HANDBOOK OF

Measuring System

DESIGN

EDITORS **PETER H. SYDENHAM** **RICHARD THORN**

ARTICLE
OFFPRINT

69: Requirements Allocation

Andrew Kusiak

The University of Iowa, Iowa City, IA, USA

Fang Qin

The University of Iowa, Iowa City, IA, USA

1 INTRODUCTION

Design is information-processing activity resulting in the creation of an object. An early design stage is referred to as *conceptual design* (Pahl and Beitz, 1988) – *see* **Article 64, Executing A Measuring System Design, Volume 2** and **Article 66, Phases of System Life Cycle, Volume 2**. In conceptual design, design requirements are transformed into a functional and then physical description. At an early design stage, requirements are formed and then allocated to functions. This article covers the methodology.

To date, relatively little research has been done in the area of design specifications. Kota (1990) developed a function-decomposition hierarchy to identify a set of basic design building blocks. Qualitative functional specifications and synthesis for conceptual design of microelectromechanical systems are presented in Crary and Kota (1990). A methodology for hydraulic circuit synthesis based on functional and structural design building blocks is presented in Kota and Lee (1990). Kannapan and Marshek (1990) presented procedures for specification and synthesis using algebraic and logical transformation rules. Rinderle and Finger (1990) used a graph-based language to describe the behavioral specifications of a design as well as the behavior of the available physical components. Morrell (1988) described the concept of quality function deployment (QFD) applied to improve product quality based on customer requirements.

2 REQUIREMENTS FORMING

Design *requirements are* 'demands' and 'wishes' that clarify the design task in the space of needs (Pahl and Beitz, 1988). They provide an abstraction of the design task from the most general demand (the overall requirement) to more specific demands (subrequirements). A design *function* indicates the behavior that is required for the device to satisfy a given requirement (Kota and Ward, 1990). Requirements and functions are domain specific and represent part of the knowledge base of the design system. A designer, as the needs or his/her design experience changes, can also add them.

A design task is given overall requirements that are domain dependent. They are further decomposed into a set of subrequirements. The number of levels of requirements depends on the complexity of the design task.

Handbook of Measuring System Design, edited by Peter H. Sydenham and Richard Thorn.

ISBN: 0-470-02143-8.

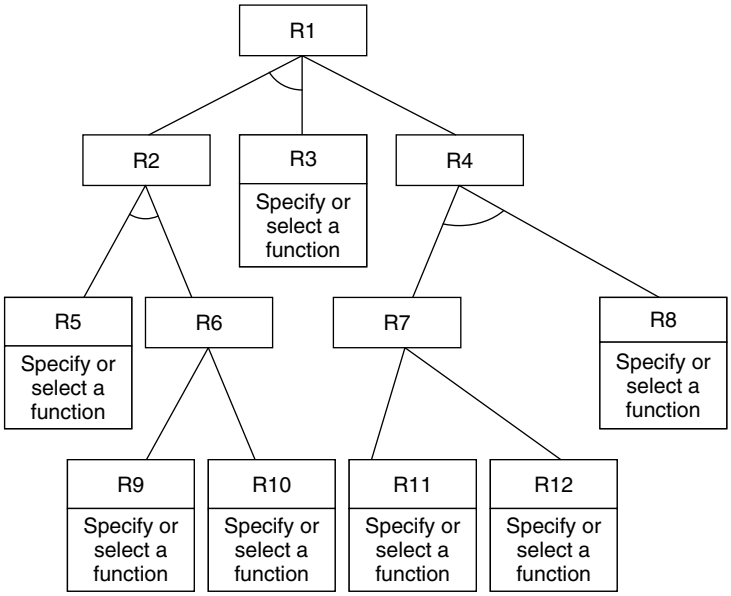

Figure 1. Decomposition of requirements.

Figure 1 shows the overall requirement R1, its decomposition into subrequirements, and the corresponding functions (Kusiak and Szczerbicki, 1992).

An arc between nodes of the tree in Figure 1 represents a conjunction. A node without an arc represents a disjunction. In the example presented in Figure 1, the overall requirement R1 is satisfied by each of the following four sets of subrequirements:

1. {R11, R8}
2. {R12, R8}
3. {R10, R5, R3}
4. {R9, R5, R3}

Each of the above sets may lead to a different conceptual design. Some such designs are discussed in Kusiak (1999).

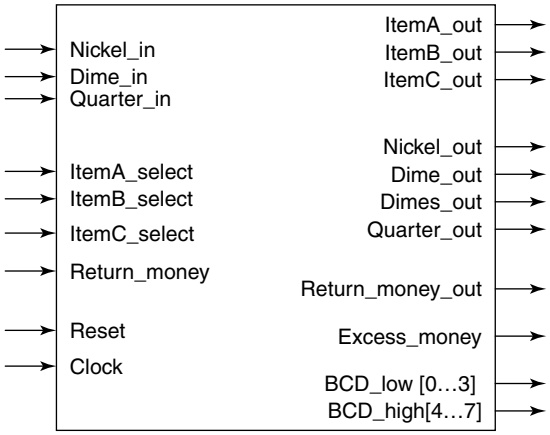

Figure 2. Model of a vending machine controller.

3 DECOMPOSITION

Decomposition is useful in analysis of requirements and functions (Kusiak 1999). It reduces the complexity of the problem at hand. The application of decomposition will be demonstrated here using the design of a vending machine controller. The controller of a typical vending machine has 9 inputs and 11 outputs as shown in Figure 2.

The vending machine needs to be designed for purchasing items, for example, A, B, C, by a providing a change and then selecting an appropriate button to get the item. If the change exceeds the price of the item selected, the vending machine returns the difference. If the Return_money button is selected, the controller will return all the coins deposited since the last transaction (Return_money_out will be activated). The machine accepts the money only up to 95 cents. If more change is inserted, then the Excess_money output variable activates a bypass chute and extra change inserted is routed directly to the change return. Two 4-bit binary coded decimal code combinations represent the current amount of money (in cents) the user has deposited in the vending machine.

4 SOLUTION APPROACH

Having formed the design requirements for the vending machine, the next step is to analyze the various Product Functions. The main functions are now discussed.

1. **Coin counting**
   The function 'count coins' of the vending machine is illustrated in Figure 3.
2. **Money return**
   The vending machine returns change if the button of Return_money is selected (see Figure 4).
3. **Excess money**
   The excess of 95 cents is returned (see Figure 5).
4. **Item selection**
   When the customer selects an item and the amount deposited in the vending machine is equal to or greater than the price of the selected item, the item is delivered (see Figure 6).
   Note: Item_select includes itemA_select, itemB_select, itemC_select.
   Item_out includes itemA_out, itemB_out, itemC_out.
5. **Change calculation**
   If the customer drops in more money than the price of the item selected, the change needed is calculated. The change equals the total amount deposited in the vending machine minus the price of the selected item (see Figure 7).
6. **Change out**
   After the vending machine has calculated the amount of change, the proper change amount is returned to the customer (see Figure 8).
7. **BCD indicator**
   The vending machine needs to indicate the current amount of money (in cents) the user of the vending machine has deposited with two 4-bit BCD code combinations (see Figure 9).

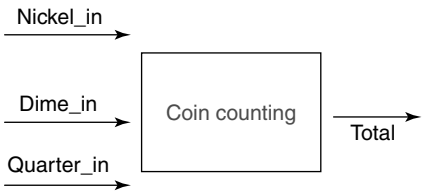

Figure 3. Coin counting.

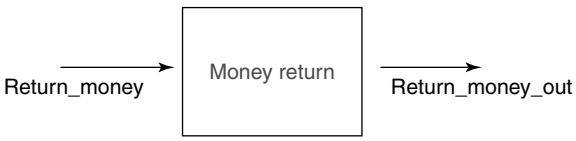

Figure 4. Return money.

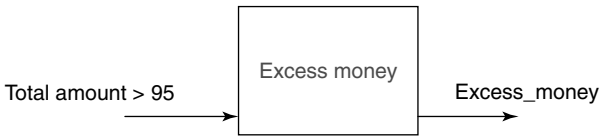

Figure 5. Excess money.

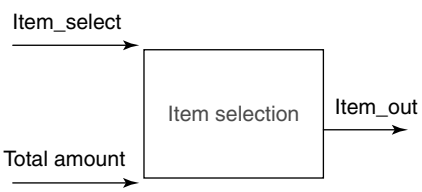

Figure 6. Item selection.

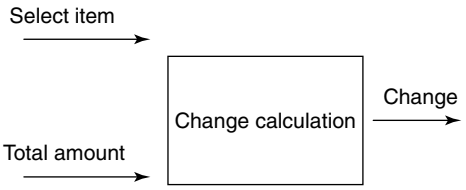

Figure 7. Change calculation.

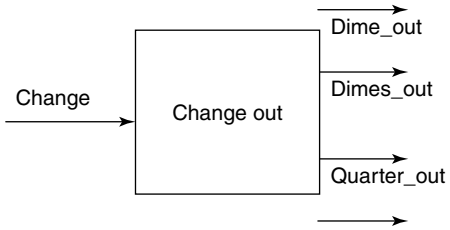

Figure 8. Change out.

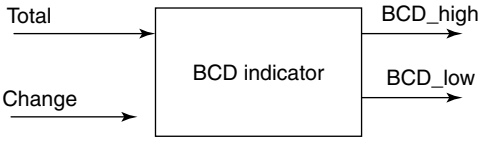

Figure 9. BCD indicator.

5 STRUCTURAL DECOMPOSITION

On the basis of the functions defined in Figure 2 through to Figure 9, the function-input/output matrix is formed (see Figure 10).

Each "*" in Figure 10 indicates a relationship between the corresponding function i, $i = 1, \ldots, 7$ and input/output j, $j = 1, \ldots, 16$.

A list of functions is obtained from the rows in Figure 10:

1. Coin counting
2. Return money
3. Excess money
4. Item selection

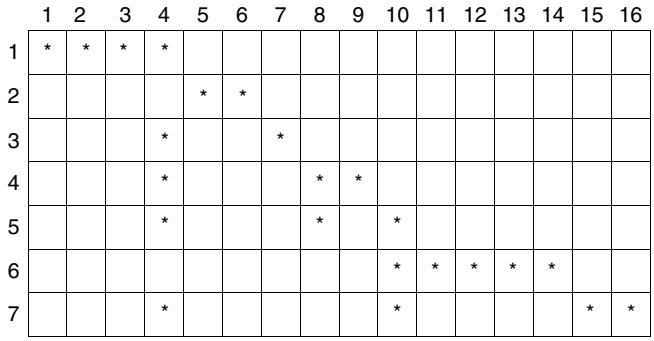

Figure 10. Function-input/output incidence matrix.

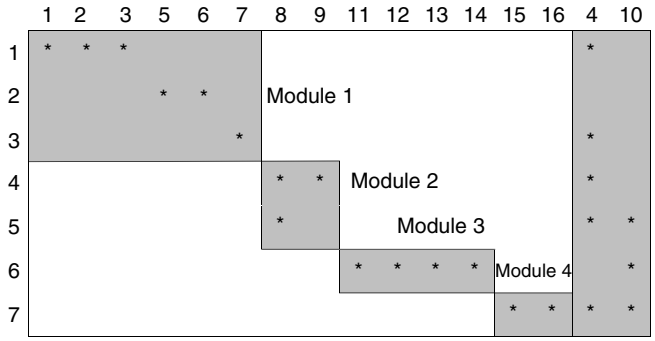

Figure 11. Function-input/output matrix of vending machine clustered with the branch-and-bound algorithm.

5. Change calculation
6. Change out
7. BCD indicator.

List of inputs/outputs (columns in Figure 10):

1. Nickel_in
2. Dime_in
3. Quarter_in
4. Total_amount
5. Return_money
6. Return_money_out
7. Excess_money
8. Item_select (includes ItemA_select, ItemB_select, ItemC_select)
9. Item_out (includes ItemA_out, ItemB_out, ItemC_out)
10. Change
11. Nickel_out
12. Dime_out
13. Dimes_out
14. Quarter_out
15. BCD_high
16. BCD_low.

Clustering the matrix with the branch-and-bound algorithm (Kusiak, 2000) results in a new matrix (Figure 11) that is more suited to design allocation. Four groups of functions – {1, 2, 3}, {4, 5}, {6} and {7} – and four groups of inputs/outputs – {1, 2, 3, 4, 5, 6, 7}, {8, 9}, {11, 12, 13, 14} and {15, 16} – are visible in Figure 11. Inputs/outputs 4 and 10 interact with more than one function.

Introducing redundant inputs/outputs for performing the functions allows decomposition of Figure 11 into the four mutually separable submatrices shown in Figure 12 (for details see Kusiak (1999)).

6 MODULE DESIGN

The four function modules are introduced to fulfill the requirements from Figure 12. VHDL logic is now used to describe the behavior of each module (Perry, 1994).

Module 1: Coin handler

A coin handler counts coins, calculates the total amount, returns money, and handles excess money according to the VHDL logic presented next and the function illustrated in Figure 13.

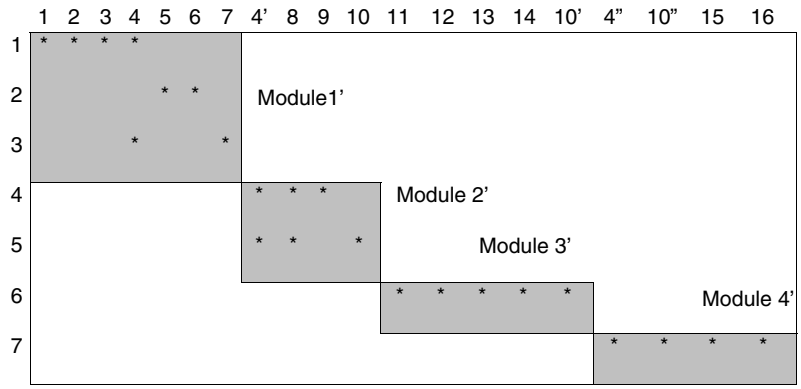

Figure 12. Structured function-input/output matrix of vending machine after introduction of redundant inputs/outputs.

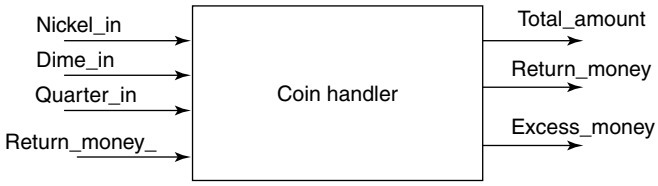

Figure 13. Coin handler.

1. **Count coins**

```
IF nickel_in = '1' THEN
  total_amount := total_amount + 05;
END IF;
----------
IF dime_in = '1' THEN
  total_amount := total_amount + 10;
END IF;
----------
IF quarter_in = '1' THEN
  total_amount := total_amount + 25;
END IF;
----------
```

2. **Return money**

```
IF return_money = '1' THEN
  return_money_out = '1';
END IF;
```

3. **Handle excess money**

```
IF total amount > 95 THEN
  excess_money = '1';
END IF;
```

Module 2: Item processor

The Item processor accepts purchase requests and determines if the money entered is enough to purchase an item (see Figure 14).

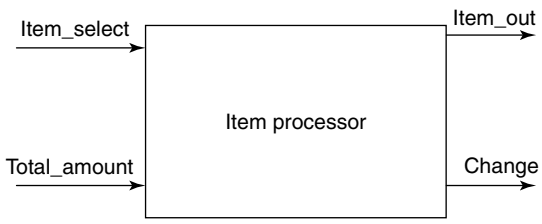

Figure 14. Item processor.

Item	Price(cents)
A	55
B	65
C	75

Figure 15. Item price list.

The price of each of the three items A, item B, and item C is listed in Figure 15.

1. **Select Item A**

```
IF ItemA_select = '1' THEN
 IF total_amount >= 55 THEN
  ItemA_out = '1';
  Change := total_amount −55;
 END IF;
END IF;
```

2. **Select Item B**

```
IF ItemB_select = '1' THEN
 IF total_amount >= 65 THEN
  ItemB_out = '1';
  Change := total_amount −65;
 END IF;
END IF;
```

3. **Select Item C**

```
IF ItemC_select = '1' THEN
 IF total_amount >= 75 THEN
  ItemC_out = '1';
  Change := total_amount −75;
 END IF;
END IF;
```

Module 3: Change maker

Change maker outputs provide change to the customer (see Figure 16).

The input/output of the change maker is shown in Figure 17:

The following VHDL statements illustrate the process of handling a change of 35 cents.

```
IF change = 35 THEN
  Nickel_out = '1';
  Dime_out = '1';
  Dimes_out = '1';
  Quarter_out = '0';
  Total_amount := 0;
  Change := 0;
END IF;
```

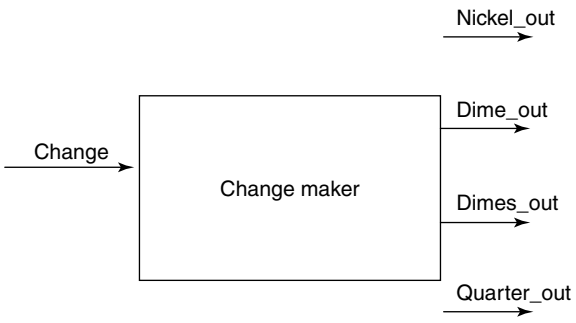

Figure 16. Change maker.

Change (cents)	Nickel_out (5 cents)	Dime_out (10 cents)	Dimes_out (20 cents)	Quarter_out (25 cents)
0	0	0	0	0
5	1	0	0	0
10	0	1	0	0
15	1	1	0	0
20	0	0	1	0
25	1	0	1	0
30	0	1	1	0
35	1	1	1	0
40	1	1	0	1

Figure 17. Input/output table of the change maker.

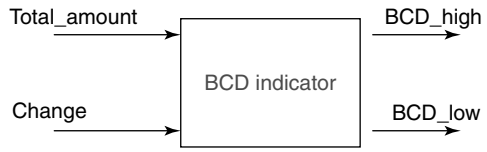

Figure 18. BCD indicator.

Module 4: BCD indicator

Two 4-bit BCD code combinations represent the current amount of money (in cents) the user of the vending machine has deposited. Before purchase, the amount equals the total amount, after purchase, the amount equals the change (see Figure 18).

The input/output of change maker is presented in Figure 19.

Suppose the amount is 35, the following VHDL sequence is used:

```
IF amount = 35 THEN
  BCD_LOW = ''0101'';
  BCD_HI = ''0011'';
END IF;
```

7 SYSTEM DESIGN

7.1 Module interaction

From Figure 11, it was determined that input/output 4 (total amount) and input/out 10 (change) interact with multiple modules. The relationship of the modules is obtained from Figure 11 and shown below. Thus, the inner connections of modules are established (see Figure 20).

Amount (cents)	BCD_high	BCD_low
0	0000	0000
5	0000	0101
10	0001	0000
15	0001	0101
20	0010	0000
25	0010	0101
30	0011	0000
35	0011	0101
40	0100	0000
45	0100	0101
50	0101	0000
55	0101	0101
60	0110	0000
65	0110	0101
70	0111	0000
75	0111	0101
80	1000	0000
85	1000	0101
90	1001	0000
95	1001	0101

Figure 19. Input/output of BCD indicator.

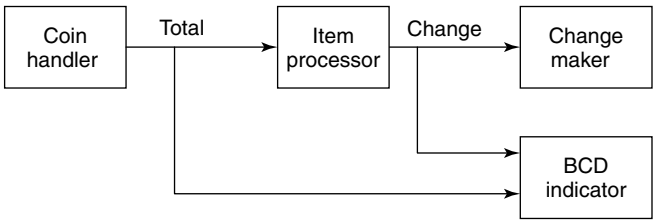

Figure 20. Inner connections of modules of vending machine.

8 VENDING MACHINE CONFIGURATION

After the module design is finished and the inner connections established of modules, the top-level vending machine configuration is obtained. Clock and Reset are added to synchronize the operation of each module (see Figure 21).

This process has systematically worked its way through the many options. By reorganizing the relationship matrix, several units have been identified that have less interfaces to other units than was first developed. With less interfaces, the chance of system failure is reduced and design made simpler. Measuring systems can similarly be broken down in this manner. More information on the nature and scope of requirements is found in Sage and Rouse (1999) and Sydenham (2004).

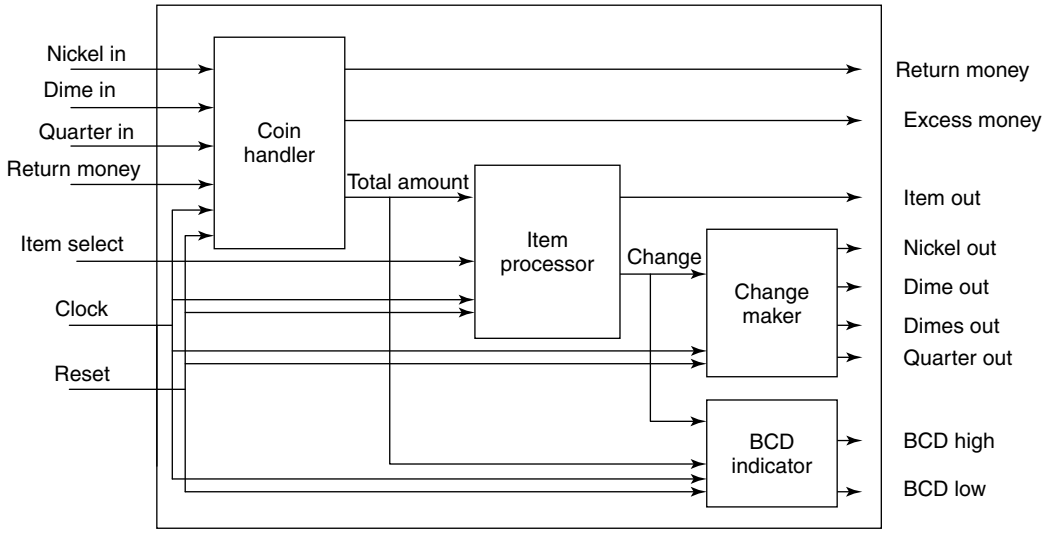

Figure 21. Vending machine.

REFERENCES

Crary, S. and Kota, S. (1990) Conceptual Design of Micro-Electro-Mechanical Systems, in *Proceedings of the Micro Systems Conference*, Berlin.

Kannapan, S.M. and Marshek, K.M. (1990) An Algebraic and Predicate Logic Approach to Representation and Reasoning in Mechanical Design. *Mechanism and Machine Theory*, **25**, 335–353.

Kota, S. (1990) Qualitative Motion Synthesis: Towards Automating Mechanical Systems Configuration, *Proceedings of the NSF Design and Manufacturing Systems Conference*, Society of Manufacturing Engineers, Dearborn, MI, (pp. 77–91).

Kota, S. and Lee, C.-L. (1990) A Computational Model for Conceptual Design: Configuration of Hydraulic Systems, *Proceedings of the NSF Design and Manufacturing Systems Conference*, Society of Manufacturing Engineers, Dearborn, MI, (pp. 93–104).

Kota, S. and Ward, A.C. (1990) Functions, Structures, and Constraints in Conceptual Design, in *Proceedings of The 2nd International Conference on Design Theory and Methodology*, (ed. J. Rinderle), Chicago, IL, (pp. 239–250).

Kusiak, A. (1999) *Engineering Design: Products, Processes, and Systems*, Academic Press, San Diego, CA.

Kusiak, A. (2000) *Computational Intelligence in Design and Manufacturing*, Wiley, New York.

Kusiak, A. and Szczerbicki, E. (1992) A Formal Approach to Specifications in Conceptual Design. *ASME Transactions: Journal of Mechanical Design*, **114**, 659–666.

Morrell, N.E. (1988) Quality Function Deployment; Disciplined Quality Control. *Automotive Engineering*, **9**, 122–128.

Pahl, G. and Beitz, W. (1988) *Engineering Design*, Springer-Verlag, New York.

Perry, D. (1994) *VHDL*, McGraw-Hill, New York.

Rinderle, J.R. and Finger, S. (1990) A Transformational Approach to Mechanical Design Synthesis, *Proceedings of the NSF Design and Manufacturing Systems Conference*, Society of Manufacturing Engineers, Dearborn, MI, (pp. 67–75).

Sage, A.G. and Rouse, W.B. (eds) (1999) in *Handbook of Systems Engineering and Management*, (ed. A. Sage), Wiley, New York.

Sydenham, P.H. (2004) *Systems Approach to Engineering Design*, Artech House, Boston, MA.