第7卷第11期
1996年11月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 7, No. 11
Nov. 1996

用XYZ/E语言描述和验证硬件的行为*

韩俊刚　　王岩冰

（西安邮电学院　西安　710061）　　（山东师范大学　济南　250014）

沈武威

（中国科学院软件研究所　北京　100080）

摘要　本文考虑用时态逻辑语言XYZ/E描述硬件行为的可行性.作为实例,用XYZ/E语言描述了一个基于微处理器的容错计算机系统,这种描述可以在XYZ系统上执行,从而可对系统进行模拟.特别有意义的是利用XYZ/VERI验证子系统对所期望的性质进行了形式化证明.本文还将XYZ/E描述与相应的VHDL(VHSIC hardware description language)描述进行了比较.从中可以看出时态逻辑语言的描述具有其独特的优点.

关键词　时态逻辑,硬件描述语言,形式化方法.

VLSI系统的日益复杂化使得传统的设计和分析方法难以保证设计的正确性.关于形式化验证的研究工作表明设计和验证应当集成在一起使得设计本身保证其结果的正确性(Correct by Construction).已经报道的若干这类系统都是基于高阶逻辑或Petri网的.[1,2,6]时态逻辑也已经用于软件和硬件的描述和验证[3,4],有人估计时态逻辑可能会成为工业开发方法的基础.由于硬件本身含有大量的并发性和非确定性,所以时态逻辑也适合于硬件的描述.XYZ/E特别适合于软件描述和验证的逐步求精、快速原型和分层描述方法.它比一般的线性时态逻辑优越之处是XYZ/E的描述可以经过编译后执行,描述的对象的性质可以用XYZ/VERI子系统加以证明.本文将探讨XYZ/E在硬件的形式化设计中应用的可能性.

形式化设计通常先从产生高级行为规范开始,然后通过保持行为正确性的转换到达较低的设计层次上并加入结构信息,这种正确性由形式化证明来保证,这种逐步求精的最终结果就是规范的完全的实现,所有这些变换和求精过程应当在一个统一的形式系统中进行.

本文第1节将用实例说明如何用XYZ/E描述硬件的行为,并与相应的VHDL(VHSIC hardware description language)描述进行比较.用XYZ/E描述一个基于微处理器的容错计算机系统,并通过在XYZ系统上运行模拟其行为.第2节将用XYZ/VERI来证明所描述的硬件的某些性质.最后将讨论本工作的某些结果和将来的研究方向.阅读本文需要XYZ/E

* 本文研究得到国家自然科学基金资助.作者**韩俊刚**,1943年生,教授,主要研究领域为CAD和硬件形式化设计验证方法.**王岩冰**,1967年生,博士生,主要研究领域为人工智能和语义学.**沈武威**,1967年生,助理研究员,主要研究领域为软件开发环境和并发程序设计.

本文通讯联系人:韩俊刚,西安 710061,西安邮电学院

本文1995-09-26收到修改稿

和 XYZ/SE 的一些知识，限于篇幅，对 XYZ/E 和 XYZ/SE 的介绍可参见文献[4].

1　用 XYZ/E 描述硬件的行为

我们先来说明用 XYZ/E 描述硬件的方法，然后给出实例. 由于硬件具有很好的模块性，其描述语言应能描述系统的模块及模块之间的相互作用. XYZ/E 的过程和进程则满足这种要求. 多数硬件系统具有并发的本质，并发操作应能方便地表达出来. 我们可以用 XYZ/E 的并发进程和它们之间的通道完成这些功能. 至于顺序操作，则可以用类型为%FSM 的程序体加以描述. 此外，我们用数组表示存储器，用向量表示寄存器等等. 我们将用一个具有适度的规模的实例说明描述方法.

1.1　基于微处理器的容错计算机系统

该系统可用作高可靠性的工业控制器，经过简化的系统框图如图 1 所示. 其容错功能是用表决技术实现的. 其中 3 个微处理器用来进行同样的并行计算. 计算结果分别通过 3 个端口送给表决器. 端口根据表决的结果来控制数据的传送. 每个端口通过 8 位数据总线 $D(0\sim7)$，控制线 S 和写控制信号线 WR，表决器的忙标志 B 和中断请求信号线 $INTR$ 与微处理器相连接，其中 S 代表微处理的状态，如图 2. 只有在忙标志变成 '0' 时(表示最近一次表决过程结束)，才能把计算的结果由微处理器传送给表决器，一旦传送完成后则端口把忙标志置为 '1'. 当 3 个忙标志都为 '1' 时，表决过程开始. 在正常情况下，表决的结果将导致数据被送到输出寄存器然后送到受控对象，并把 3 个忙标志复位为 '0'. 当检测到错误时，则端口向微处理器发出中断请求信号使其运行故障分析程序并使状态标志置为 '1'，使在此状态下中止数据的传送. 当只有 1 个处理器退出时，系统处于降级方式 1，此时表决器蜕化为 1 个错误检测器，当 2 个处理器退出时，系统处于降级状态 2，此时表决器成为 1 个数据通路.

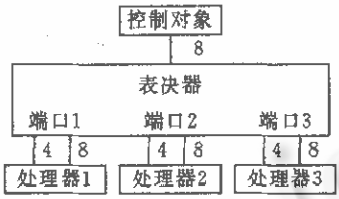

图1　系统框图

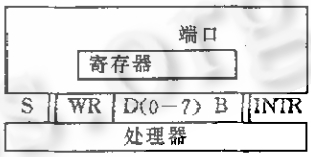

图2　端口结构

1.2　用 XYZ/E 描述系统

我们用 4 个进程描述系统，1 个用于表决器，3 个用于微处理器，表决器的形式进程可定义如下：

```
%PROC[voter (%CHN ch11 (pro1: MN, *), ch12 (pro2: MN, *), ch13 (pro3: MN,
*),          ch21 (pro1: MN), ch22 (*,pro2: MN,), ch23 (*, pro3: MN))
==[%LOC[dr:A(I;3); norm, int, q:A(I;3); outbuf,n,end:I]
        %FSM[.....]]]
```

其中以%FSM 开头的部分为程序体，这里省略，它描述表决算法. 在关键字%CHN 后面有 6 个通道，它们用来在进程之间传送消息. 对于每个处理器的形式进程，我们定义如下：

```
%POS[processor(%INP x11, x12,...,x25:I;
                %CHN ch1 (*,vot:MN),ch2(vot:MN,*))==[
```

%LOC[j1,j2:I;signal_in,signal_out:I;da:A(I;2;5);]

%FSM[.....]]]

其中在%FSM后面的程序单元描述在相关信号的控制下，通过通道发送和接收数据的过程.我们可以用如下的形式进行形式进程的动态实例化而产生3个进程.

$\bigcirc$ ProcessInstanceName==ProcessCall

即Processor1，Processor2和Processor3.这样我们就可用下面的条件元来并行地运行4个进程：　　label=11=> || [voter;processor1;processor2;processor3]

这4个进程的运行即可模拟整个硬件系统的行为，特别是验证并发行为是否为预期的.程序的细节和运行情况不再赘述.

1.3 采用VHDL的描述

VHDL(VHSIC hardware description language)是标准的硬件描述语言.[5]它用于电子系统的行为和内部联接关系的描述和归档，这种描述既可以作为模拟器的输入来验证系统的设计，也可以作为综合工具的输入来产生较低一级的设计并进行优化.但它的主要目的不是用来支持形式化设计和验证的.为了比较2种描述以理解XYZ/E语言的描述能力.这里给出上述的基于微处理器的容错计算机系统的VHDL描述.在VHDL中描述并发行为的关键语句是进程(PROCESS)，进程内部只包含顺序语句，而进程与进程可以并行地运行.进程之间的通讯是通过WAIT语句和信号(Signal)赋值来实现的.我们这里同XYZ/E的描述类似，采用了4个进程：VOTER，PROCESSOR1，PROCESSOR2和PROCESSOR3.整个VHDL描述程序结构如下.

```
    --declare the ports, signals and variables
entity FAULT_TOLERENT_SYSTEM is
port (S1, S2, S3:in BOOLEAN; --status bits for microprocessors
    W1, W2, W3:in BOOLEAN; - control signals for writing
    D1, D2, D3:in BIT_VECTOR(0 to 7);--data
    OUTPUT: out BIT_VECTOR(0 to 7);
    INT1, INT2, INT3: out BOOLEAN; --interrupt signals
    B1, B2, B3:out BOOLEAN); --busy tags for the voter
    end FAULT_TOLERANT_SYSTEM;
architecture BEHAVIOR of FAULT_TOLERANT_SYSTEM is
    constant DATA_LENGTH:INTEGER:=4;
    signal BB1, BB2, BB3:BOOLEAN:=false;
    variable DR1, DR2, DR3, OUTBUF:BIT_VECTOR(0 to 7);
    variable NORM1, NORM2, NORM3: BOOLEAN:=true;
    variable n: INTEGER:=0;
begin
    VOTER: process
      begin
        wait until (S1 or BB1) and (S2 or BB2) and (S3 or BB3);
        ......
      end process;
PROCESSOR1: process
    begin
      wait until S1 or W1;
      ......
     end process;
PROCESSOR2: process
    begin
```

```
    wait until S2 or W2;
    ......
    end process;
PROCESSOR3: process
  begin
    wait until S3 or W3;
    ......
    end process;
end BEHAVIOR;
```

注意信号 $BB1$，$BB2$ 和 $BB3$ 的值总是分别跟着 $B1$，$B2$ 和 $B3$ 的值改变的，比较 2 种描述可见 XYZ/E 语言具备了 VHDL 语言描述并发行为的能力和机制.

2　硬件行为的验证

由于 XYZ/E 是可编译执行的命令式语言，因而上述的硬件描述可在 XYZ 系统上运行，通过适当的输入和输出即可观测描述对象的运行情况，这实际上是行为模拟. 又由于 XYZ/E 是一个逻辑系统，我们可以利用其验证子系统证明有关描述对象的性质的逻辑断言，这里只做简单介绍.

• Succession Statement

$$\vdash\{R[e1/x1,\ldots,en/xn]\}LB=l1 \Rightarrow \$\bigcirc(x1=e1\wedge\ldots\wedge xn=en\wedge LB=NEXT)\{R\}$$

• Sequential Composition

$$\frac{\vdash\{R1\}LB=l1[|X|]\$\bigcirc LB=l3\{R2\}\quad \vdash\{R2\}LB=l3[|X|]\$\bigcirc LB=l2\{R3\}}{\vdash\{R1\}LB=l1[|X;Y|]\$\bigcirc LB=l2\{R3\}}$$

• Loop－block

$$\frac{\vdash\{INV\wedge P\}LB=w\{[|R|]\$\bigcirc LB=y\{INV\}}{\vdash\{INV\}LB=y\{[|X|]\$\bigcirc LB=EXIT\{\sim P\wedge INV\}}$$

• Case－block

$$\frac{\begin{array}{l}\vdash\{Pre\wedge P1\}LB=z1\{[|Q1|]\$\bigcirc LB=EXIT\{Post\}\\ \quad\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ \vdash\{Pre\wedge Pk\}LB=zk\{[|Qk|]\$\bigcirc LB=EXIT\{Post\}\end{array}}{\vdash\{Pre\}LB=y\{[|X|]\$\bigcirc LB=EXIT\{Post\}}$$

上述的规则与 Hoare 逻辑的规则在形式上是类似的. 它们的正确性和完全性已得到证实. 我们还可以通过把并行程序变为顺序形式来证明并发程序的性质. 下面解释验证的主要思想.

为了使下面的证明规则

$$\frac{\begin{array}{l}\vdash\{P1\}\ ProsInstId1\{Q1\}\\ \quad\ldots\ldots\ldots\ldots\\ \vdash\{Pn\}\ ProsInstIdn\{Qn\}\end{array}}{\vdash\{P1\wedge\ldots\wedge Pn\}||[ProsInstId1,\ldots,ProsInstIdn]\{Q1\wedge\ldots\wedge Qn\}}$$

成立，我们要把每个输入命令转换成适当的形式，例如可将下面的命令

LB=y=>C? x∧ $○LB=z

转换为下列形式

LB=y=>C? x∧ $○LB=NEXT;

LB=y′ ∧ ~pre(x) =>$○LB=y;

LB=y′ ∧ pre(x) =>$○LB=z;

其中 *pre*(*x*)是对输入参数 *X* 的前断言.

2.1 证明实例

我们来说明如何验证前述基于微处理器的容错计算机系统的行为,为简明起见只考虑其关键部分 Voter 的行为的验证,为了说明方便,程序的关键部分做下述改写,其中省去了有关变量的说明.

```
%PROC Voter(...)==[ ][
%LOC[...]
%FSM[
  LB=START=>$○LB=l1;
  ? [LB=l1 ∧ (q[1]=1 ∧ Norm[1]=1)=>$○LB=l2 |~=>○LB=l3;
    ? [LB=l2 ∧ (q[2]=1 ∧ Norm[2]=1)=>$○LB=l4 |~=>○LB=l5;
      ? [LB=l4 ∧ (q[3]=1 ∧ Norm[3]=1)=>$○LB=l6 |~=>○LB=l7;
        ? [LB=l6 ∧ (dr[1]=dr[2])=>$○LB=l8 |~=>○LB=l9;
          ? [LB=l8 ∧ (dr[1]=dr[3])=>$○LB=l10 |~=>○LB=l11;
            LB=l10=>SWRITELN("1,2,3 Normal") ∧ $○LB=NEXT;
            LB=l11=>SWRITELN("1,2 Normal; 3 Interrupt") ∧ $○LB=NEXT;]
          ? [LB=l9 ∧ (dr[1]=dr[3])=>$○LB=l12 |~=>○LB=l13;
              LB=l12=>SWRITELN("1,3 Normal; 2 Interrupt") ∧ $○LB=NEXT;
          ? [LB=l13 ∧ (dr[2]=dr[3])=>$○LB=l14 |~=>○LB=l15;
             LB=l14=>SWRITELN("2,3 Normal; 1 Interrupt") ∧ $○LB=NEXT;
             LB=l15=>SWRITELN("1,2,3 Interrupt") ∧ $○LB=NEXT;]]
          ? [LB=l7 ∧ (dr[1]=dr[2])=>$○LB=l16 |~=>○LB=l17;
              LB=l16=>SWRITELN("1,2, Normal; 3 fail") ∧ $○LB=NEXT;
              LB=l17=>SWRITELN("1,2, Interrupt; 3 fail") ∧ $○LB=NEXT;]
          ? [LB=l5 ∧ (q[3]=1 ∧ Norm[3]=1)=>$○LB=l18 |~=>○LB=l19;
          ? [LB=l18 ∧ (dr[1]=1 ∧ dr[3]=1)=>$○LB=l20 |~=>○LB=l21;
              LB=l20=>SWRITELN("1,3, Normal; 2 fail") ∧ $○LB=NEXT;
              LB=l21=>SWRITELN("1,3, Interrupt; 2 fail") ∧ $○LB=NEXT;]
              LB=l19=>SWRITELN("1 Normal; 2,3 fail") ∧ $○LB=NEXT;]]
            ? [LB=l3 ∧ (q[2]=1 ∧ Norm[2]=1)=>$○LB=l22 |~=>○LB=l23;
              ? [LB=l22 ∧ (q[3]=1 ∧ Norm[3]=1)=>$○LB=l24 |~=>○LB=l25;
              ? [LB=l24 ∧ (dr[2]=1 ∧ dr[3])=>$○LB=l26 |~=>○LB=l27;
                LB=l26=>SWRITELN("2,3, Normal; 1 Fail") ∧ $○LB=NEXT;
                LB=l27=>SWRITELN("2,3, Interrupt; 1 Fail") ∧ $○LB=NEXT;]
                LB=l25=>SWRITELN("2 Normal; 1,3 Fail") ∧ $○LB=NEXT;]
            ? [LB=l23 ∧ (q[3]=1 ∧ Norm[3]=1)=>$○LB=l28 |~=>○LB=l29;
                LB=l28=>SWRITELN("3, Normal; 1,2 Fail") ∧ $○LB=NEXT;
                LB=l29=>SWRITELN("1,2,3, Fail") ∧ $○LB=NEXT;]]
                LB=l30=>$○LB=EXIT;]]].
```

我们定义下列 3 个谓词:

Int(*i*) 为真,表示微处理器 *i* 发出中断信号;

Norm(*i*) 为真,表示微处理器 *i* 正常工作;

Fail(*i*) 为真,表示微处理器 *i* 不正常且退出系统.

对于 Voter 进程前断言虽然为真,而后断言可表示为如下形式:

(Norm(1) ∧ Norm(2) ∧ Norm(3)) $ V (Norm(1) ∧ Norm(2) ∧ Int(3)) $ V

(Norm(1) ∧ Int(2) ∧ Norm(3)) $ V (Int(1) ∧ Norm(2) ∧ Norm(3)) $ V

(Int(1) ∧ Int(2) ∧ Int(3)) $ V (Norm(1) ∧ Norm(2) ∧ Fail(3)) $ V

(Int(1) ∧ Int(2) ∧ Fail(3)) $ V (Norm(1) ∧ Fail(2) ∧ Norm(3)) $ V
(Int(1) ∧ Fail(2) ∧ Int(3)) $ V (Norm(1) ∧ Fail(2) ∧ Fail(3)) $ V
(Fail(1) ∧ Norm(2) ∧ Norm(3)) $ V (Fail(1) ∧ Int(2) ∧ Int(3)) $ V
(Fail(1) ∧ Norm(2) ∧ Fail(3)) $ V (Fail(1) ∧ Fail(2) ∧ Norm(3)) $ V
(Fail(1) ∧ Fail(2) ∧ Fail(3))

这里需要说明的是在前面的 XYZ/E 程序中，为了增加可读性，加入了如下的语句：
SWRITELN("i Interrupt"),
SWRITELN("i Fail"), and
SWRITELN("i Norm"),

实际上它们分别表示谓词 $Int(i)$，$Fail(i)$ 和 $Norm(i)$. 另外，$q(i)=1 \wedge Norm(i)=1$ 表示处理 i 自检正常，即 $\sim Fail(i)$.

为了上述 *XYZ/E* 程序的证明，还需要如下的证明规则：

$$\frac{\vdash P \rightarrow P_1, \{P_1\} LB=y\{|X|\} \$ \bigcirc LB=z\{Q_1\}, Q_1 \rightarrow Q}{\vdash \{P\} LB=y\{|X|\} \$ \bigcirc LB=z\{Q\}}$$

利用上面的规则，我们只要考虑下列程序段的证明：

$?[LB=18 \wedge (dr[1]=dr[3] => \$ \bigcirc LB=l10 \mid\sim = \$ \bigcirc LB=l11;$
$LB=l10 => SWRITELN("A11\ normal") \wedge \$ \bigcirc LB=NEXT;$
$LB=l11 => SWRITELN("3\ Interrupt") \wedge \$ \bigcirc LB=NEXT;]$

其它部分的证明用类似的方法进行. 这段程序的前断言(Pre-condition)为

$P=Norm\ (1) \wedge Norm\ (2)$

其后断言(Post-condition)为

$Q=(Norm\ (1) \wedge Norm(2) \wedge Norm(3)\ \$\ V(Norm\ (1) \wedge Norm(2) \wedge Int(3))$

由下面的 2 个子断言：

(1) $\{Norm\ (1) \wedge Norm(2) \wedge dr[1]=dr[3]\}$
$LB=l10 => SWRITELN("ALL\ Norm") \wedge \$ \bigcirc LB=NEXT$
$\{Norm\ (1) \wedge Norm(2) \wedge Norm(3)\}$

(2) $\{Norm\ (1) \wedge Norm(2) \wedge \sim dr[1]=dr[3]\}$
$LB=l11 => SWRITELN("3\ Interrupt") \wedge \$ \bigcirc LB=NEXT$
$\{Norm\ (1) \wedge Norm(2) \wedge Int(3)\}$

及上述证明规则显然可推出断言 Q. 因此 P 和 Q 分别为上述程序段的前后断言. 同样方法可推出 *Voter* 满足前面给出的断言.

3　小　结

本文给出了用 XYZ/E 描述硬件行为的实例，并与相应的 VHDL 描述进行了比较，对于基于微处理器容错计算机系统，其 XYZ/E 描述大约有 150 行程序，其中含有 I/O 进程和读写数据文件的语句，而相应的 VHDL 描述大约 100 行. 后者的进程通讯采用讯号显得更为方便，但 XYZ/E 描述最大优点是能够用 XYZ/VERI 验证系统对程序进行验证，从而描述对象的一些重要性质可以得到证明. 对硬件设计者来说，写出 XYZ/E 程序不是容易的. 我

们准备构造一个从 VHDL 到 XYZ/E 的自动翻译程序使得设计对象的 XYZ/E 描述可以由 VHDL 描述自动生成. 这样就可能把形式化设计和验证方法集成到商用 EDA 系统中去.

致谢 清华大学薛宏熙教授为我们提供了基于微处理器的容错计算机系统的 EC 语言描述,这是他在 Toronto 大学的研究成果.

参考文献

1 Hanna F K *et al*. Formal synthesis of digital system. In: Claesen L J M ed. Formal VLSI Specification and Synthesis, VLSI Design Method I, Elsevier Science Publisher, B. V. (North-Holland), 1990.

2 Staunstrup J. Formal method for VLSI design. IFIP, 1990.

3 Bochmann G. Hardware specification with temporal logic: an example. IEEE Transaction on Computer, 1982, C-31 (3).

4 Tang C S. A temporal logic language toward software engineering. Tech. Rep. No. CAS-IS-XYA-93-12, Institute of Software, The Chinese Academy of Sciences, 1993.

5 Stanley M, Patrcia L. Introduction to VHDL. KAP, 1992.

6 Peng Z *et al*. Automated transformation of algorithms into register-transfer level implementations. IEEE Trans. on CAD, Feb. 1994, 13(2): 150～166.

DESCRIBING AND VERIFYING THE BEHAVIOR OF HARDWARE IN XYZ/E LANGUAGE

Han Jungang

(*Xi'an Institute of Posts and Telecommunications Xi'an* 710061)

Wang Yanbing

(*Shandong Normal University Ji'nan* 250014)

Shen Wuwei

(*Institute of Software The Chinese Academy of Sciences Beijing* 100080)

Abstract The necessities of VLSI design by formal method lead to consider the feasibility of specifying hardware behavior using temporal logic language XYZ/E which is the basis of the CASE tool system XYZ. A microprocessor based fault-tolerant computer system is described behaviorally, simulated by executing the description on XYZ system, and the intended properties have been formally verified by using the XYZ/VERI. The comparison between XYZ/E and VHDL(VHSIC hardware description language) has been made. The results have shown that XYZ/E is promising for the research of formal method for hardware design.

Key words Temporal logic, hardware description language, formal design method.