Fuzzy logic and FPGA

Jaakko Yli-Luukko

13. joulukuuta 2011
1. About fuzzy logic

2. Fuzzy logic and FPGA

3. VHDL implementation
About fuzzy logic

Fuzzy logic in Wikipedia.
Fuzzy logic is a form of many-valued logic; it deals with reasoning that is approximate rather than fixed and exact. In contrast with traditional logic theory, where binary sets have two-valued logic: true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions.
Fuzzy logic began with the 1965 proposal of fuzzy set theory by Lotfi Zadeh.
Why to use FPGA in fuzzy controller?

- It can be done.
- Speed and parallel computation.
Once the basic design of the fuzzy logic control system has been defined, the implementation of the fuzzy logic controller is very straightforward by coding each component of the fuzzy inference system in VHDL according to the design specifications. [Vuong et.al.]
Trapetzoidal membership function:

[Vuong et.al.]
type membership is (term, none);
type mfs is record
  linguistic: membership;
  point1: std_logic_vector(7 downto 0);
  slope1: std_logic_vector(7 downto 0);
  point2: std_logic_vector(7 downto 0);
  slope2: std_logic_vector(7 downto 0);
end record;
type membership_functions is
  array(natural range <>) mfs;
constant linguistic_name :
  membership_functions :=
    ( (linguistic => term,
      point1 => x"04", slope1 => x"7F",
      point2 => x"09", slope2 => x"55"),
    linguistic => none,
    point1 => x"FF", slope1 => x"FF",
    point2 => x"FF", slope2 => x"FF")
);
Basic rule evaluation

**IF** $x$ is $A$ **AND** $y$ is $B$ **THEN** $z$ is $C$
function minimum(a, b: std_logic_vector) return std_logic_vector is
  variable min:
    std_logic_vector(7 downto 0)
    := (others => '0');

begin
  if a < b then min := a;
  else min := b;
  end if;
  return min;
end minimum;
function maximum(a, b: std_logic_vector)
    return std_logic_vector is
variable max: std_logic_vector(7 downto 0)
    := (others => '0');
begin
    if a > b then max := a;
    else max := b;
    end if;
    return max;
end maximum;

-- Combining minimum and maximum.
C <= maximum( minimum(A1, B1), minimum(A2, B2));
---

**Defuzzification**.

For $i = 1$ to $n$ do
begin
product = $(s(i) \times f(i)) + \text{product}$;
sum = $f[i] + \text{sum}$;
end for loop;
output = product / sum;
type te_type is (cold, cool, mild, warm, hot, none);
type te_membership is record term: te_type;
  point1: std_logic_vector(7 downto 0);
slope1: std_logic_vector(7 downto 0);
  point2: std_logic_vector(7 downto 0);
slope2: std_logic_vector(7 downto 0);
end record;
type te_membership_functions is array(natural range <>) te_membership;
constant te_mfs: te_membership_functions :=
  ((term => cold,
    point1 => x"00", slope1 => x"FF",
    point2 => x"2A", slope2 => x"06"),
  (term => cool,
    point1 => x"2A", slope1 => x"06",
    point2 => x"55", slope2 => x"06"),
  (term => mild,
    point1 => x"55", slope1 => x"06"},
point2 => x"7F", slope2 => x"06"),
(term => warm,
point1 => x"7F",
slope1 => x"06", point2 => x"AA",
slope2 => x"06"),
(term => hot,
point1 => x"AA", slope1 => x"06",
point2 => x"D5", slope2 => x"FF"),
(term => none,
point1 => x"FF", slope1 => x"FF",
point2 => x"FF", slope2 => x"FF"));
type tr_type is (slow, moderate, fast);
type tr_membership is record
  term : tr_type;
  point1 : std_logic_vector(7 downto 0);
  slope1 : std_logic_vector(7 downto 0);
  point2 : std_logic_vector(7 downto 0);
  slope2 : std_logic_vector(7 downto 0);
end record;
type tre_membership_functions is
  array(natural range <>) tr_membership;
constant tr_mfs : tre_membership_functions :=
  ((term => slow,
    point1 => x"00", slope1 => x"00",
    point2 => x"32", slope2 => x"03"),
   (term => moderate,
    point1 => x"32", slope1 => x"03",
    point2 => x"7E", slope2 => x"03"),
   (term => fast,
    point1 => x"7E", slope1 => x"03"),
point2 => x"FF", slope2 => x"FF")
(term => none,
point1 => x"FF", slope1 => x"FF",
point2 => x"FF", slope2 => x"FF")
);
constant very_low : std_logic_vector := x"2A";
constant low : std_logic_vector := x"55";
constant medium : std_logic_vector := x"7F";
constant high : std_logic_vector := x"A8";
constant very_high : std_logic_vector := x"D2";
type singletons is array (0 to 4) of
  std_logic_vector(7 downto 0);
signal ac : singletons :=
  (very_low, low, medium, high, very_high);
Challenges and tools

Optimization

- Genetic Algorithms
- Reusability of the code.
- Complexity of the code in larger systems.

Matlab fuzzy toolbox.
References I

- Wikipedia
  http://en.wikipedia.org/wiki/Fuzzy_logic

- Philip T. Vuong, Asad M. Madni and Jim B. Vuong
  VHDL Implementation For a Fuzzy Logic Controller

- Davi Nunes Oliveira, Arthur Plinio de Souza Braga, Otacílio da Mota Almeida
  Fuzzy Logic Controller Implementation on a FPGA using VHDL

- Sham Sagaria
  Fuzzy Logic Design using VHDL on FPGA