1. Introduction

This chapter describes the automatic generation method of VHDL which lays out the control logic of a control system. This framework releases a designer from the work of describing the VHDL directly. Instead, the designer inputs the equation of motion of a system and target operation.

In this chapter, first, FPGA, CPLD, VHDL and evolutionary computation are outlined. This is basic knowledge required for an understanding of this chapter. Next, the framework of automatic generation of FPGA using evolutionary computation is described. VHDL description is expressed by several kinds of data structures called a chromosome. VHDL expressed in a chromosome is changed using evolutionary computation and changes to a more suitable code for controller purposes. Finally two example applications are shown. The first one is the controller for a simple inverted pendulum. After that, the framework is applied to a more complicated system, an air-conditioning system. The simulation results show that the controller automatically generated using this framework can control the system appropriately.

2. Computer-aided controller design using evolutionary computation

2.1 FPGA/CPLD/ASIC and VHDL

CPLDs and FPGAs are both sorts of programmable LSIs. The internal logic of both can be designed using HDL. The ASIC is one example of a device that can be designed using HDL in the same way as programmable LSIs. CPLDs and FPGAs can be immediately evaluated on the system for the designed logical circuit. In addition, they are flexible for the rearrangement of a specification. These merits make them suitable for the intended use in the case of a rapid prototyping. For this reason, a CPLD is used as a controller. However, the proposed framework is applicable to all devices that can be designed by HDL. VHDL is one of the most popular HDLs and is therefore used in this paper.

The logic described by VHDL is verified and synthesized using a simulator or a logic synthesis tool so that it can be written into a device. When CPLD or FPGA serves as target devices, the programming code which determines the function of the target device can be, through a download cable, written into it in order to obtain the target LSI easily. The VHDL for a simple logical circuit is shown in Fig. 1.
library IEEE;
use IEEE.std_logic_1164.all;

entity HALF_ADDER is
port(
    A,B : in std_logic;
    S,CO : out std_logic);
end HALF_ADDER;

architecture DATAFLOW of HALF_ADDER is
signal C, D : std_logic;
beg
    C  <= A or B;
    D  <= A nand B;
    CO <= not D;
    S  <= C and D;
end DATAFLOW;

2.2 Genetic algorithm

The genetic algorithm used as a basis of this framework is outlined below (Fig. 2). The decision-variable vector \( x \) of an optimization problem is expressed with the sequence of \( N \) notations \( s_j (j = 1, \cdots, N) \) as follows:

\[
x : s = s_1 s_2 s_3 \cdots s_N
\]  

It is assumed that symbol string is a chromosome consisting of \( N \) loci. \( s_j \) is a gene in the \( j \)-th locus and value \( s_j \) is an allelomorph. The value is assumed to be a real number, a mere notation, and so on of the group of a certain integer or a certain range of observations as an allelomorph. The population consists of \( K \) individuals expressed with Eq. (1). Individual population \( p(n) \) in generation \( n \) changes to individual population \( p(n + 1) \) in next generation \( n + 1 \) through the reproduction of a gene. If reproduction in a generation is repeated and if the individual who expresses solution \( x \) nearer to an optimum value is chosen with high probability, then the value increases and an optimum solution is obtained (Goldberg, 1989; Koza, 1994).

2.3 Evolvable hardware

Higuchi et al. proposed evolvable hardware that regards the architecture bit of CPLD as a chromosome of a genetic algorithm to find a better hardware structure by genetic algorithm (Higuchi et al., 1992). They applied this to myoelectric controllers for electrical prosthetic hands, image data compression and so on (Kajitani et al., 2005; Sakanasi et al., 2004). This approach features coding of an architecture bit directly for a chromosome. The designer must determine which CPLD is used beforehand, implicitly determining the meaning of the architecture bit. If hardware is changed for the problem of circuit scale or structure, it is necessary to recalculate evolutionary computation.

Henmi et al. evolved a hardware description language (HDL) called structured function description language (SFL) and applied it to robot walking acquisition (Henmi et al., 1997).
The basic motion, called an “action primitive”, must be designed in a binary string. In our approach, the designer needs only to define I/O pins for CPLD. VHDL rather than an architecture bit is coded directly onto the chromosome, so the chromosome structure does not depend on CPLD scale or type, and after VHDL is generated automatically, CPLD is selected appropriately to the VHDL scale. Pin assignment is set after VHDL generation.

2.4 Controller design framework using evolutionary computation

The study of optimizing rewritable logical-circuit ICs, such as CPLD, using a genetic algorithm has increased. The framework that changes an internal logic circuit IC configuration to attain evolution is called evolvable hardware (EHW). With this framework, the designer needs only to define the criteria used to evaluate a controller. We explain the controller design framework using evolutionary computation with XC95144 (Xilinx, 1998) as the test device. Internal blocks of XC95144 are shown in Fig. 3. XC95144 is a small CPLD that has 144 pins (117 user input-outputs), 144 macro cells and 3200 usable gates. A designer chooses input and output signals from 117 user I/Os and assigns these pins. Each signal is defined for each I/O. If CPLD is used in control, sensors and actuators can be associated with CPLD I/O pins (Fig. 4). Here, I/O pins are associated with two sensors and one actuator. The sensor values are inputs to CPLD as two 16-bit digital signals and a 10-bit digital signal is output as a reference signal to the actuator.

VHDL, which describes internal CPLD logic, is encoded on a chromosome. An example of the VHDL generated is shown in Fig. 5, corresponding to Fig. 4. This VHDL consists of three declarations — (a) entity declaration, (b) signal declaration, and (c) architecture body. CPLD I/O signals are defined in (a) and internal CPLD signals are in (b). Signals mainly used in VHDL are a std_logic type and a std_logic_vector type. The std_logic type is used when dealing with a signal alone and the std_logic_vector type is used when dealing with signals collectively. The std_logic and the std_logic_vector types should be combined to optimize maintenance and readability. A description of VHDL can be restored if all of the input output signals and internal signals are used as the same std_logic type and only the number is encoded on a chromosome. The number of input signals, output
signals and internal signals are encoded on the head of a chromosome (Fig. 6). In Fig. 4, two input signals are set up with 16 bits and an output signal with 10 bits.

A chromosome that represents a VHDL assignment statement is shown in Fig. 7. A chromosome structure corresponding to a process statement is shown in Fig. 8. The value currently described is equivalent to the process statement in which "DI009" and "DI014" are enumerated in the sensitivity list.
VHDL Design Automation Using Evolutionary Computation

The VHDL description has an if-statement inside of a process statement and the description has two nesting levels. The hierarchy of the list structure is deep compared to the substitution statement. When the gene of such a multilist structure is prepared, it is possible to represent various VHDL expressions.

2.5 Variable length chromosome and genetic operations

The structure of a chromosome changes with the control design specification. The number of internal signals is set arbitrarily and different descriptions in VHDL are expressed with different locus lengths. The length of a chromosome is determined by the VHDL line count. The length determines the number of internal signals enumerated on the sensitivity list or
the length of the right-hand side of an assignment statement. When dealing with such a variable length chromosome, the problem is that genetic operations will generate conflict on a chromosome. To avoid this problem, we prepared the following restrictions:

1. With a top layer, the length of a chromosome is equal to the number that added one to the summary of the number of internal signals and the number of output signals.

2. All signals are encoded on a chromosome using a reference number.
3. The signal with a large reference number is described by only the signal whose reference number is smaller than the signal.

4. The top layer of a chromosome describes the entity declaration using all internal signals and output signals in order with a low reference number. Each signal can be used only once.

5. Crossover operates on the top layer of a chromosome.

These restrictions avoid the conflict caused by genetic operations.

Figure 9 shows an example of crossover operation. The back of the 6th gene is chosen in this example. Chromosome (A) and chromosome (B) cross and change to chromosome (A’) and chromosome (B’). Only the gene before and behind the crossover point of each chromosome
show the gene of a lower layer. In the figure, chromosome (A) has two sensitivity lists and chromosome (B) has two assignment statements. The structure of a chromosome changes by replacing the gene from the back of a top gene to before a crossover point. Both chromosome (A') and chromosome (B) came to have an assignment statement and a sensitivity list.

3. Application to HDL-based controller of inverted pendulum

In this section an application to an HDL-based controller for the inverted pendulum is described (Kojima, 2011).
3.1 Equations of motion

Figure 10 shows the model of the inverted pendulum. The equations of motion are given by

\[(m\ell^2 + f)\ddot{\theta} = ml\sin\theta - ml\dot{\theta} \cos\theta - c_1 \dot{\theta} + d \]  
\[(M + m)\ddot{z} = f - c_2 \dot{z} \]
\[ f = Kv \]  
where \( \theta \) [rad] is the angle of the pendulum, \( m \) [kg] is the mass of the pendulum, \( M \) [kg] is the mass of the cart, \( J \) [kgm^2] is the inertia of the rod, \( l \) [m] is the length between the gravity point and fulcrum, \( z \) [m] is the position of the cart, \( d \) [Nm] is the disturbance torque, \( c_1 \) [Ns/rad] is the viscous resistance of the pendulum, \( c_2 \) [Ns/m] is the viscous resistance of the cart, \( f \) [N] is the controlling force, \( v \) is the parameter for the control input and \( K \) is the gain. \( \bar{J}, a \) and \( b \) are defined as:

\[ \bar{J} = J + ml^2 \]  
\[ a = \frac{c_2}{M + m} \]  
\[ b = \frac{K}{M + m} \]
then, equations (2) and (3) become:

\[ \ddot{\theta} = \frac{mgl}{\bar{J}} \sin \theta - \frac{ml\dot{z}}{\bar{J}} \cos \theta - \frac{c_1 \dot{\theta}}{\bar{J}} + \frac{d}{\bar{J}} \]

(8)

\[ \ddot{z} = -a \dot{z} + bv. \]

(9)

Each parameter can be determined by measurement and experiment as

\[ l = 0.15, \quad m = 46.53 \times 10^{-3}, \quad \bar{J} = 1.58 \times 10^{-3}, \quad c_1 = 2.05 \times 10^{-2}, \quad a = 4.44 \quad \text{and} \quad b = 2.46 \times 10^{-1}. \]

Using these parameters, evolutionary simulations are conducted.

### 3.2 Control system and CPLD pin assignment

Figure 11 shows the control system block diagram. The CPLD is used as the controller of the inverted pendulum. The position of the cart and the angle of the pendulum are converted 16 bit digital signals respectively and input to the 32 CPLD pins. The control logic in the CPLD, which is formed using the framework previously mentioned, determines the 10 bit control signal driving the motor of the inverted pendulum. Figure 12 shows the CPLD pin assignment for this application. In this case, 32 bit parallel inputs and 10 bit parallel outputs are adopted. Instead of using them, we can use serial I/Os connected with A/D converters and D/A converters. In this case, serial to parallel logics should be formed in the CPLD and even though serial inputs are used, automatically generated VHDL can be used as a VHDL component.

### 3.3 Fitness

The fitness value is calculated as a penalty to the differences in the rod angle and the cart position.

\[ \text{fitness} = -\frac{1}{\pi} \int_0^{30} |\theta| \, dt - \int_0^{30} |z| \, dt \]

(10)

Disturbances are given as a random torque during the control simulation. When calculating fitness value, disturbance torque is always initialized. Therefore, all individuals are given different disturbance at each evaluation. This kind of disturbance makes the controller robust to various disturbances.

### 3.4 Simulation

Simulations are conducted under two conditions – (1) \( \theta_0 = 1^o, z_0 = 0 \), (2) \( \theta_0 = 180^o, z_0 = 0 \). Population size is 50, mutation rate is 0.5, crossover rate is 1.0, tournament strategy, tournament size is 10 and the elite strategy is adopted.

Figures 13 and 14 show the simulation results. Figure 13 shows the result of condition (1), Figure 14 shows the result of condition (2). (a) the result at 0 generation and (b) the result at 1000 generation are represented respectively. The angle of the rod, the position of the cart, disturbance and the control signal are shown at each generation. At zero generation, in both conditions (1) and (2), the obtained controller cannot control the inverted pendulum adequately. The rod moves in a vibrating manner at around 180°. At 1000 generation, the controller controls the inverted pendulum successfully in both conditions. Further, in condition (2), swing up motion can be observed (Fig.14(b)). The control signal at 1000 generation has more various patterns than the signal at 0 generation.
Fig. 13. Simulation results (θ₀ = 1°, z₀ = 0)
Fig. 14. Simulation results ($\theta_0 = 180^\circ$, $z_0 = 0$)
4. Application to an air-conditioning controller

Next, the framework is applied to an air-conditioning controller. Simulation model, task
definition and fitness function are described in this section. Simulation results will be shown
at the end of the section (Kojima et al., 2007; Kojima, 2009).

4.1 Simulation model

In the targeted air-conditioning system (Fig. 15), the outside air imported is cooled or warmed
and sent to the console to adjust temperature. Air entering from the inlet (a) is cooled by the
refrigerator (c). Part of the cooled air is warmed with a heater (e), then mixed with cool air to
adjust the temperature. The angle of the mixture door (d) controls the mixing ratio of warm
and cold air. Mixed air is sent to console (f), changing the indoor temperature. The system
controller controls blower motor rotation and the angle of the air mixture door.

![Air-conditioning system diagram](image)

**Fig. 15.** Air-conditioning system

To evaluate the controller performance, an air-conditioning simulation model is required in
evolutionary computation. We consider an air-conditioning model combining the console and
duct using nine control volumes (Fig. 16). Temperature and humidity, but not compressibility,
are considered in this model. To calculate the predicted mean vote (PMV), which indicates
thermal comfort, we require globe temperature, metabolism, flow rate, insulation of
clothes and external work.

**Fig. 16.** Control volumes

To evaluate the controller performance, an air-conditioning simulation model is required in
evolutionary computation. We consider an air-conditioning model combining the console and
duct using nine control volumes (Fig. 16). Temperature and humidity, but not compressibility,
are considered in this model. To calculate the predicted mean vote (PMV), which indicates
thermal comfort, we require globe temperature, metabolism, flow rate, insulation of
clothes and external work.

Figure 16 shows three control volumes in this system. We assume that air flows toward
the control volume indexed with \((i + 1)\) from the control volume indexed with \((i)\). Air
temperature \(T_{i+1}[K]\) and relative humidity \(\phi_{i+1}\) vary with temperature \(T_i[K]\), relative
humidity $\varphi_i$, mass flow $G_i[\text{kg/s}]$ and heat transfer $Q[\text{kJ/s}]$. In air-conditioning simulation, mass flow is proportional to the opening of blower motor $\alpha$.

$$G_0 = G_i = \alpha G_{\text{max}} \quad (0 \leq \alpha \leq 1) \quad (11)$$

where $G_{\text{max}}$ is maximum flow at full blower opening. Flow rate $G_0$ is the sum of air flow $G_{a0}$ and steam flow $G_{w0}$.

$$G_0 = G_{a0} + G_{w0} \quad (12)$$

Water vapour pressure $P_{w0}$ is as follows:

$$P_{w0} = \varphi_0 \cdot P_{s0} \quad (13)$$

where $P_{s0}$ saturation water vapour pressure from Tetens’ formula (Tetens, 1930):

$$P_{s0} = 610.78 \times e^{17.2694(T_0 - 273.15)/234.4} \quad (14)$$

Specific humidity $x_0$ is given by water vapour pressure $P_{w0}$.

$$x_0 = 0.622 \frac{P_{w0}}{P_0 - P_{w0}} \quad (15)$$

Air mass flow $G_{a0}$ is calculated from the following gas equation:

$$G_{a0} = P_{a0} \cdot \frac{V}{R_a T_0} \quad (16)$$

where $R_a$ is a gas constant. Water vapour mass flow $G_{w0}$ is given by:

$$G_{w0} = x_0 \cdot G_{a0} \quad (17)$$

Air flow rate $G_a$ is constant. Steam flow $G_w$ is constant except during dehumidification. Considering the air flow into control volume $(i)$ from control volume $(i-1)$ in unit time $dt[\text{s}]$, temperature $T_i'$, humidity $x_i'$ and mass of control volume $M_i'$ after $dt$ is given as follows:

$$T_i' = \frac{G_{i-1}C_{i-1}T_{i-1}dt + (M_i - G_i dt)C_i T_i}{G_{i-1}C_{i-1}dt + (M_i - G_i dt)C_i} + \frac{Q_i dt}{M_i C_i} \quad (18)$$

$$x_i' = \frac{M_i x_i + (1 + x_i)(G_{wi} - G_{w0}) dt}{M_i + (1 + x_i)(G_{ai} - G_{a0}) dt} \quad (19)$$

$$M_i' = M_i + (G_{i-1} - G_i) dt \quad (20)$$

where specific heat $C_i[\text{kJ/kg} \cdot \text{K}]$ is given by

$$C_i = \frac{1.005 + x_i \{ (2501.6/T_i) + 1.859 \}}{1 + x_i} \quad (21)$$

At the mixture door, air is divided into two flows. The ratio of the mass of divided air depends on the mixture door angle. Here $G_1$ is the mass flow at location (1) in Fig. 15, $G_2$ that at location (2) and $G_3$ that at location (3),

$$G_2 = \beta G_1 \quad (22)$$
\[ G_3 = (1 - \beta)G_1 \]  \hspace{1cm} (23)

where \( \beta \) (0 ≤ \( \beta \) ≤ 1) is the opening ratio of the mixture door. Adding two mass flow rates enables us to calculate the downstream mass at a juncture. Here \( G_4 \) is the mass flow at location (4) in Fig.15, \( G_5 \) that at location (5) and \( G_6 \) that at location (6),

\[ G_6 = G_4 + G_5 \]  \hspace{1cm} (24)

Temperature and humidity at a juncture are given in the same way as for when three control volumes are considered.

4.2 Predicted mean vote (PMV)

PMV is the predicted mean vote of a large population of people exposed to a certain environment. PMV represents the thermal comfort condition on a scale from -3 to 3, derived from the physics of heat transfer combined with an empirical fit to sensation. Thermal sensation is matched as follows: "+3" is "hot." "+2" is "warm." "+1" is slightly warm." "0" is "neutral." "-1" is slightly cool." "-2" is "cool." "-3" is "cold." Fanger derived his comfort equation from an extensive survey of the literature on experiments on thermal comfort (Fanger, 1970). This equation contains terms that relate to clothing insulation \( I_{cl} \)[clo], metabolic heat production \( M \)[W/m²], external work \( W \)[W/m²], air temperature \( T_a \)[°C], mean radiant temperature \( T_r \)[°C], relative air speed \( v \)[m/s] and vapour pressure of water vapour \( P \)[hPa].

\[
\text{PMV} = \left\{ 0.33 \exp(-0.036M) + 0.028 \right\} \left( M - W \right) \\
-3.05\left\{ 5.73 - 0.007(M - W) - P \right\} \\
-0.42\left\{ (M - W) - 58.1 \right\} \\
-0.0173M(5.87 - P) \\
-3.96 \times 10^{-8} f_{cl} \\{ (T_{cl} + 273.15)^4 \\
- (T_{mrt} + 273.15)^4 \} \\
-f_{cl} h_c(T_{cl} - T_a) \right\}
\]  \hspace{1cm} (25)

\( f_{cl} \) is the ratio of clothed and nude surface areas given by:

\[
f_{cl} = \begin{cases} 
1.0 + 0.2I_{cl} & (I_{cl} \leq 0.5) \\
1.05 + 0.1I_{cl} & (I_{cl} > 0.5)
\end{cases}
\]  \hspace{1cm} (26)

where \( T_{cl} \) is the clothing surface temperature given by repeated calculation of:

\[
T_{cl} = 35.7 - 0.028(M - W) \\
-0.155I_{cl} \left\{ 3.96 \times 10^{-8} f_{cl} \\{ (T_{cl} + 273.15)^4 \\
-(T_{mrt} + 273.15)^4 \} + f_{cl} h_c(T_{cl} - T_a) \right\}
\]  \hspace{1cm} (27)
where $h_c$ is the heat transfer coefficient,

$$h_c = \max\{2.38(T_{cl} - T_a)^{0.25}, 0.0121\sqrt{v}\}$$

and $T_{mrt}$ is mean radiant temperature. PMV is detailed in (Fanger, 1970).

### 4.3 Task definition

The task is to adjust PMV in the console despite heat transfer from outside changes. The air-conditioning controller controls the opening ratio of blower $n$ and the opening ration of mixture door $m$ appropriately.

### 4.4 Control system and CPLD pin assignment

Figure 17 shows the control system block diagram. PMV is input to the CPLD as a 8 bit signal. The control logic in the CPLD determines the 8 bit blower opening control signal and the 8 bit mixture door control signal. Figure 18 shows the CPLD pin assignment for this application.

![Control system block diagram](image)

**Fig. 17. Control system block diagram**

![CPLD pin assignment](image)

**Fig. 18. Control system block diagram**
4.5 Fitness

The fitness function is as follows:

$$\text{fitness} = -\int_{0}^{t_{\text{end}}} |\text{PMV}_{\text{ctrl}} - \text{PMV}_{\text{target}}| dt$$

(29)

where $t_{\text{end}}$ is the end of simulation time. The difference between target and controlled PMV is integrated as a penalty in the controller simulation.

4.6 Simulation results

Figure 19 shows the simulation results. PMV in a console is fed back to the controller. In the graphs, trends change every 60 seconds. Variations are based on the load effect change each 60 seconds. At zero generation (Fig. 19 (a)), temperature rises or falls with the change in heat load. PMV also changes simultaneously. This means that simply optimizing a controller is not enough. After 100 generations of calculation, the difference between the target and the
estimated value decreases (Fig. 19 (b)). In the 10,000th generation (Fig. 19 (c)), tolerance decreases further. These results show that hardware corresponding to the purpose is obtained automatically using this framework.

Figure 20 shows simulation results under other conditions. Nine graphs result for three thermal loads. Three graphs — temperature, PMV and control — are shown for each thermal condition. The controller used under three conditions was obtained after 10,000 generations of calculation. Although thermal load and load change timing were random, the blower and mix door were controlled so that PMV is set to zero.

Fig. 20. Simulation results

5. Conclusion

In this chapter, to automate controller design, CPLD was used for controller data-processing and VHDL to describe the logical circuit was optimized using the genetic algorithm. Two example cases (an inverted pendulum and an air-conditioner) were shown and we confirmed this framework was able to be applied to both systems. Since this framework is a generalized framework, so in these kinds of systems which process data from some sensors and drive some actuators, this framework will work functionally.
6. References


