AUTOMATING THE DESIGN PROCESS: A DESIGN PROCESS MODEL IMPLEMENTATION

Lorna Carmichael
U.S. Army Research Laboratory
Ft. Monmouth, N.J. 07703
Attention: AMSRL-EP-MA
lorna@monmouth-etd1.army.mil

Abstract

In 1990, Andrzej Wnuk of Siemens A.G. proposed a comprehensive model of the system design process. The model combines an extended design AND/OR decision tree, aspects of the design process for the High Level Design Assistant (HILDA) developed by Siemens Research and Technology Laboratory, structured requirements, and process decomposition. A suitable requirements description language did not exist which resulted in a model which could not be implemented, that is, until now. The recent development of the MIMIC (Microwave and Millimeter Wave Integrated Circuits) Hardware Description Language (MHDL) relieves that problem. Hierarchical, functional, and object oriented features of the language make it ideal for modeling the system design process, as well as capturing/cataloging the system and its components as they progress towards implementation. This paper reports on the implementation of a design process model and the associated impacts (i.e. automation of the design process; the creation of an homogeneous design environment; complete, historical, electronic documentation, etc.). Additionally, solutions/migration paths for upgrading existing systems will be discussed.

I. INTRODUCTION.

The size and complexity of electronic systems forces one to search for an easier way to design and develop them. Analysis of specification and requirements, subsystem & component design, decision support, and eventual system implementation should be supported in a consistent manner. A methodological means to capture and communicate system requirements and design information amongst diverse disciplines, people, and the tools they use, makes the development of a design process model more of a necessity rather than a desire; it also implies that a standard means of communicating such information is required. Additionally, translation of design specifications and requirements into quantities manageable by computer aided engineering systems and environments is an implied requirement. Although the concepts and philosophies of automation and a virtually paperless society are prevalent in the engineering community, automating all aspects of the design process is not practical. Intelligent application and careful consideration must be given to the for each system design task in order to result in a significant, verifiable improvement in the system design process. In 1990 “an appropriate requirement description language did not exist.” Today, a hardware description language exists and is capable of complete system support, from specification & requirements capture, through design, development, production, and maintenance.

The desires for such a language are not new. Wnuk desired a requirements description language[1]; DeMan of Bell Telephone Mfg Co, Belgium reported on the use of a functional language to describe systems[2]; Narayan, Vahid and Gajski of University of Irvine identified the requirement and use of an executable specification language[3]. Several IEEE and DoD aim to accomplish similar goals (i.e. AVHDL, TRSL). The MIMIC Hardware Description Language was designed and develop to meet the above requirements (a simulatable/executable language capable of documenting and capturing requirements, system descriptions and detailed designs). One of the major objectives of MHDL was/is to be an universally accepted, technology independent language, serving as a “living description” throughout the system design process, and capable of support throughout the entire system life. The language is to be robust enough to convey information in sufficient detail (such as, electrical and mechanical functional and detailed hardware description, budgeting, and scheduling data), for complete system support.

427

U.S. Government Work Not Protected By U.S. Copyright.
II. MHDL OVERVIEW.

MHDL is an IEEE effort to establish a common language for the communication, documentation, and design of electronic systems. The original effort began under ARPA and DoD sponsorship in 1989 and is now under the auspices of IEEE Standards Coordinating Committee (SCC) 30 for further development and standardization. An implementation of the language and tool development are currently sponsored by ARPA.

A hardware description language (HDL) is a formal standardized mean of representing physical designs and the information associated with the designs. MHDL and its associated tools allow you to easily manage large amounts of engineering data, generated and/or manipulated by many diverse parties. MHDL unifies many different disciplines associated with the development of a piece of hardware (i.e. mechanical, analog, digital, software design and cost analysis). A design philosophy, such as "top down" or "bottom up" is not imposed on the user by MHDL, but full hierarchical and askew design methodologies are supported. MHDL serves as a living document, capturing design rationale as products are developed. It literally is "virtual hardware", allowing new technologies to be inserted and their impacts analyzed prior to production.

MHDL is a functionally based, object-oriented language, borrowing most of its concepts from the functional language Haskell. Functional languages treat functions the same as any other parameter; functions can be returned as values. MHDL meets the criteria for being an object-oriented language; it supports polymorphism, inheritance(multiple), it is hierarchical, and is modular. MHDL was designed syntactically to use terminology familiar with the end users. Some of the major features and elements of the language are: Models: which represents the design entity; Structures: describes the contents of models and their associated connections; Connectors: the external connections of the model; Signals: values with physical units associated with them; Constraints: specifies an assertion to be tested; Alternatives: one or more mutually exclusive selections. MHDL was designed with the goal of providing a means to make the task of system design, documentation and subsequent maintenance cost effective, time efficient and technically superior. The language provides the necessary infrastructure to support this objective, practical application will achieve it.

III. WNUK MODEL OVERVIEW

Andrezj Wnuk developed a model of the system design process which utilized a design decision support mechanism. The decision support mechanism could conceivably be an advanced, intelligent CAD tool or the user/system engineer himself. The model is appropriately designed to accept decisions support to complex for CAE system. The design process is divided up into design task. Design tasks are generated from the overall system requirements and represents an activity to be accomplished. The design task is the single entity modeled and can be recursively used to represent each aspect and level in the design hierarchy. There are six input and output connection on the design task; Ispec: Weighted input specifications for current task; Ospec: Output specifications for subtasks; lrep: Requirements fulfillment reports from subtasks; Orepl: Requirements fulfillment reports from current task; lstate: Current state of task received form parent task; Ostate: Current state of task sent from parent task. Wnuk represents the design task as a nine tuple having the form:

\[
\text{Design}_\text{task} = [\text{Ispec, Ospec, lrep, Orepl, lstate, Ostate, Nodes, State, Operations}]
\]

Where State is the global state of the design task (it is not a connector because it is merely data). Nodes represent internal states in the design task and can be one of three possible choices; state node, choice node, or version node. Operations represent activities which change the internal states of a design task. The following operations are defined for the design task/system process model:

Extraction: acquiring information about the task; Construction: actually using or "instantiating" the model (the model now has a real physical counterpart that exists or can be associated with it); Generating Alternatives: creating alternatives; Decomposition: decomposing the design object into components and interconnects; Generating Subtask Specifications: defining subtask requirements; Synthesis: aggregating the design objects into one component/design object; Evaluation: comparing design results against task specification; Selection of next task: Selecting the next design task; Change subtask requirements: modification of alternatives or optimization or modification of subtask requirements; Report Preparation: Report preparation in support of decision on state for tasks.
IV. MODEL IMPLEMENTATION.

The model to support the entire system design process utilizes the Wnuk model and MHDL constructs. The Wnuk model, which captures "system engineering level" information (that is, to a component designer), is implemented in MHDL and the remaining portions of the model, which represent detailed design information, use primitive MHDL constructs. Constructs and terminology for MHDL will be in bold italic print and Wnuk constructs in italic print.

A. Wnuk Model.

The entire task is represented as a model; the design-task is the encapsulating mechanism for the Wnuk model and corresponds directly to a model in MHDL. The model has connectors which represents input/output (Ispec, Ospec, Irep, Orep, Istate, Ostate) of the design object. However, only three connectors (spec_con, rep_con, state_con) are provided in the MHDL implementation; distinction between inputs and outputs in the Wnuk model were actually due to parent and child relationships. These relationships are represented by attributes. Attributes and signals, properties on the connectors, represent information traversing the connectors. At requirement and specification stages, connectors also represent the logical flow of information. Nodes in the wnuk model represent internal design states. The function of nodes have been extended to mean a place where the design-task can be queried and decision support provided. Nodes will occur at each internal connector. The distinction between the three nodes, choice, state, and version, are also irrelevant in this implementation. Attributes on the connectors of the design-task model are functions which return the following information: Ispec: a function which gathers/specifications/requirements for current task and has an optionally applied weight function; Ospec: function which gathers/specifications/requirements from parent task and produces a set of requirements/specifications for all subtasks; Ireport: requirement fulfillment reports of all subtasks; Oreport: requirement fulfillment reports of current task; Istate: global state of current task; Ostate: global state of all subtask. The attributes are composed of operations defined by Wnuk and those operations are represented as primitives in MHDL. The Operations and their MHDL implementations are as follows:

extraction: get more information for global or current task by processing or revising task specifications;
extract_zw = primitive "extract.decision";

The process of elaboration in MHDL "flattens" the design hierarchy;

The model itself is quite simple now:

model wnuk_design_task
includes wnuk_pack;
structure wnuk_struct
   connectors spec_con, rep_con, state_con;
end wnuk_struct;
end wnuk_design_task;

As the model evolves towards a physical entity, additional information (i.e. definitions, comments, functions, etc. utilizing MHDL) is added to the model and/or operations are applied. This information includes detailed design information necessary for simulation and production. The operations are necessary to progress through the design process.
When utilizing the model, each connector must be explicitly typed; this is due to the concept of shadow networks. Connections which indicate the logical flow of information, as in the allocation of requirements or the generation of alternative designs, are sometimes meaningless when the models are instantiated. Also, they (logical design partitions) may not correspond directly to their physical implementation. However, the logical and physical design representations are most definitely related. By typing each connector in the MHDL model, this relationship is maintained were appropriate and shadow network(s) can be easily created using the type of the connectors as a filter.

B. Example.
The example is taken from [1]. Suppose the following design task was given:

1. Design the cheapest computer system with a processor speed no less than 0.5 MIPS
2. Design an alternative system with a processor speed at least 1.2 MIPS and cost less than $1000.00

The initial requirements for the system would be inputted into the MHDL tool, an initial guess of $1500 is assumed:

- Computer system Total cost = $1500
- CPU speed >= 0.5 MIPS
- CPU speed >= 1.2 MIPS and CPU cost < $1000

The MHDL model information would look like:

```java
model Computer
    includes wnuk_design_task;
    structure Computer_struct
        connectors computer_child;
        constraint Total_cost: Computer.cost <= 1500
            report "cost exceed allocated amount";
        constraint CPU_design:
            (CPU_speed >= 0.5) ||
            (CPU_speed >= 1.2) &&
            (CPU_cost <= 1000);
    end Computer_struct;
end Computer;
```

Applying `gen_sub_specs` operation adds the line `gen_sub_specs` (Computer); to the model and results in four new models being created for Disk, Bus, RAM, and CPU, and inheriting the same constraints. Component declarations for Disk, Bus, RAM, and CPU will also be added to `model Computer`. Although the terms have physical meanings, we are dealing only with specifications now and therefore only logical connections.

---

Fig. 1. MHDL Package for Wnuk model.

---

430
Applying `gen_alternate` to `CPU` results in two processors, `proc1` and `proc2`:

```mhdl
model CPU
  includes Computer;
  structure CPU_struct
    connectors CPU_child, CPU_parent;
  components
    proc1 :: ;
    proc2 :: ;
  end components;
  connect CPU_parent, computer_child;
  connect CPU_child proc1_parent,
    proc2_parent;
  gen_alternate (CPU);
end CPU_struct;
end CPU;
```

The specifications (constraints) in `model Computer` are automatically inherited by including `model Computer`. The components have no definition because only specifications exist.

Working with the first processor, `proc1`, generate subtask specifications operation is applied, resulting in `models` for `processor`, `ROM`, and `BusCtrl`. Up until now, all connections have been logical, although there is physical parity. If the `decompose` operation was now applied, the `connectors` would have to be explicitly typed and how the processor is intended to physically connect to the control bus and `ROM` would be added to the model. Each subsequent execution of an operation generates more MHDL code.

A detailed design model of the computer could also be represented in VHDL (VHSIC Hardware Description Language). VHDL also readily integrates and interfaces to MHDL without difficulty and must obey the following conventions: a VHDL entity architecture pair is created and the name of that pair is defined as an attribute in MHDL; the connectors must be of the same name and type.

V. CONCLUSION.

The integration of a hardware description language for specification/requirements capture, allocation, and analysis, complex system and component design (MHDL), with an established hardware description language for digital applications (VHDL) supports the entire system design process. Additionally, a homogeneous design environment is created. Now, and for the twenty first century, information is power; those whom effectively manage and use it will be the ones to contend with. In a knowledge based society, the quality of information flow determines the degree of communication, competitiveness and ultimately, success. MHDL is a precise, unambiguous language and a pending international standard. Clearly new strategies must be developed that facilitates entrance into the market place, technology transfer, and reduces engineering costs in order to compete in a global economy. Strategies are required throughout the commercial market, but especially in engineering environments, where the deployment of rapidly changing technology (computer hardware, for instance) is the key to success. Effective utilization of MHDL technology facilitates technology transfer and will result in a superior, low cost product with a hyper-extended system life. Additionally, MHDL allows sound concurrent engineering practices to be exercised and captures design rationale.

The impact and applications of MHDL technology are numerous, limited only by industry’s willingness to capitalize on it. Introducing new technology usually brings about a level of skepticism and risk. The benefits of MHDL are enormous and increase with time, positively impacting a company’s competitiveness, market presence, products, etc., in essence, the bottom line. The prospect of such novel technology should spur us all to embrace and take advantage of MHDL.

REFERENCES.

