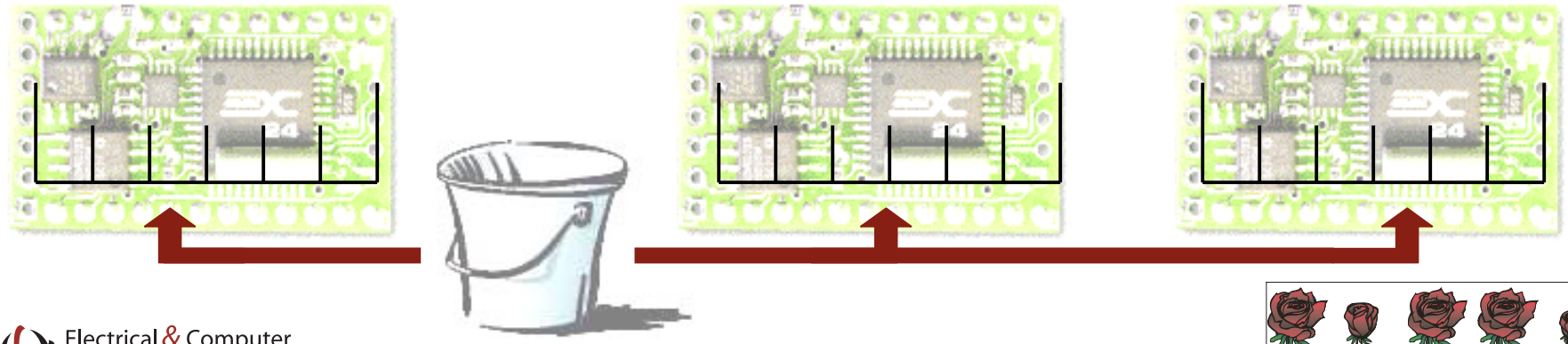


- Software model
  - Synchronous data flow graph
    - Graph nodes are tasks
      - Requirements specified as a vector, same terms as PEs
    - Arcs are communications
      - Labeled with dataflow requirement
    - Graph is directed, cyclic



Allocating Tasks to PEs (2)

- Network is modeled as a scalar bin
  - Size is the bandwidth of the net
  - Control Area Network (CAN) - 1 Mbps
    - 44 bit header / message
    - 0-8 byte payload
    - No contention (media access) overhead



Allocating Tasks to PEs (3)

- Bin-packing problem (NP-complete)
- Heuristic solution (Beck95/98)
  - 1) Sort tasks by size
  - 2) Iterate, starting with largest task
    - Pack task on the PE that minimizes use of network
    - Break ties randomly
  - 3) Declare success if
    - All task requirements do not overflow PE resources
    - All message objects fit in the bus's bin
      - Don't count messages between co-located nodes
  - 4) Fail otherwise

Interlude

- Two sub-problems are linked and should be solved simultaneously
  - How do you allocate tasks to hardware without knowing if it will fit or not?
  - How do you size the hardware without knowing what tasks will run on it?
- What happens when packing fails?
  - Possibly bad luck. Try again??
  - More likely, need to grow the H/W specification
    - Add another PE
    - Use bigger PE

Sub-Problem2: Specifying the H/W

- Start with minimum number of smallest PEs
- Begin packing tasks
- When packing fails, invoke design advisor (DA)
  - DA chooses from
    - Grow a PE
      - DA also picks which PE to grow
    - Add another PE

Specifying the H/W (2)

- The DA must balance:
  - Desire to keep PE small
  - Cost of a new PE
  - Change in usage of net BW (scarce resource)
- For each alternative, calculate ΔCost and ΔBW
- Balance ΔCost and ΔBW with aggressiveness factor, k
- Iterate the algorithm, changing k
- Keep lowest cost solution

Integrated Approach

- Sort task nodes by size
- For some set of K
  - Initialize PE collection to a single, small PE
  - Take largest remaining task node
    - Pack it on the PE which will minimize bus traffic
    - If it fails to pack anywhere, invoke the design advisor
  - Repeat until all nodes are packed
  - Measure cost of the solution. Save if cost is lowest so far

Approach (Design Advisor)

- Design advisor knows:
  - Current state of solution
  - Task that didn't pack
- Test the task against each PE available
  - If PE could be grown to fit the task, measure the ΔCost and ΔBW
- Test the node against a new PE
  - Measure ΔCost (apportioned) and ΔBW
- Choose solution with smallest ΔCost/ΔBW (balanced by k)

Adding Sensor/Actuators

- Designer doesn't have complete freedom as to task placement
  - Location requirements of sensors & actuators
- Sensors & actuators added to Beck's data set
  - Source tasks -> sensor
  - Sink tasks -> actuators

Adding Sensor/Actuators (2)

- Each S/A has an inclusion / exclusion list
  - Specify other S/A that must (not) be collocated
  - Experiment -- Randomly selected
- Additional "sensor clustering" phase before task sorting
  - Combine sensors into groups which act like a single sensor
  - Requirements calculated as sum of sensor group
- Allow packing to proceed normally

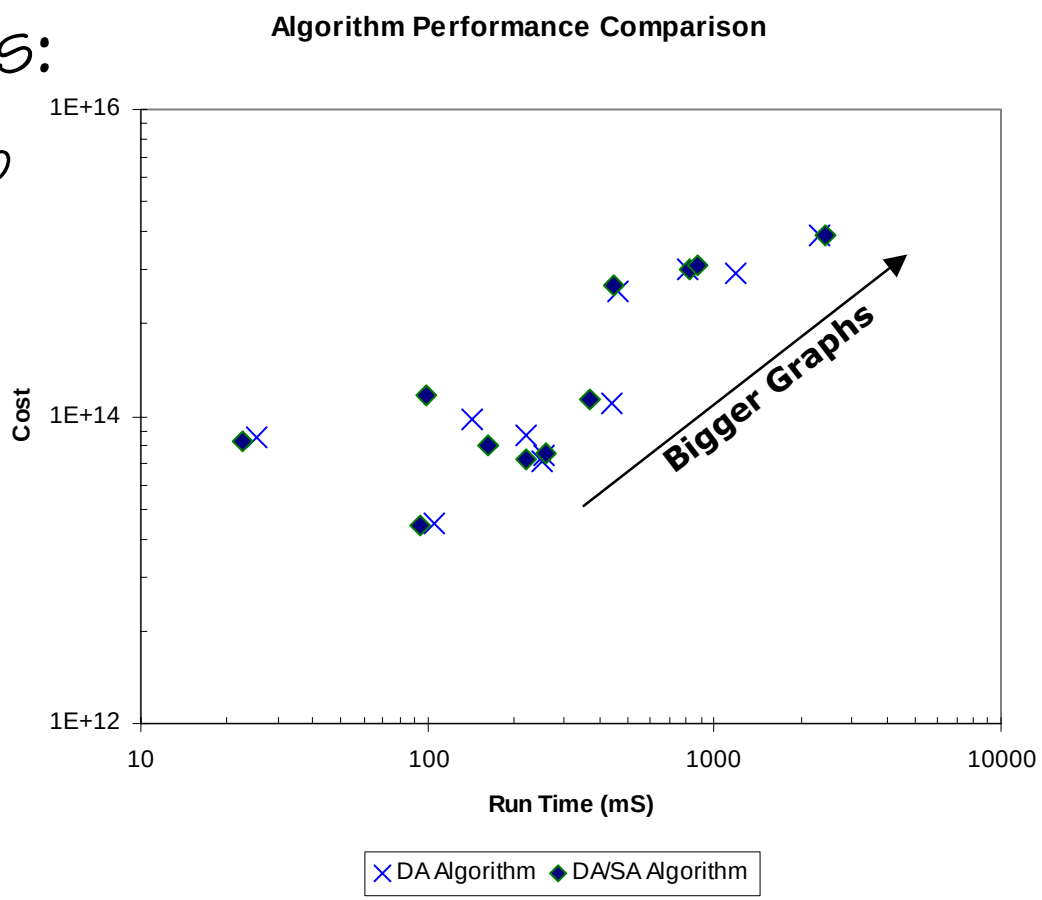
Input Data

- Beck95/98 confirmed utility of algorithm using 16 data sets
  - Random & Synthetic
  - Real
    - Automotive
- 5 discarded as pathological

Data Set	Nodes	#S/A	Avg #Grps	Avg Grp Size
Ran01	100	20	2	4.9
Ran02	100	20	2	6.9
Ran05	50	28	4	6.7
Ran06	200	196	7	5.6
Ran07	50	8	2	3.3
Ran08	75	32	4	4.1
Ran09	90	73	7	5.7
Ran10	25	5	2	2.0
Ran11	100	99	7	2.1
Syn01	8	5	2	2.0
Traction	160	117	7	4.9

Results

- S/A clustering shows:
  - 15% runtime speedup
    - × moves left to ◆
  - 3% cost penalty
    - Slight rise visible



Future Work

- RoSES Configuration Manager
  - Multi-algorithm sensor/actuators
  - Functional optimization
    - Many algorithms (task graphs)
  - Integration of Real-Time scheduling (net & PEs)
  - Metrics to determine how much slack to apportion at design time

Summary

- Designer of Embedded Distributed System faces simultaneous solution of linked problems
  - Assigning tasks to Processing Elements
  - Specifying size of Processing Elements
- Specification of sensor and actuator clusters
  - Adds realistic constraint
  - Makes fast estimator even faster
    - Managing clusters as a unit reduce number of objects