

Electrical Architecture Optimization and Selection - Cost Minimization via Wire Routing and Wire Sizing

Chung-Wei Lin
University of California

Lei Rao and Joseph D'Ambrosio
General Motors Co.

Alberto Sangiovanni-Vincentelli
University of California

ABSTRACT

In this paper, we propose algorithms for cost minimization of physical wires that are used to connect electronic devices in the vehicle. The wiring cost is one of the most important drivers of electrical architecture selection. Our algorithms perform wire routing from a source device to a destination device through harnesses, by selecting the optimized wire size. In addition, we provide optimized splice allocation with limited constraints. Based on the algorithms, we develop a tool which is integrated into an off-the-shelf optimization and workflow system-level design tool. The algorithms and the tool provide an efficient, flexible, scalable, and maintainable approach for cost analysis and architecture selection.

CITATION: Lin, C., Rao, L., D'Ambrosio, J., and Sangiovanni-Vincentelli, A., "Electrical Architecture Optimization and Selection - Cost Minimization via Wire Routing and Wire Sizing," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 7(2):2014, doi:10.4271/2014-01-0320.

INTRODUCTION

As automotive in-vehicle feature complexities increase, so do the architectures implementing them. Designing, evaluating, and selecting different architectures is a difficult task that cannot be done manually. In this work, we propose algorithms for cost minimization of architectures where the cost drivers include the size of physical wires and their routing through the harness. In the early phases of E/E architecture definition and selection, decisions about different architecture topologies are determined among others by wire routing and sizing, and splice allocation. We have gathered the cost models for the wiring harness architecture are based upon ad-hoc Excel-based tools which are limited in their exploration capabilities. Because of scalability limitations, designers' experience leads to manually trimming the design space which only provides suboptimal solutions. Also, wire sizing is done by a manual trial-and-error approach. Furthermore, the allocation of splices is also done manually according to designers' experience. To deal with these problems, we propose efficient algorithms to enable an automatic exploration of a larger design space.

There are several existing tools [1, 2] for the electrical and wiring harness design. The Capital Suite [1] provides, among other features, a design flow from logical connectivity designs to manufacturing-ready harness designs for transportation platforms such as aircraft and vehicles. Especially, the Capital Integrator can automatically synthesize wiring objects and generate fully-detailed wiring designs. The software also provides interfaces for cost optimization and design validation. E3.series [2] is another tool for creating wiring designs.

In this work, we focus on wire routing and wire sizing and consider splice allocation. We formulate a problem based upon different (architecture, circuit, physical wire) architecture abstractions. To solve the problem, we propose efficient data structures and algorithms to perform wire routing, wire sizing, and splice allocation. We implement the proposed algorithms as a MATLAB-based tool which can be integrated into an off-the-shelf optimization and workflow system-level design tool (e.g., Model Center by Phoenix Integration [3]). In the system-level design tool, users can create workflows by wrapping design models and apply optimization methods and third-party tools to perform trade studies or optimize designs.

A case study inspired by a real industrial design has been completed, which shows that the algorithms and the tool can provide an efficient, flexible, scalable, and maintainable approach for cost analysis and architecture selection.

DEFINITIONS AND FORMULATIONS

Our model is based upon the following artifacts.

Harness Model

Definition 1 (part): A *part* is an ECU, a sensor, an actuator, or a splice.

Definition 2 (hyper-edge): A *hyper-edge* is a set of ECUs, sensors, and actuators.

In other words, a hyper-edge is a set of parts excluding splices.

Definition 3 (architecture): An *architecture* consists of a set of parts and a set of hyper-edges connecting those parts.

Figure 1(a) shows an architecture and its equivalent graphical view. There are two hyper-edges, {Part A, Part B, Part C} and {Part A, Part D, Part E}, meaning that they must be connected in the architecture.

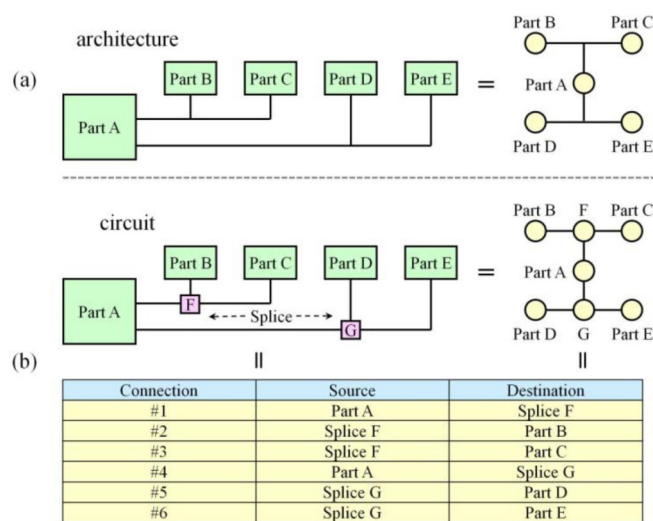


Figure 1. An architecture and a circuit which consists of six connections.

Definition 4 (connection): A *connection* connects two parts.

Definition 5 (terminal): A *terminal* is an end point of a connection.

Definition 6 (circuit): A *circuit* consists of a set of connections.

Figure 1(b) shows a circuit and its equivalent graphical view. The circuit consists of six connections listed in the table. For the hyper-edges {Part A, Part B, Part C} and {Part A, Part D, Part E} in Figure 1(a), Splice F and Splice G are used to connect the three parts, respectively. For each connection, we

need to connect its corresponding parts and decide its wire size and terminal size. We will explain this in the section of parameters, constraints, and cost functions.

Definition 7 (harness): A *harness* is the complete wire assembly.

Definition 8 (location): A *location* of a harness is a place where wires can get in or out of the harness.

An example harness model is shown in Figure 2 with two locations. Two parts are connected by a wire, and the wire goes through the harness. The end points of a harness are usually also locations, and a harness can have several branches.

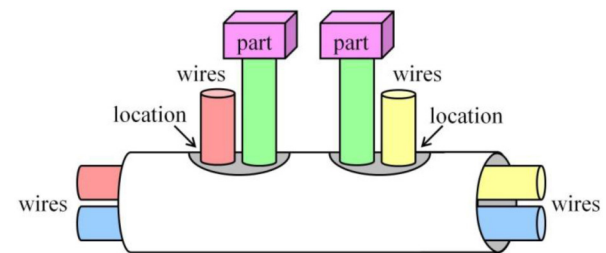


Figure 2. A harness model with its locations; two parts are connected by a wire, and the wire goes through the harness.

Definition 9 (routing graph): A *routing graph* consists of a set of parts, a set of harnesses, and a set of edges where an edge connects (1) between a part and a location or (2) between two locations of the same harness.

An example of routing graph is shown in Figure 3, which is also the test case used in our case study.

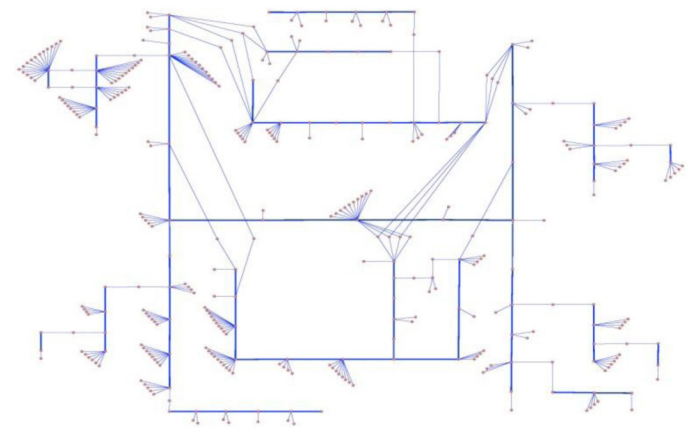


Figure 3. A routing graph.

Definition 10 (routing path): A *routing path* is a path in the routing graph connecting two parts.

Definition 11 (cutlead): The number of *cutleads* of a routing path is the number of harnesses the routing path goes through.

Mathematical Definitions

We also provide mathematical definitions of above terms. As defined earlier, every connection in a circuit connects exactly two parts, while a hyper-edge in an architecture can connect more than two parts.

Definition 12 (set of parts and set of splices): The set of parts is defined by V_P , and the set of splices is defined by V_S (V_S is a subset of V_P).

Definition 13 (architecture): An architecture A is defined by a hyper-graph (V_A, E_A) where V_A is a subset of V_P/V_S , E_A is the set of hyper-edges, and each hyper-edge is a subset of V_A .

Definition 14 (connection): A connection N is defined by v_N and v'_N where v_N and v'_N are in V_P .

Definition 15 (circuit): A circuit C is defined by a graph (V_C, E_C) where V_C is a subset of V_P , E_C is the set of edges, and each edge connects two parts in V_C .

Definition 16 (harness): A harness H is defined by a graph (V_H, E_H) where V_H is the set of locations in the harness, E_H is the set of edges, and each edge connects two locations in V_H .

Definition 17 (set of locations): Assuming there are n harnesses $H_1, H_2, \dots, H_n$, the set of all locations is defined by V_L which is the union of $V_{H_1}, V_{H_2}, \dots, V_{H_n}$.

Definition 18 (routing graph): Assuming there are n harnesses $H_1, H_2, \dots, H_n$, a routing graph is defined by (V, E) where V is the union of V_P and V_L , and E is the union of $E_{H_1}, E_{H_2}, \dots, E_{H_n}$ and the set $\{(v, v') \mid (v \text{ in } V_P, v' \text{ in } V_L) \text{ or } (v \text{ in } V_L, v' \text{ in } V_P)\}$.

Formulations

We also need to define the different types of implementations. Because all edges are undirected, edges (v, v') and (v', v) are equivalent.

Definition 19 (connection implementation): A routing path implements a connection if their (the routing path's and the connection's) source and target parts are the same.

Definition 20 (circuit implementation): A set of routing paths implements a circuit if every connection in the circuit is implemented by a routing path in the set.

Definition 21 (hyper-edge implementation): A set of edges $E'_C = \{(s_1, t_1), (s_2, t_2), \dots, (s_n, t_n)\}$ implements a hyper-edge $\{v_1, v_2, \dots, v_m\}$ in E_A if, for any pair of vertices (v_i, v_j) , $i \neq j$, there exists a path from v_i to v_j where only v_i, v_j , vertices in V_S , and edges in E'_C can be involved in the path.

Definition 22 (architecture implementation): A circuit (V_C, E_C) implements an architecture (V_A, E_A) if $V_C = V_A$ and every edge in E_A is implemented by a disjoint subset of edges in E_C .

For example, the architecture in Figure 4(a) is implemented by the circuit in Figure 4(b) where P_7 is a splice used to connect P_1, P_5 , and P_6 . Given the routing graph in Figure 4(c), the circuit in Figure 4(b) is implemented by the five routing paths in Figure 4(d) where each routing path implements a connection of the circuit (blue: (P_1, P_7) ; green: (P_5, P_7) ; red: (P_6, P_7) ; orange: (P_3, P_5) ; purple: (P_6, P_9)).

Definition 23 (feature): A feature is a requirement to be designed in a vehicle.

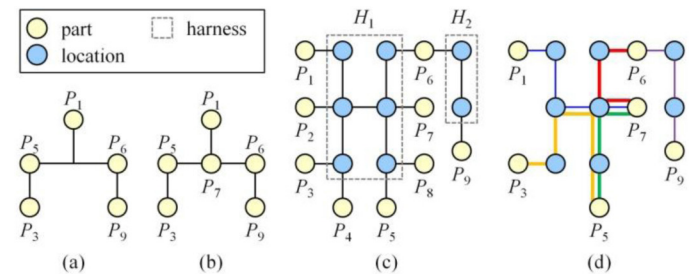


Figure 4. (a) An architecture. (b) A circuit which implements the architecture where P_7 is a splice. (c) A routing graph with 9 parts, 2 harnesses, and totally 8 locations. (d) A set of routing paths which implement the 5 connections of the circuit (blue: (P_1, P_7) ; green: (P_5, P_7) ; red: (P_6, P_7) ; orange: (P_3, P_5) ; purple: (P_6, P_9)).

Definition 24 (feature implementation): For each feature, there is a pre-defined set of architectures. Each architecture in the set is an alternative implementation of the feature.

The problem here is to decide an optimal architecture to implement a feature. To evaluate the cost of an architecture, we have to find its cost from the perspective of the harness model. We divide the problem to three sub-problems:

- Problem I: Given a circuit, find minimal-cost routing paths which can implement the circuit.
- Problem II: Given an architecture, find a minimal-cost circuit which can implement the architecture.
- Problem III: Given a feature, find a minimal-cost architecture which can implement the feature.

Problem I involves wire routing and wire sizing for each connection of a circuit. A minimal-cost path means a cheapest path, typically shortest but not always. Problem II is a more general problem which can provide close-to-optimal locations for splices. It is basically a Steiner tree problem which is NP-Complete. We do not discuss about Problem III in details and leave it as a future work. In this work, we will regard an architecture as an input and focus on Problems I and II.

Parameters, Constraints, and Cost Functions

Each edge in a routing graph is associated with one value, representing its length. As mentioned earlier, an edge in a routing graph connects between a part and a location or between two locations of the same harness. There are two types of edges, and each of them connects two locations in