HEWLETT PACKARD

A Fuzzy Inference Design on Hewlett-Packard Logic Synthesis System

Raymond L. Chen*
HP Laboratories Palo Alto External Research Program
HPL-93-70
August, 1993

fuzzy logic
synthesis, digital
integrated circuits

A set of hierarchical fuzzy logic system components are created and integrated into the HPL's Tsutsuji logic synthesis system. As a demonstration, these components are applied to simulate a fuzzy inference control system of an inverted pendulum. Special graphical virtual instruments are also created that allow the designer to tune the system at a high level.

*Electronics Research Laboratory, University of California, Berkeley, California

Table of Contents

List of Figures

List of Tables

1 Introduction

The Tsutsuji system, also known as the Hewlett-Packard Logic Synthesis (HPLS) system, is a new system for digital hardware design which includes many features to speed the development of new products [Culbertson93]. HPLS transforms the high-level block diagram design to gate-level layout and netlist which can be implemented in various technologies. HPLS has been jointly developed by Hewlett-Packard Laboratories (HPL) in the U.S.A. and Yokagawa-Hewlett-Packard Design Systems Laboratory (YSL) in Japan. The HPLS system is named Tsutsuji, which is the Japanese name for azalea, a popular flower in Japan.

HPLS offers fast and efficient ways to do hardware logic design, simulation and optimization [Culbertson93, YHP91]. Its graphical human interface allows a design to be simulated interactively on an X-window workstation environment. The inputs of a design can be dynamically adjusted though window widgets ("virtual instruments") which imitate real physical instruments. The topology graph generated from a design shows graphically the critical path on a gate-level.

The objective for this project was to add a fuzzy logic-based inference module to the Tsutsuji system.

1.1 Motivation

Area of Investigation:

> The motivation for this project was to expand our repertoire of logic synthesis blocks to include ones specifically applicable to fuzzy logic based control systems. The goal was to add functionality to synthesize the digital logic of a fuzzy logic based control system such as a camera focus control system or an automobile automatic transmission shift control.

Specific tasks:

- Identify building blocks used to implement fuzzy logic systems. These blocks will be integrated into the logic synthesis system at the same level as existing basic blocks (adders, multipliers, registers, etc.).
- Ascertain how to create parameter-driven fuzzy logic building blocks. For example, the existing carry-lookahead adder logic synthesis module can synthesize an unlimited number of designs as a function of operand bus widths and "tuning" parameters such as carry-group size.
- Explore the possibility of system-level fuzzy logic building blocks. For example, a parameterized, tunable fuzzy logic control system block composed of basic fuzzy logic modules.
- Provide a convenient way for the application domain expert to specify the parameters necessary for the synthesis of the fuzzy logic building blocks.

1.2 Report Outline

This report first reviews some background knowledge of the Fuzzy logic methodology (Section 2), then describes the Tsutsuji diagram design for the fuzzy inference systems with one and two input variables, respectively (Section 3). A description about the programming structure of Tsutsuji, including the socket interface communication, and Virtual Instrument (VI) widgets, is given in Section 4. As a result of this project, a prototype fuzzy inference system for balancing an inverted pendulum is created (Section 5). The report is concluded with a summary and a few suggestions/ comments for the future improvement of Tsutsuji and the fuzzy system.

Basically, three main tasks were carried out in the project: the block design for fuzzy logic components on Tsutsuji, the specific VIs suitable for fuzzy inference, and the connections between designs and VIs through sockets in the simulation.

2 An Introduction to Elementary Fuzzy Logic

The concept of Fuzzy Logic was first introduced by Professor Lotfi A. Zadeh [Zadeh65] of the University of California at Berkeley, in June 1965. It was not very well-known to the science and technology community in U.S. until recent years. In the last few years, however, the subject has flourished and applications of this theory can now be found in many disciplines. In this section, we will explain the basics of fuzzy logic and a fuzzy inference decision-making system. For brevity, only a subset of the fuzzy logic theory will be introduced as it pertains to this project.

2.1 Fuzzy sets and membership function

Fuzzy logic is based on the concept of fuzzy set [Zimmermann91]. The fuzzy set theory is in many ways a generalization of the classical set theory. A *classical* (crisp) set A is normally defined as a collection of elements or objects $x \in X$ which satisfy certain conditions. Each single element $x \in X$ can either belong to or not belong to the set A, where $A \subseteq X$. To generalize this definition, we can introduce a membership function μ (on X) for each element to specify its relation to the crisp set A, i.e. $\mu(x)=1$ if $x \in A$ and $\mu(x)=0$ if $x \notin A$. If this membership function is continuous, a *fuzzy* set B can be defined as a collection of elements $x \in X$ with membership function $\mu(x)$, where $\mu(x)$ can be any real number between 0 and 1:

$$B = \{ (x, \mu_B(x)) \mid x \in X \} \qquad (2\text{-}1)$$

2.2 Fuzzy logic and rules

In Boolean logic, the most basic logic operations are "PASS", "COMPLEMENTARY", "AND" and "OR". The rules of those operations can be expressed in the following table:

Table 1. Boolean logic rules

Name	Rule	Equivalent Fuzzy Membership Value
PASS	IF $x \in A$, THEN $z \in Z$	IF $\mu_A(x)=1$, THEN $\mu_Z(z)=1$
COMPL.	IF $x \notin A$, THEN $z \in Z$	IF $\mu_A(x)=0$, THEN $\mu_Z(z)=1$
OR	IF $x \in A \vee y \in B$, THEN $z \in Z$	IF $\mu_A(x)=1$ OR $\mu_B(y)=1$, THEN $\mu_Z(z)=1$
AND	IF $x \in A \wedge y \in B$, THEN $z \in Z$	IF $\mu_A(x)=1$ AND $\mu_B(y)=1$, THEN $\mu_Z(z)=1$

where "1" can be defined as "true" and "0" as "false".

Fuzzy logic can be regarded as a generalization of the classical Boolean logic by allowing the "membership function" $\mu(x)$ to be any number between 0 and 1. Equivalently, Boolean logic is a special case of the fuzzy logic. Thus, the equivalent fuzzy logic rules ("fuzzy rules") can be defined for the above four basic logic operations as follows:

Table 2. Fuzzy logic rules

Name	Rule	Membership Value
PASS	IF $(x, \mu_A(x)) \in A$, THEN $(z, \mu_Z(z)) \in Z$	$\mu_Z(z) = \mu_A(x)$
COMPL.	IF $(x, \mu_A(x)) \notin A$, THEN $(z, \mu_Z(z)) \in Z$	$\mu_Z(z) = 1-\mu_A(x)$
OR	IF $(x, \mu_A(x)) \in A$ OR $(x, \mu_B(x)) \in B$, THEN $(z, \mu_Z(z)) \in Z$	$\mu_Z(z) = \max(\mu_A(x), \mu_B(x))$
AND	IF $(x, \mu_A(x)) \in A$ AND $(x, \mu_B(x)) \in B$, THEN $(z, \mu_Z(z)) \in Z$	$\mu_Z(z) = \min(\mu_A(x), \mu_B(x))$

where all of the values of membership functions μ's are between 0 and 1.

2.3 The concept of the linguistic variable

A linguistic variable has a name, and a set of *linguistic* values [Zimmermann91]. Each of these linguistic values is associated with a membership function. For example: a person's *age A* (a linguistic variable) can be a linguistic variable having linguistic values such as *juvenile, young, middle-aged, old, very old* with membership functions for all ages between 10 and 100 as shown in Figure 1.

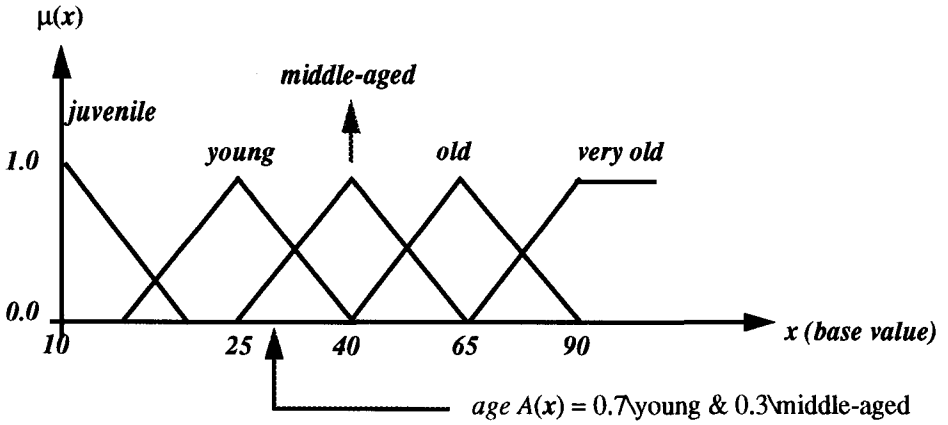


Figure 1. Membership functions of the various age groups

Consider that if x=30, the value of linguistic "age" is then $A(x)$ = 0.7\young & 0.3\middle-aged, which means that for x=30, the linguistic variable A has the value "young" with a membership of 0.75 and the value "middle-aged" with a membership of 0.25.

Now suppose the following rule was imposed: IF A is young THEN P is energetic, where P stands for a person's physical condition. In the fuzzy logic terminology, this rule can be expressed as: $\mu_{\text{energetic}}(P) = \mu_{\text{young}}(A)$.

A clearer picture is offered in Figure 2.

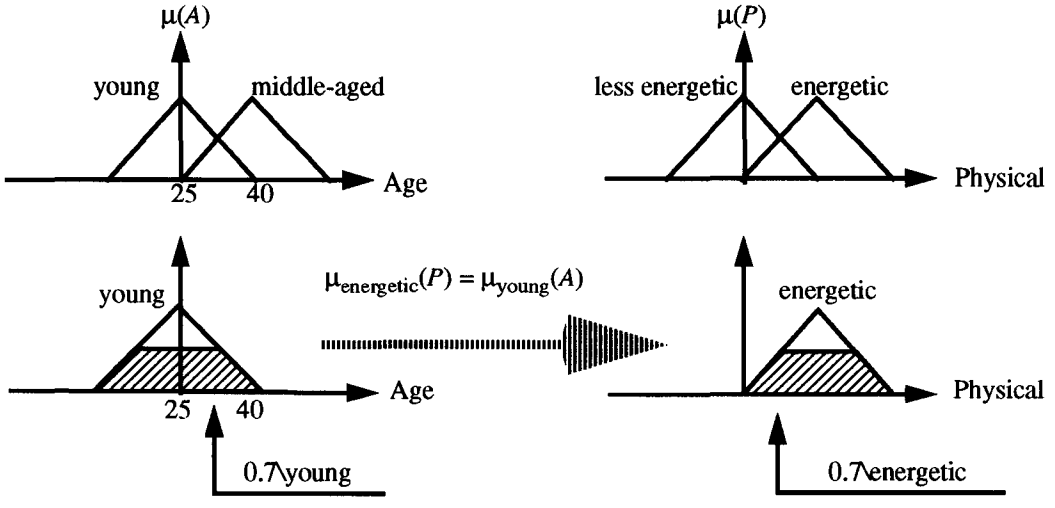


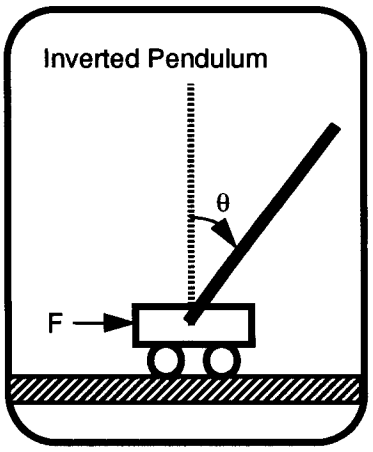
Figure 2. Example of fuzzy logic inference rule

This is actually a simple case of fuzzy inference with only one input (A) and one output (P). More general cases of fuzzy inference are discussed in the next section.

2.4 Fuzzy inference

The concept of fuzzy inference is illustrated with an "approximate reasoning" technique based on fuzzy logic rules, linguistic variables, and their membership functions. A typical example that model the balancing of an inverted pendulum is given below.

Figure 3 shows the formula of θ and ω as a function of time. According to fuzzy logic



$$\begin{cases} \omega(t) = \dot{\theta}(t) \\ \alpha(t) = \dot{\omega}(t) \end{cases}$$

$$M = \frac{m}{m_c + m}$$

$$\text{force} = \frac{F}{(m_c + m)\,L}$$

$$\begin{cases} \theta(t + \Delta t) = \Delta t\,\omega(t) + \theta(t) \\ \omega(t + \Delta t) = \Delta t\,\alpha(t) + \omega(t) \end{cases}$$

$$\alpha(t) = \frac{\frac{g}{L}\sin(\theta) - \cos(\theta)\,[\text{force} + M\,\omega^2 \sin(\theta)\,]}{\frac{4}{3} - M\cos^2(\theta)}$$

Figure 3. The motion of an inverted pendulum

methodology, these numerical variables θ, ω and the applied force F can be transformed as linguistic variables having several linguistic values (i.e. fuzzy sets) such as {negative, zero, positive}. This process is also called *fuzzification*. The fuzzified variables (θ, ω, F) can be represented through fuzzy sets and membership functions. For example, a θ of 30 degrees might be expressed as follows:

θ (30 degrees) = {(negative, 0), (zero, 0.2), (positive, 0.3)}

This is also depicted in Figure 4.

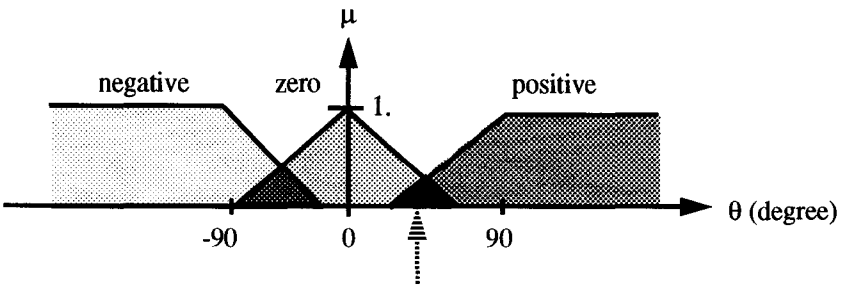


Figure 4. θ as a linguistic variable

Suppose the following fuzzy rules are concluded for balancing the inverted pendulum from human experience:

1. IF θ is negative AND ω is almost zero, THEN apply negative force F.
2. IF θ is negative AND ω is negative, THEN don't do anything (F = 0).
3. IF θ is positive AND ω is positive, THEN apply large positive force F.

And these rules can be written in some rule table as shown below:

Table 3. Fuzzy rule table

Force F	ω = negative	ω = zero	ω = positive
θ = negative	large negative	negative	zero
θ = zero	negative	zero	positive
θ = positive	zero	positive	large positive

Suppose there is a set of specific input values (θ, ω), which can be fuzzified into linguistic values through their membership setting:

θ = 0.8\zero & 0.25\negative;

ω = 0.45\zero & 0.5\positive;

By applying the fuzzy rules of table 3, the following information is obtained:

w_1(θ =zero, ω =zero) = min(0.8, 0.45) = 0.45;

w_2(θ =zero, ω =positive) = min(0.8, 0.5) = 0.5.

w_3(θ =negative, ω =zero) = min(0.25, 0.45) = 0.25.

w_4(θ =negative, ω =positive) = min(0.25, 0.5) = 0.25.

Thus our output grain size *s* is:

F = w_1\zero & w_2\negative & w_3\positive & w_4\zero .

There are many ways other than the above "minimum" operation to obtain the weights w_1 to w_4 from the "AND" rules for multiple inputs. An alternative is to use the product of membership values of each individual input (see Section 2.5).

Finally, a "defuzzification" technique should be applied to obtain a numerical value of an output linguistic variable after fuzzy inference. There are also many ways to defuzzify an output. A simple way is to do a "weighted average":

$$F = \frac{\sum_i w_i \cdot A_i}{\sum_i w_i} \tag{2-2}$$

where A_i are corresponding numerical values for F = negative, positive or zero. This algorithm is better illustrated in Figure 5 (w_1 and w_4 are omitted in the figure).

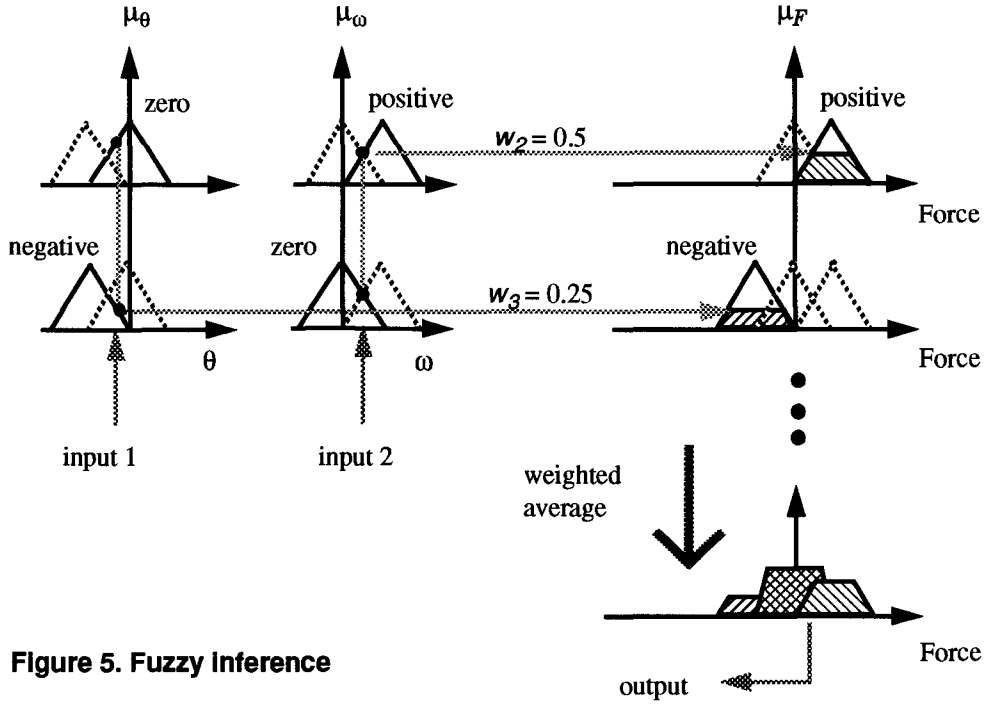


Figure 5. Fuzzy Inference

In a more general situation, as shown in Figure 5, different rule uses may lead to "conflicting" results, say, there may be two or more different weights for one specific output value. All that is required is to sum up the weights for the same output (linguistic) value and then apply the above defuzzifying algorithm to obtain the numerical output value.

2.5 A simple fuzzy system model

Specifically, a fuzzy inference system with two independent inputs $\{x_1, x_2\}$ and one output y is depicted in this section. However, it can be easily generalized to a multiple n-input system.

The inference rule of such a 2-input fuzzy system could be expressed as:

- Rule-{i,j}: IF x_1 is A_1[i] and x_2 is A_2[j] THEN y is w[i,j].

where {i,j} (i=0,1,...,m, j=0,1,...,n) are rule numbers, A_1[i] and A_2[j] are membership functions of the antecedent part, and w[i,j] is a real number of the consequent part of the output value of y.

The membership functions A_1[i] are expressed by triangles as shown:

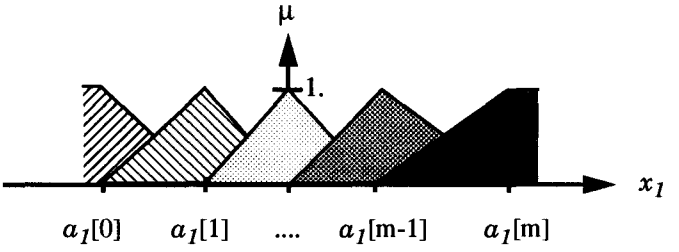


Figure 6. Membership function for x_1

And their mathematical expression is written as:

$$A_1[i] = \begin{cases} \dfrac{x_1 - a_1[i-1]}{a_1[i] - a_1[i-1]}, & \text{when } a_1[i-1] < x_1 \le a_1[i]; \\ \dfrac{a_1[i+1] - x_1}{a_1[i+1] - a_1[i]}, & \text{when } a_1[i] < x_1 \le a_1[i+1]; \\ 0, & \text{otherwise.} \end{cases} \tag{2-3}$$

Similarly the expression for A_2[j] is acquired simply by changing the subscript 1 to 2.

When applying the Rule-{i,j}, however, a different approach is employed to the "AND" rule from those in Table 2. Namely, the product of A_1[i] and A_2[j], instead of their minimum value, is used to obtain the weight for output value w[i,j]. The output y thus can be derived by weighted average:

$$y = \sum_{i,j} A_1[i] \cdot A_2[j] \cdot w[i,j] \tag{2-4}$$

For the membership functions specifically defined in Eq. (2-3), the sum of weights is a constant: $\sum_{i,j} A_1[i] \cdot A_2[j] = 1.$

3 Diagram Design for the Fuzzy Inference System

A thorough description of Tsutsuji diagram design for the one-dimensional and two-dimensional fuzzy inference system/module will be presented in this chapter.

Starting from basic library components, a hierarchical series of building blocks of deign modules are built on the "Workbook" of Tsutsuji.

3.1 Workbook System Design

The system design is carried out by connecting various components, both intrinsic library components and the user defined lower level systems, i.e. the user defined components.

3.1.1 Library Components

The library components in Tsutsuji are those basic logic components such as Adders, Input, Bus,... (see Figure 7). They are the basic building blocks for the user to build up higher level block designs.

The bus width for each library component can be arbitrarily chosen. Some basic logic components, including *AND* and *OR* gates, can take mutable number of inputs.

After a module (or a user component) is built based on library components and lower level user components, it can be represented as a "black box" with only relevant input/output contact points on a user-drawn symbol. This symbol represents the user component being used just the same as a library component for a higher level component design.

However, up to this point, only the input/output bus size of a user component can be changed like an library component. It is impossible to make the number of inputs or outputs of a user component tunable at the "black box" level. This is one of the limitations of the current version of Tsutsuji system. (See more comments on this matter in Section 7).

3.1.2 Hierarchical User Components

In the "Workbook" menu, a user can build his/her own module consisting of library components connected by wires. Then, by using a simpler symbol to represent this module, the symbol can be treated as a new "personal library" module for higher level design. It works like a subroutine in a big program. The main program consists of library functions and subroutines. The subroutines may contain other subroutines and some library functions.

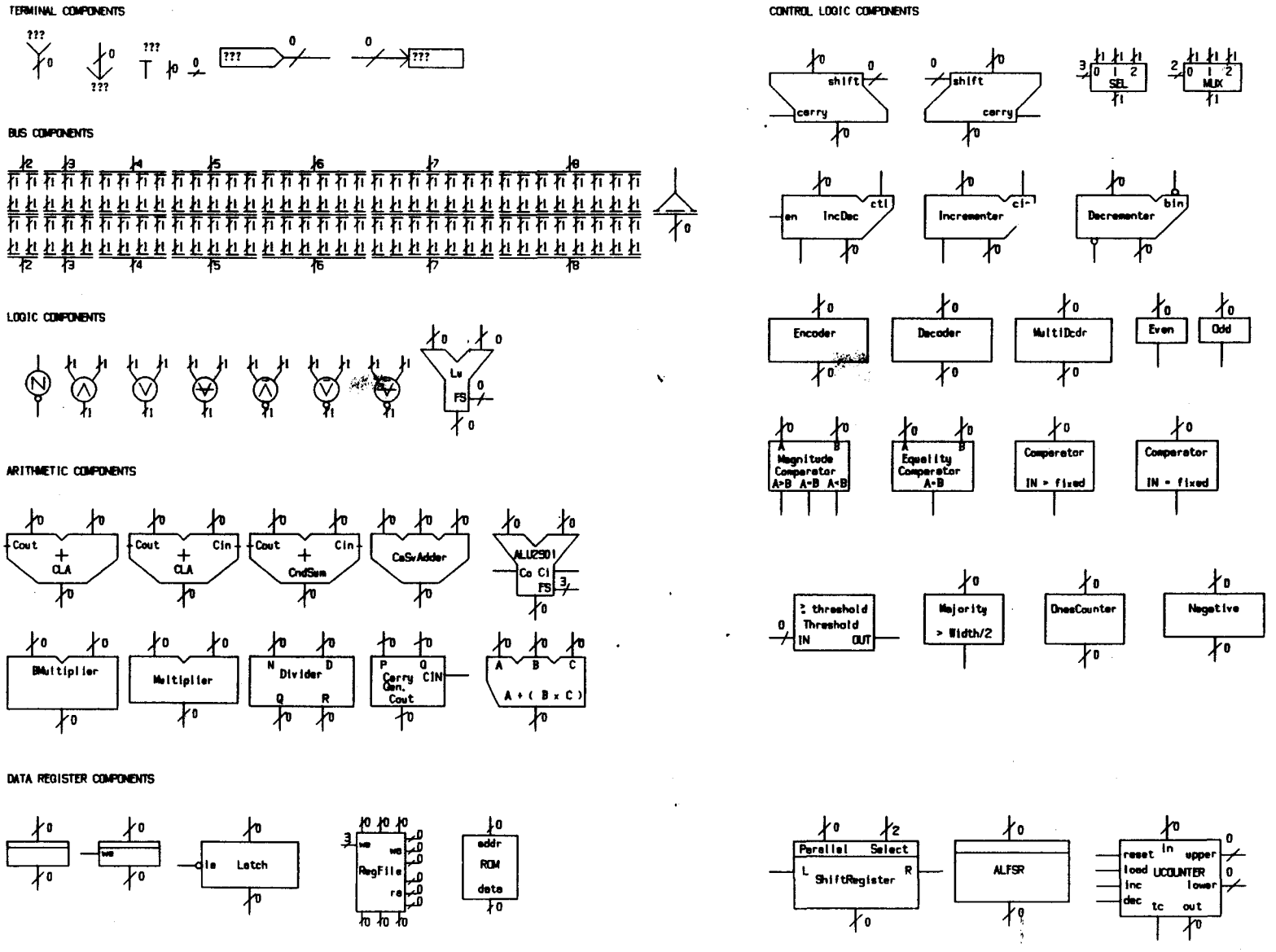


Figure 7. Library Components

3.2 Fuzzy Inference Control System.

This is a system having one input and one output: i.e. y=f(x). Here both x and y are numeric variables with their corresponding linguistic variables X and Y.

There are 4 designs [Chen93] whose names are listed as follows (the name "fish" comes from "Fuzzy Inference System at HP Lab"):

- **fish0** - an original version of a one-dimensional fuzzy inference system.
- **fish1** - same as fish0, but the number of membership functions can be tunable at the system ("black box") level. However, the tuning is only effective for the shape of design on the BlockDiagram page, as it can not be recognized by the simulation part of the present version of Tsutsuji. (See more comments on this in the Section 7).
- **fish2** - a better version of fish0, which can be used with the newly generated VIs (see Section 4.1 for VIs).
- **2d_fish** - a 2-dimensional fuzzy inference system.

Only fish2 and 2d_fish are described in this chapter.

3.2.1 One-Dimensional Fuzzy Inference System ("fish2")

The function of this system is to accomplish a simple single-input/single-output inference from the rules:

- Rule-{i}: IF x_1 is A_1[i] THEN y is w[i].

(Also see Figure 6 for its graphical interpretation).

To implement these fuzzy rules with the TSUTSJI system, several steps must be carried out. First of all, a membership function module needs to be constructed, so that for any input x, the module will output the corresponding membership values corresponding to x. This membership module is referred to as a fuzzifier (details are shown in Figures 8 and 9). Membership functions in this simple system are represented by triangular shape functions. After obtaining all the membership values for x, the system will do a weighted average, or defuzzification, to obtain the final numerical output y. The whole system takes in A_1[i] and w[i] as parameter inputs. Depending on the number of membership functions, or the number of linguistic values, the one-dimensional *FISH* system stacks identical lower-level modules (called *fish-bone* in "fish2") which correspond to a single membership function (linguistic value).

All the "fish2" components are listed in Figure 8. The details of their layout are shown in Figure 9.

As shown from the layout, a *zero* consists of one *constant* input and one *output*; both are library components. A *subtracter* consists of two *input*'s, one *output*, one *constant*, one *NOR* gate, and one *adder*. All of them are library components. Similarly, a *sum* consists of only the library components. Thus, *zero*, *subtracter*, and *sum* become the lowest level user components. However, *slopeUp* and *slopeDown* consist of not only

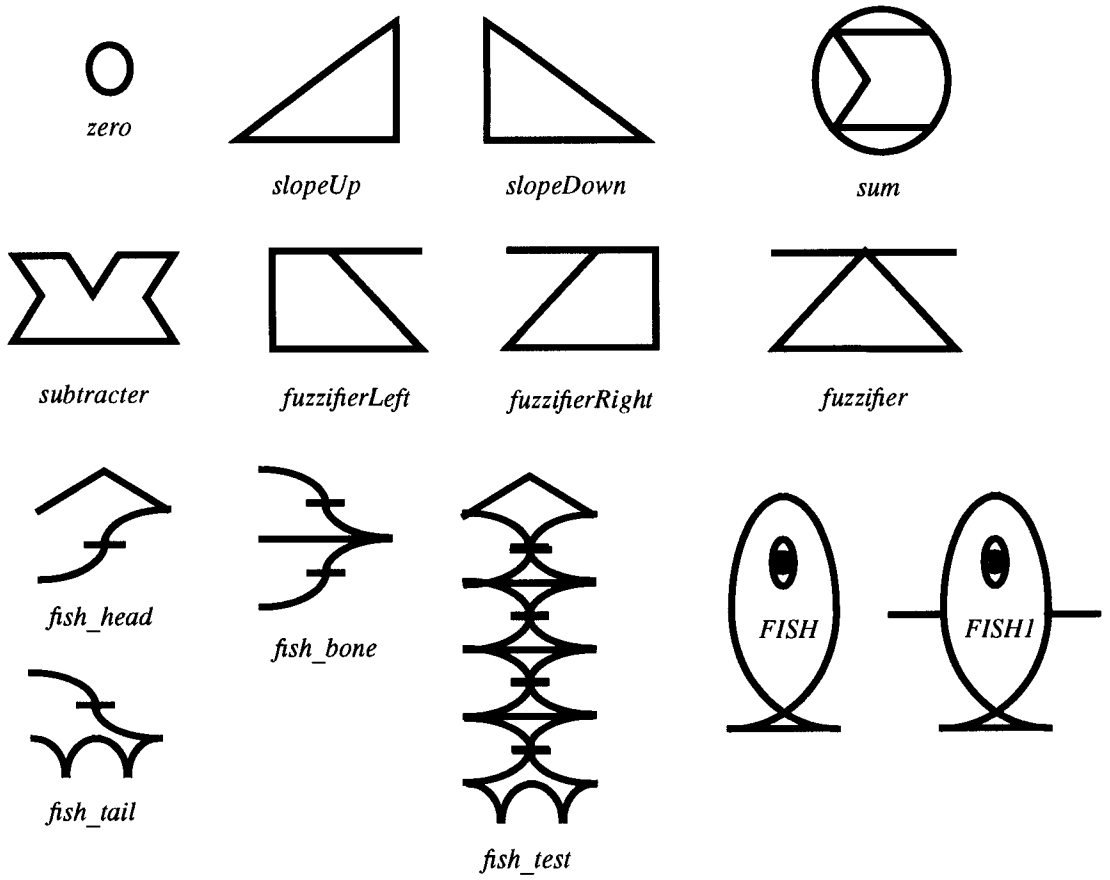


Figure 8. One-Dimensional Fuzzy Inference System (*FISH*)

the library components, but also the lower-level user components, i.e. one *zero* and two *subtracter*'s. The even higher level user components, *fuzzifierLeft*, *fuzzifier*, *fuzzifierRight*, consist of the library components, basic user components and the non-basic lower level user components. They become the lower level user components for higher level user components, i.e. *fish_head*, *fish_bone*, and *fish_tail*. The highest level user component is *fish_test*, which carries out the one dimensional inference rule, and which can be simulated in the Tsutsuji environment with VIs on an X-window interface. There are three *fish_bone* components in Figure 7, which means that the final system consists of five (=3 + 1 *fish_tail* + 1 *fish_head*) membership functions. However, a user can easily design a system for n-membership *FISH* system by stacking $n-2$ fish_bone components. There was an effort to make the *FISH* tunable to the number of membership functions (see design "fish1" [Chen93]), i.e. make *FISH* as a black-box and a user can tune the number of membership functions through a tuning page, as one might do with some library components, *OR*, *AND*, etc. This attempt failed, however, due to a limitation of the current version of the Tsutsuji system.

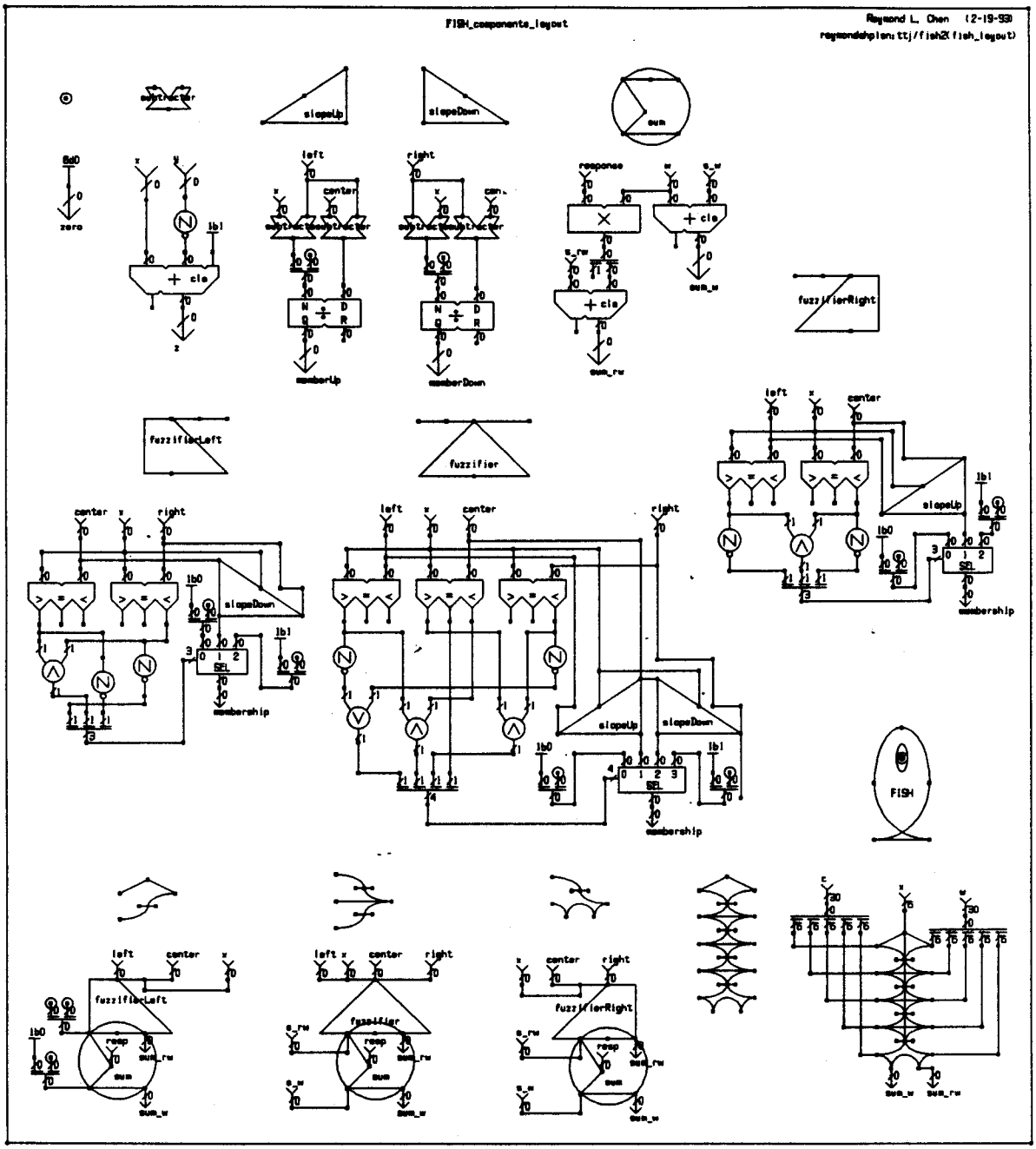


Figure 9. "fish2" Components Layout (1-D FISH)

The above components accomplish various functions as shown below:

- *zero* — input an n-bit zero.
- *subtracter* — $z=x-y$, where x, y are inputs, z is output.
- *slopeUp* — as their shapes show, $memberUp = x * (center - left)$.
- *slopeDown* — similarly, $memberDown = x * (right - center)$.
- *fuzzifierLeft*, *fuzzifier*, *fuzzifierRight* — output membership values of an input value x for three different types of membership shapes.
- *sum* — for carrying out the defuzzification procedure, i.e. weighted average.
- *fish_head*, *fish_bone*, *fish_tail* — when they are connected together, the *fish_head* corresponds to the left-most membership function (or the smallest linguistic value), the *fish_tail* corresponds the right-most membership function (or the largest linguistic value), while the *fish_bone* is the triangular membership function in between.

Thus, the final component *fish_test* consists of one *fish_head*, one *fish_tail*, and several *fish_bone*'s, depending on the number of linguistic values in a specific problem. The user need not invoke the detailed layout of the lower level components when applying them to a real problem, except to decide the number of *fish_bone*'s to use and the bus width of the input/output of *fish_test* (and *zero*).

The simulation is shown in Figure 10. In the figure, widget "c" and "w" are *consfive* VIs (virtual instruments) developed from Tsutsuji's original constant VI (named as *consgen*). Widget "x" is a *funcgen* VI, and widget "fish_test" is a *viewer* VI. When doing simulation, a user can dynamically choose a different type of input x from "x" widget, and tune the positions of membership and the values of weight parameters from widgets "c" and "w", respectively. In the particular case of Figure 7, x is a linear function of time, and thus *sum_rw* shows the corresponding output of *FISH* system. The other output *sum_w*, i.e. sum of weight, should be equal to one, but sometimes it becomes one bit smaller than one, due to the hardware truncation in the inference process.

All the numerical numbers are in fact represented in our system by decimal fraction numbers, i.e. from [−1, 1) for two's complement representation or [0, 2) for unsigned representation. Thus, the numerical numbers should be scaled to [−1, 1) or [0, 1). In Figure 7, for instance, the bus width is 6-bit, thus the number interval [0, 63] for VI which corresponds to [0.00000, 1.11111] in binary representation.

3.2.2 Two-Dimensional Fuzzy Inference System ("2d_fish")

Similarly to the one-dimensional case, a two-dimensional fuzzy inference system can be designed from hierarchical user components. Each single rule in a rule table cell (see Table 3) is represented by a module *cell*. The inference system: $y=f(x_1, x_2)$ can be realized by stacking a number of *cell* components shown as in Figure 11, which is

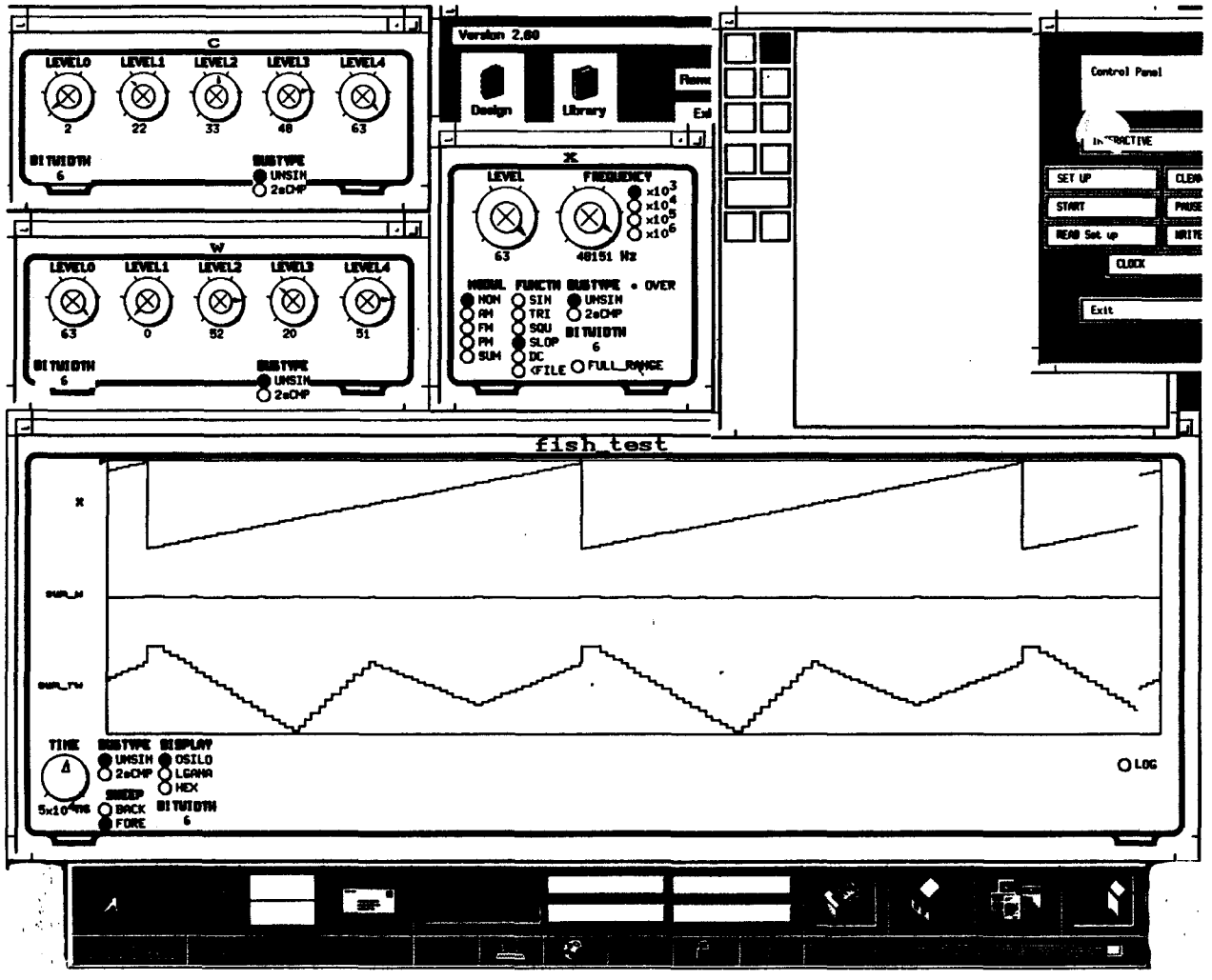


Figure 10. "fish2" Simulation Environment (1-D FISH)

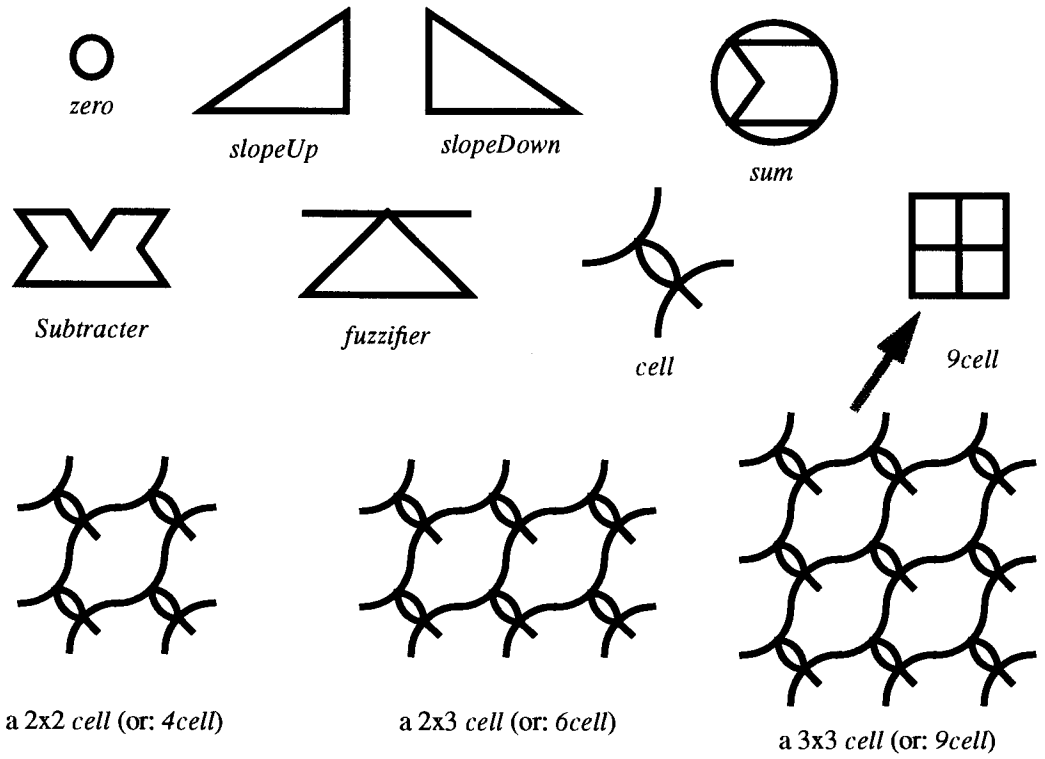


Figure 11. Two-Dimensional Fuzzy Inference System ("2d_fish")

extracted from the design named as "2d_fish". The detailed layout is shown in Figure 12.

In this case, the user components *zero*, *slopeUp*, *slopeDown* and *sum* are almost identical to those of one-dimensional case, which is further evidence of the advantages of Tsutsuji's design structure — one can always use the simpler modules to build more complex ones. *fuzzifier* is a combination of *fuzzifierLeft*, *fuzzifier*, and *fuzzifierRight* in the one-dimensional case, just for simplicity. A *cell* is build on the above simpler components. The output of *cell* corresponds to the membership function value for a pair of inputs (x_1, x_2). Thus, one may construct a two-dimensional inference system by stacking a matrix of *cell*'s, where the number and arrangement of those *cell*'s correspond to the number of linguistic values for the input linguistic variables (x_1, x_2). We will use *9cell* to construct out a prototype control system for balancing the inverted pendulum (Section 5).

3.3 Technology Mapping and Timing Verification

Among other logic synthesizing functions, Tsutsuji may generate a logical topology graph for a user design which shows the critical path, netlist, etc. This is not covered in this report, as it's already well developed within the Tsutsuji system [Culbertson93, YHP91].

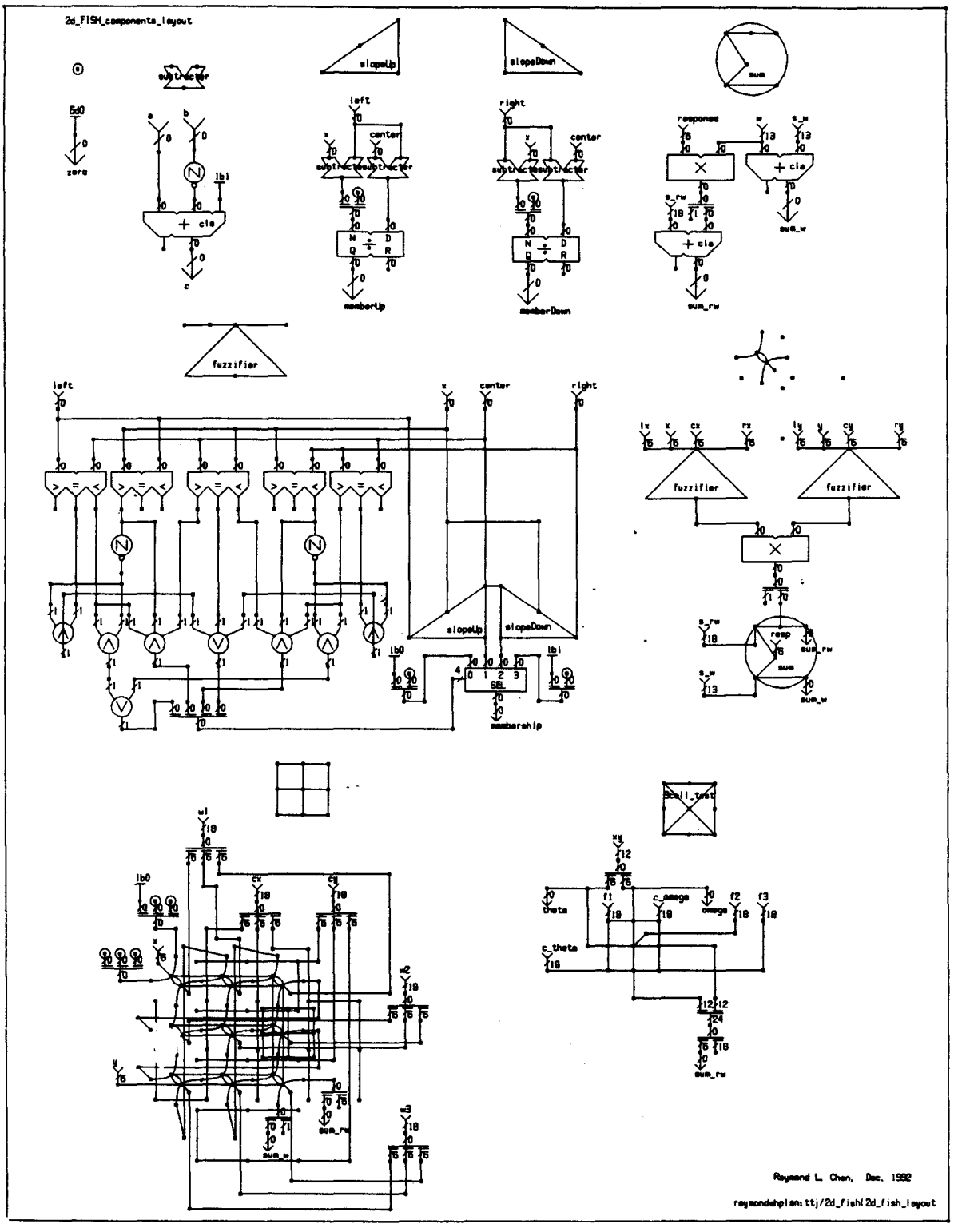


Figure 12. "2d_fish" Components Layout ("3x3=9cell")

4 Programming Structure with Tsutsuji

This chapter describes the graphical virtual instruments (VIs) specially created for the simulation of fuzzy inference modules. These VIs are created by modifying the basic Tsutsuji VIs.

4.1 Virtual Instruments (VIs)

Tsutsuji offers a nice simulation environment for the logic verification of a design. With X-window based widgets, i.e. Virtual Instruments, users can simulate the logic input-output relationship in a dynamical way. The basic programs for existing VIs was parallel but independently developed from Tsutsuji [Tanaka92], thus the whole programming structure for VIs are self-contained. The only channel for VIs and "sim" (executable file for Tsutsuji simulation programs) to communicate is through sockets in a UNIX system.

Among others, five *types* of VIs are commonly used in Tsutsuji simulation: *funcgen* (*V_INOUT*), *consgen* (*V_OUT*), *viewer* (*V_IN* or *V_MIN*) and *netana* (*V_OUTIN*) [Chen93]. However, since the fuzzy system has many special features required for dynamic simulation, the available VIs are not very suitable for the fuzzy system. For example, the original *consgen* outputs only one constant, but the fuzzy inference modules need many constant inputs as the parameters for defining the membership functions, hence some special constant generator VIs are created for this project. Several other specific VIs are made for the inverted pendulum controlling system (see Section 5), and they are named as *invpen*, *invpen1* and *invpen2* [Chen93]. The most recent version is *invpen2*.

4.2 Communication through Sockets

The whole simulation on Tsutsuji is done in a multi-process fashion. Each VI corresponds to a running process. The inter-communication between processes is carried out by sockets. To create the user defined widgets, not only the C++ programs based on Tsutsuji's existing VI code need to be re-written, but also the way sockets communicate needs to be re-defined.

The VI *invpen2* is used in this section as an example to illustrate the way to create a VI [Chen93]. Starting with the existing C++ programs for the original VIs, which are in this case *viewer* and *netana*, a new VI program for *invpen2* can be created. It uses many Tsutsuji library subroutines and subroutines for the various basic X-window widgets which are the building blocks for all VIs. Modifications are also needed for many source programs which serve as communication interfaces between VIs and Tsutsuji simulation programs, between VIs and X-window library/system programs, and among different VIs. For example, while the basic source programs of *invpen2* are contained in one sub-directory named as "*Invpen2/*", programs in other common library directories and basic widget directories are also modified to accommodate the characteristics of *invpen2*, such as its type (*V_OUTIN*) and its special needs for sockets to communicate with the rest of the Tsutsuji system. The type *V_OUTIN* implies that VI *invpen2* takes in signals from a design output and gives out signals to

the design input - all this can be done interactively in a simulation. In the case of balancing the inverted pendulum (Section 5), it takes in the output "Force" from the two-dimensional fuzzy inference component (i.e. *9cell*), and at the same time, feeds back the "theta" and "omega" values of the next time step back into the input of the *9cell*.

In summary, when creating a special purpose user-defined VI, the designer needs to re-write the subroutine "inout()" in a typical VI definition program, in addition to modifying all the rest of the program to define the number of input/output knobs, shape of the VI widget, initial values of each variable, etc. Some typical user-created VIs in this project are [Chen93]:

- *invpen*, *invpen1*, *invpen2* — Specific VI for balancing the inverted pendulum (see widget "Inverted_Pendulum" in Figure 15 in Section 5).
- *consfive* — Generating five constants horizontally in one VI. See Figure 10.
- *consthree* — Generating three constants horizontally in one VI. See Figure 15.
- *consthrey* — Generating three constants vertically in one VI. Also see Figure 15.

All the "knobs" on an VI can be adjusted dynamically in a simulation process. For instance, widget "Inverted_Pendulum" shown in Figure 15 is one example of VI *invpen2*, which has seven knobs to be dynamically tuned by the designer in a simulation. Their physical interpolation will be presented in Section 5.

5 Balancing the Inverted Pendulum

In addition to the theoretical model in Sections 2.4 and 2.5, the Tsutsuji design of a fuzzy inference control system and its simulation are discussed in this chapter. The goal of this system is to simulate the balancing control of an inverted pendulum. The inputs of the system are θ and ω (or "theta" and "omega" in the programs), the output is a force F which reduces the magnitude of both θ and ω. The control fails when θ falls out of the range $[-\pi/2, \pi/2]$.

The simulation flow chart is shown in Figure 13. In the simulation, however, the motion of a pendulum would follow the formulas shown in Figure 3. As we see in Figure 13, there are nine rules (cells) in the rule table, thus, module *9cell_test*, which is based on the *9cell* component (see Figure 12 and design "2d_fish" [Chen93]), is used as the design for this problem. The human-adjustable rule table will contain the parameter inputs for *9cell_test*. The pendulum is simulated with *invpen2*, which dynamically outputs $\theta(t + \Delta t)$ and $\omega(t + \Delta t)$ according to the values of $\theta(t)$, $\omega(t)$ and $F(t)$. *9cell_test* and *invpen2* therefore become a feed-back control system with a dynamically adjustable rule table (see Figure 14). The rule table in Figures 13 & 14 will be represented by four *consthree* VIs and one *consthrey* VI.

The simulation panel is shown in Figure 15. As shown in the figure, there are seven knobs on *invpen2* which allow users to dynamically adjust the initial values of θ and ω, as well as other parameters of the pendulum listed below (also refer to Figure 3):

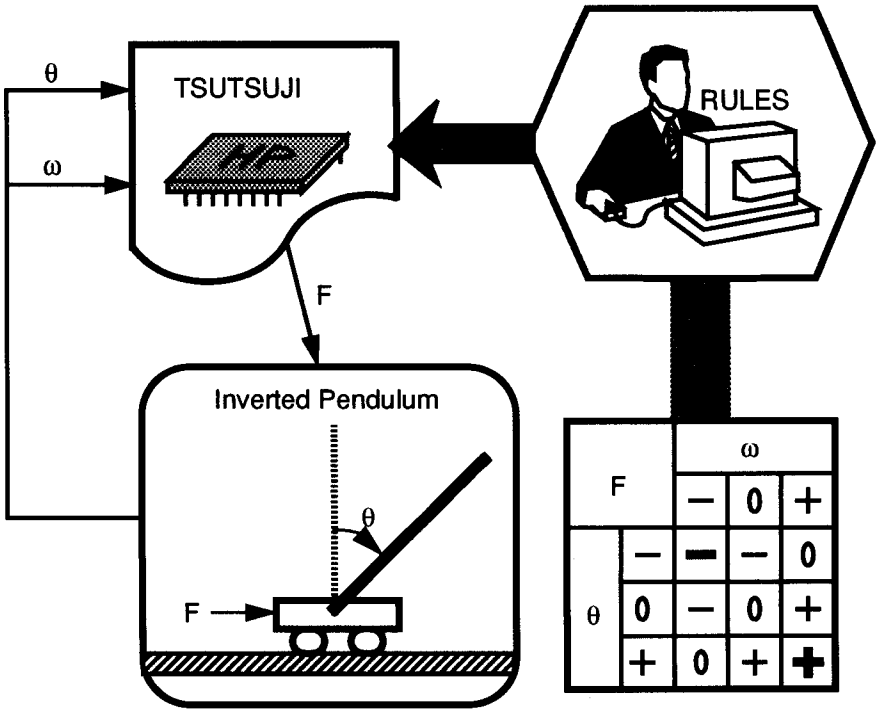


Figure 13. Balancing the Inverted Pendulum

- *LENGTH* — the scaled "length" of the pendulum, defined as g/L, where g is the gravity and L is the length of the pendulum.
- *MASS* — the mass ratio of pendulum and the total weight of pendulum and supporting cart, i.e. $M=m/(m_c + m)$.
- *DELTA* — time step Δt.
- *WAIT* — the real time lapse between two simulation steps (unit: Δt).
- *SCALE* — Scaling factor for "Force", which equals to $\frac{F}{(m_C + m)\,L}$ (see Figure 3).

Simulations with different set-ups show that the system is fairly stable and robust. This system shows that Tsutsuji could be a very effective design tool for fuzzy logic applications. Tsutsuji can also synthesize a gate-level netlist [Culbertson93, YHP91] that can be fit to different implementation technologies to produce an integrated circuit chip.

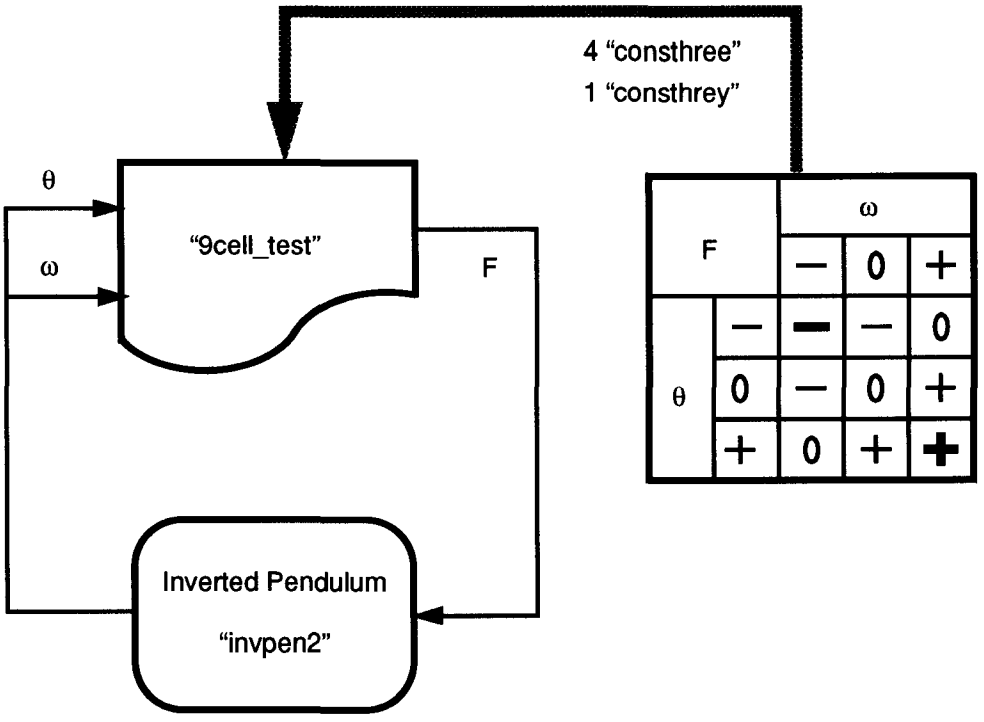


Figure 14. A Feedback Control System

6 Conclusions

A set of hierarchical fuzzy logic system components have been created and integrated into the Tsutsuji (the logic synthesis product from Yokagawa-Hewlett-Packard Design Systems Laboratory, or YSL) framework. These components are built on the existing Tsutsuji library components - adders, multipliers, AND, OR, etc. Tsutsuji synthesizes gate-level netlist, logical/physical topology report, mapping/timing report, etc. for producing the integrated circuits for a system based on different technologies.

Special graphical virtual instruments (VIs) have also been created that allow the designer to tune the design at a very high level. For example, the characteristics of fuzzy logic membership functions can be specified by adjusting knobs on several virtual instruments.

As a demonstration, a simple fuzzy logic control system for balancing an inverted pendulum is created and synthesized with the Tsutsuji simulation environment. It shows a good potential of the Tsutsuji system to be used for designing fuzzy logic related applications.

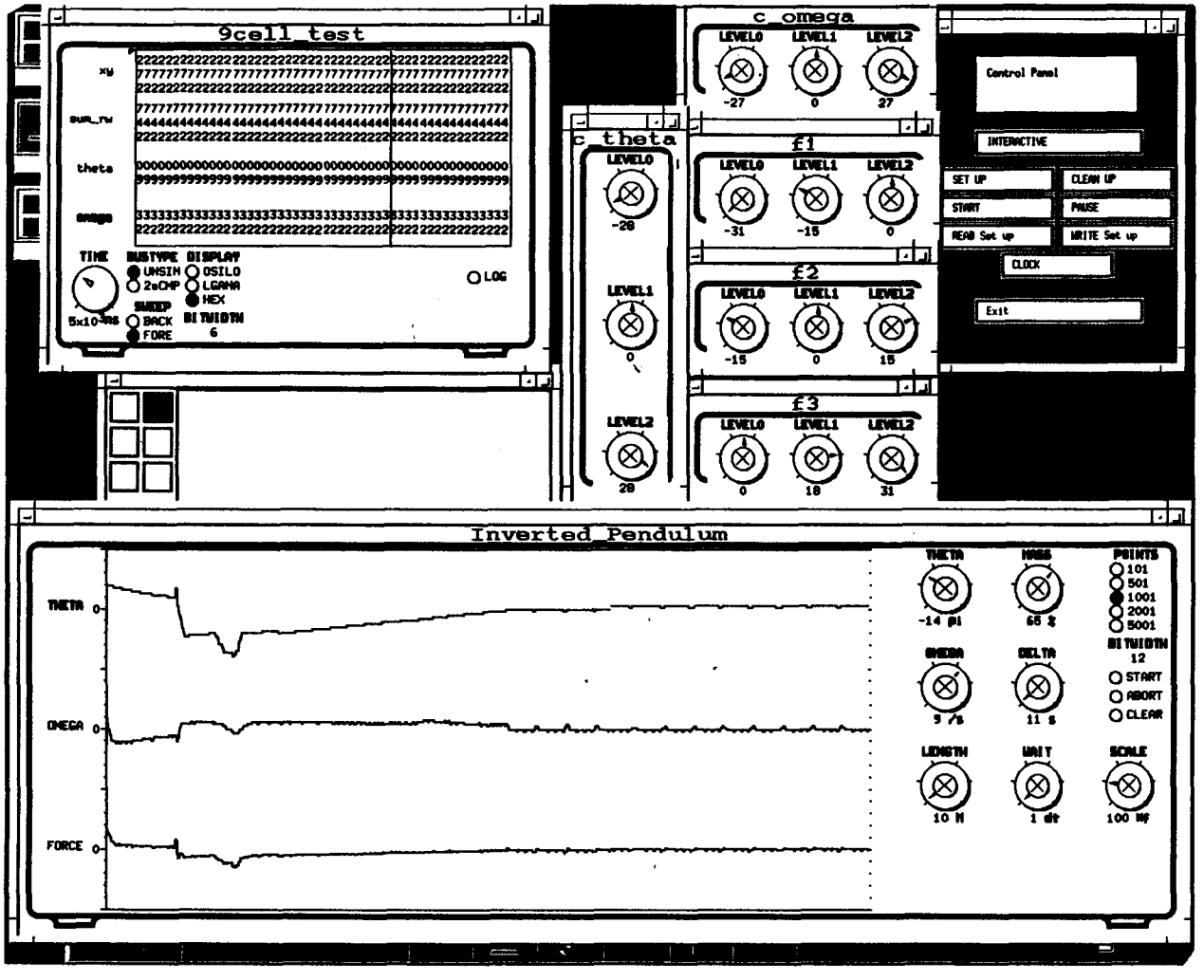


Figure 15. "2d_fish" Simulation Environment (2-D FISH)

7 Future Improvements

There are a number of improvements that can be done for Tsutsuji system, as well as the fuzzy inference system. The following is a list of miscellaneous suggestions for the Tsutsuji system:

- Add "undo" to the WorkBook ("Block-Diagram"). Even one step of undo would be very helpful for users in middle of a design process — drawing lines, moving objects and add/delete components.
- Add "scroll bar" to the WorkBook ("index" page). Right now users can use the four "arrow-keys" on the HP work-station keyboard, but scroll bars would be more convenient.
- Users should be able to *insert* a new BlockDiagram in between the "index" page of the WorkBook, instead of just adding the new one at the end. Users should also be able to re-order (sort) the index list easily. This would be convenient for a designer when there is a long list of lower level components of a design.
- There are some bugs in the "Block-Diagram" for "sim" (simulation). When an *input* component connects more than one wire(s) at the same point, the simulation gives error messages. Or when an input component connects more than one wire at different points, the simulation gives multiple *funcgen* inputs for that single input. Currently the problem can be solved by deleting the extra *funcgen* input.
- Users should be able to convert their "Block-Diagram" into a black-box module, which could be stored in their own "library" for other designs. Namely, the number of inputs/outputs could be tuned and the bus width could be tuned just like the library components/modules *AND*, *OR* etc. The tuning doesn't have to invoke the detail re-editing of individual lower level Block-Diagram (component) designs.
- Users should be able to have some "widgetName.h" in an extended "include" directories which contain the definitions of some basic VI (virtual instrument) widgets — such as: knobs, viewer windows, frequency toggles, constant generators, etc. Therefore users may put less effort into editing C/C++ programs to create their own application-specific virtual instrument widgets for the input/output interface of the "simulation" part of Tsutsuji.

The fuzzy inference system for balancing the inverted pendulum can also be improved by adding one more dimension of controlling input — the horizontal position of the supporting cart. To avoid having too many rules for such a three-input system (e.g. 3x3x3=27 rules), two separate inference systems, i.e. one two-dimensional inference system (for controlling θ and ω) and one one-dimensional inference system (for controlling position x), could be used. The final output could be a weighted average of the outputs from each inference system. Only 3x3+3=12 rules are needed this way.

8 Acknowledgment

I would take this opportunity to thank my supervisors on this project: Barry Shackleford and Bruce Culbertson. Without their attentive advice and help, this project would not have been possible. They also made my experience at HP Lab much more interesting and meaningful than I expected. I am in debt to Mr. Motoo Tanaka at YSL, who sent numerous e-mails to help me to understand the details of Tsutsuji system. I also thank other helpful people at HPL, who gave me strong support and made my work at HPL easier: Dr. Steven Rosenberg, Kathy Shaker, Wendy Fong, and Jackie Burleigh. I am most appreciative of Wendy Fong and Steven Rosenberg's generous proofreading of this report.

I am grateful to Professor Lotfi Zadeh at U.C. Berkeley for his recommendation of me to take this project. His continuous encouragement is always my inspiration. I am also delighted to have many people come to see the demonstration of this project on Dec. 23, 1993, including Professor Zadeh and Dr. Joel Birnbaum (HPL Director).

Finally, I appreciate my Ph.D research advisor at U.C. Berkeley, Professor Costas Spanos, for his understanding and support of my doing this project at HPL.

9 References

[Chen93] All the design programs are currently stored in author's account at an HPL workstation: see directory "*raymond@hplsn.hpl.hp.com:**ttj/***". The programs for specifically created fuzzy inference VIs are stored in the same account: see directory "*raymond@hplsn.hpl.hp.com:**VI_japan/***". The compiled VI executable files, however, are copied into the relevant Tsutsuji directory: "*hplsn.hpl.hp.com:/**usr/tsutsuji/lib/VI/***" and their corresponding definition files are in "*hplsn.hpl.hp.com:/**usr/tsutsuji/db/VI/***".

[Culbertson93] W. Bruce Culbertson, Toshiki Osame, Yoshisuke Otsuru, J. Barry Shackleford, and Motoo Tanaka, "The HP Tsutsuji Logic Synthesis System", *Hewlett-Packard Journal,* Volume **44**, Number 4, 11-24 (August, 1993).

[Tanaka92] Motoo Tanaka, Manager for Tsutsuji system, Yokagawa-Hewlett-Packard Design Systems Laboratory, *private communication* through e-mail. (See files at "*raymond@hplsn.hpl.hp.com:**mails_japan/***"). Mr. Tanaka also provided copies of all the source programs for Tsutsuji VIs, from which the special fuzzy inference VIs were created.

[YHP91] YHP Design System Laboratory, "ASIC Design System", i.e. *Tsutsuji manuals* (1991).

[Zadeh65] Lotfi A. Zadeh, "Fuzzy Sets", *Information and Control* **8**, 338-353 (1965).

[Zimmermann91] H. -J. Zimmermann, "Fuzzy Set Theory - and its Applications", *Kluwer Academic Publishers* (1991).