(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102163248 A
(43)申请公布日 2011.08.24

(21)申请号 201110084177.0

(22)申请日 2011.04.02

(71)申请人 北京大学深圳研究生院
地址 518055 广东省深圳市南山区西丽深圳大学城北大校区

(72)发明人 蓝晶 王新安 雍珊珊 吴承昊 龙晓波

(74)专利代理机构 深圳鼎合诚知识产权代理有限公司 44281
代理人 宋鹰武

(51)Int.Cl.
G06F 17/50(2006.01)

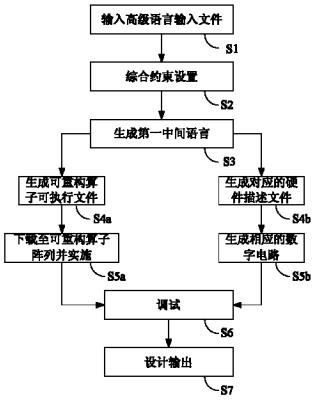
权利要求书 2 页 说明书 10 页 附图 14 页

(54)发明名称

一种集成电路的高级综合方法

(57)摘要

本发明公开了一种集成电路的高级综合方法，通过综合生成第一中间语言，再由第一中间语言生成相应的可重构算子可执行文件或者硬件描述文件，从而输出具有多目标的特性，即既可以通过生成可重构算子阵列可执行文件把高级语言输入文件所描述的系统下载到可重构算子阵列上实施，又可以通过生成硬件描述文件把输入文件所描述的系统下载到FPGA或ASIC上实施。



CN 102163248 A

1. 一种集成电路的高级综合方法，其特征在于，包括步骤：

输入描述数字电路的高级语言输入文件，并进行综合约束设置；

根据所述数字电路的高级语言输入文件和综合约束，生成第一中间语言，所述第一中间语言的每一条语句代表一个特定的数字电路结构及各数字电路结构间的连接关系；

将所述第一中间语言转换为可重构算子可执行文件，并下载至所述可重构算子阵列中实施，或者将所述第一中间语言转换为对应的硬件描述文件，并根据所述硬件描述文件生成相应的数字电路。

2. 如权利要求 1 所述的方法，其特征在于，生成所述第一中间语言包括步骤：

将输入的所述高级语言输入文件和综合约束进行预综合，生成第三中间语言，所述第三中间语言为由带有列表接口的语句块或者带有列表接口的嵌套语句块构成的主函数；

将所述第三中间语言进行综合前端，综合生成抽象语法树，所述抽象语法树包含所述高级语言输入文件所描述的语句块信息、语句块之间接口信息与输入文件所描述的数字电路系统输入输出接口信息的；

遍历所述抽象语法树，并按照预定的综合策略转换为第二中间语言，所述第二中间语言的每一条语句都代表一个功能模块与此功能模块与其他模块的连接关系；

将所述第二中间语言进行综合后端，生成第一中间语言，并进行仿真。

3. 如权利要求 2 所述的方法，其特征在于，所述预定的综合策略包括综合后输出系统的运算速度、功率消耗、实际使用的电路面积、各类电路资源相对比例、目标可重构算子阵列的硬件结构、输入文件语言格式、电路资源使用策略。

4. 如权利要求 2 所述的方法，其特征在于，所述抽象语法树包括根节点；第一节点，用于存储所述高级语言输入文件所描述系统的输入输出信息；第二节点，用于存储高级语言输入文件中的全局变量或数组的声明信息；第三节点，用于存储高级语言输入文件中的所有函数的信息，所述第一节点、第二节点和第三节点均与所述根节点相连，则遍历所述抽象语法树生成所述第二中间语言包括步骤：

读取抽象语法树的每一个节点，得到每一个节点的信息；

收集抽象语法树中与函数语句块相关的信息，并存储；

根据所述节点信息和函数信息，检查所述抽象语法树生成是否正确，如是，则根据读取的节点信息发射第二中间语言。

5. 如权利要求 2 所述的方法，其特征在于，对所述第一中间语言进行仿真包括步骤：

在仿真系统启动时配置仿真系统运行参数，包括输入待仿真的第一中间语言、输入仿真数据文件、输出仿真数据文件、仿真系统运行时间等；

读取输入待仿真的第一中间语言的系统描述文件，生成系统描述文件中描述的各子模块；

将输入仿真数据文件中的数据输入到系统的数据存储容器中，等待各子模块启动；

读取输入待仿真的系统描述文件，连接生成的所述所有子模块；

启动整个仿真的运行，即启动所有子模块工作；

输出仿真数据文件中所描述的输出数据到指定文件。

6. 如权利要求 1 所述的方法，其特征在于，将所述第一中间语言转换为可重构算子可执行文件包括步骤：

根据所述第一中间语言在可重构算子阵列上进行布局布线；

对布局布线后的可重构算子阵列进行仿真与验证；

将经过验证后的输出的文件转换为可重构算子可执行文件。

7. 如权利要求 6 所述的方法，其特征在于，根据所述第一中间语言再可重构算子阵列上进行布局布线包括步骤：

根据所述第一中间语言的每条语句中的特定数字电路结构确定其在可重构算子阵列中的实施位置；

再根据所述第一中间语言的各个语句中各特定数字电路结构之间的连接关系，确定完成各特定的数字电路结构之间的连接关系所消耗的电路资源情况。

8. 如权利要求 1 所述的方法，其特征在于，所述第一中间语言的每一条语句包括功能信息和连接信息，则将所述第一中间语言转换为相应的硬件描述文件包括步骤：

主要读取所述第一中间语言的功能信息和连接信息；

根据所述功能信息查找硬件描述语言模型库，得到相应的硬件描述语言描述；

结合所述连接信息，生成相应的硬件描述语言文件。

一种集成电路的高级综合方法

技术领域

[0001] 本发明涉及集成电路设计技术领域，尤其涉及一种集成电路的高级综合方法。

背景技术

[0002] 随着通信、计算机、消费电子等领域的快速发展，对承载这些应用的硬件系统在性能、功耗、成本、上市时间、灵活性、可扩展性等方面提出了更高的要求，传统的设计方法如 ASIC(Application Specific Intergrated Circuits，专用集成电路)，DSP(Digital Signal Processing，数字信号处理器)/CPU 以及 FPGA(Field Programmable Gate Array，现场可编程门阵列) 都面临着一些问题。

[0003] ASIC 设计的复杂性和规模不断提升，使得 ASIC 设计成本增加，设计周期变长，同时由于不具灵活性，不可扩展，产品的快速上市和低成本的需求使得 ASIC 设计遇到了瓶颈。而 CPU/DSP 依靠复杂的体系结构和指令系统的方法得到性能提升所需要付出的代价越来越大，单个处理器的处理能力已经远远不能满足当前应用的需要。FPGA 作为并行计算较早的应用，它的性能和能耗比介于 ASIC 和 DSP/CPU 之间，具有一定的灵活性，可反复编程，能够满足快速上市和低成本的应用需求。但是 FPGA 的设计并不支持从算法到硬件的直接映射，设计需要算法人员和硬件人员的共同参与，硬件描述语言的抽象层次较低，应用的开发仍然存在着很大的复杂性。

[0004] 北京大学深圳研究生院集成微系统实验室提出的一种基于并行计算技术的统一架构的可重构算子阵列结构，图 1 为 APU(Array Processing for UnificationArchitecture) 的总体结构图。该 APU 结构 1000 由算术类可重构算子 1001、路径类可重构算子 1002、调度类可重构算子 1003、DSP 类可重构算子 1004、存储类可重构算子 1005 以及 IO1006 组成。在 APU 结构 1000 的内部，算术类可重构算子 1001、路径类可重构算子 1002、调度类可重构算子 1003、DSP 类可重构算子 1004、存储类可重构算子 1005 按照一定的比例，按照类型独立地以列为单位，分布式交叉排列。APU 支持大量数据并行 / 串行的运算和传输需求，并且能够支撑多种应用实现的需要。

[0005] APU 应用设计描述语言为高级语言，抽象层次较高，缩短了应用开发时间。故须引入对应的高级综合方法以实施应用。

[0006] 常规的高级综合方法根据输入文件和约束文件只能生成对应的数字电路信息，生成的数字电路信息通常只能在 FPGA 上实施或转换为对应的 ASIC。

[0007] 常规高级综合方法把输入文件的语言元素转换成 CDFG(Control Data FlowGraph，数据控制流图)，再给每个 CDFG 节点分配对应的时间节点信息，输出文件的发射过程把 CDFG 对应的时间节点信息转换成数字电路的状态机信息。由于数字电路状态机的状态切换有一定时间间隔，所以常规高级综合方法相当于通过时间标识调度输入文件所描述的系统，执行效率和灵活性都很难拓展。

发明内容

[0008] 本发明要解决的主要技术问题是，提供一种集成电路的高级综合方法，其输出具有多目标性，且具有灵活、可执行性高、执行效率高的特点，同时其输出既可以在可重构算子阵列上实施，也可以下载到 FPGA 或 ASIC 上实施。

[0009] 为解决上述技术问题，本发明采用的技术方案如下：

[0010] 一种集成电路的高级综合方法，包括步骤：

[0011] 输入描述数字电路的高级语言输入文件，并进行综合约束设置；

[0012] 根据所述数字电路的高级语言输入文件和综合约束，生成第一中间语言，所述第一中间语言的每一条语句代表一个特定的数字电路结构及各数字电路结构间的连接关系；

[0013] 将所述第一中间语言转换为可重构算子可执行文件，并下载至所述可重构算子阵列中实施，或者将所述第一中间语言转换为对应的硬件描述文件，并根据所述硬件描述文件生成相应的数字电路。

[0014] 进一步地，生成所述第一中间语言包括步骤：

[0015] 将输入的所述高级语言输入文件和综合约束进行预综合，生成第三中间语言，所述第三中间语言为由带有列表接口的语句块或者带有列表接口的嵌套语句块构成的主函数；

[0016] 将所述第三中间语言进行综合前端，综合生成抽象语法树，所述抽象语法树包含所述高级语言输入文件所描述的语句块信息、语句块之间接口信息与输入文件所描述的数字电路系统输入输出接口信息的；

[0017] 遍历所述抽象语法树，并按照预定的综合策略转换为第二中间语言，所述第二中间语言的每一条语句都代表一个功能模块与此功能模块与其他模块的连接关系；

[0018] 将所述第二中间语言进行综合后端，生成第一中间语言。

[0019] 进一步地，所述预定的综合策略包括综合后输出系统的运算速度、功率消耗、实际使用的电路面积、各类电路资源相对比例、目标可重构算子阵列的硬件结构、输入文件语言格式、电路资源使用策略。

[0020] 进一步地，所述抽象语法树包括根节点；第一节点，用于存储所述高级语言输入文件所描述系统的输入输出信息；第二节点，用于存储高级语言输入文件中的全局变量或数组的声明信息；第三节点，用于存储高级语言输入文件中的所有函数的信息，所述第一节点、第二节点和第三节点均与所述根节点相连，则遍历所述抽象语法树生成所述第二中间语言包括步骤：

[0021] 读取抽象语法树的每一个节点，得到每一个节点的信息；

[0022] 收集抽象语法树中与函数语句块相关的信息，并存储；

[0023] 根据所述节点信息和函数信息，检查所述抽象语法树生成是否正确，如是，则根据读取的节点信息发射第二中间语言。

[0024] 进一步地，对所述第一中间语言进行仿真包括步骤：

[0025] 在仿真系统启动时配置仿真系统运行参数，包括输入待仿真的第一中间语言、输入仿真数据文件、输出仿真数据文件、仿真系统运行时间等；

[0026] 读取输入待仿真的第一中间语言的系统描述文件，生成系统描述文件中描述的各子模块；

[0027] 将输入仿真数据文件中的数据输入到系统的数据存储容器中，等待各子模块启动；

[0028] 读取输入待仿真的系统描述文件，连接生成的所述所有子模块；

[0029] 启动整个仿真的运行，即启动所有子模块工作；

[0030] 输出仿真数据文件中所描述的输出数据到指定文件。

[0031] 进一步地，将所述第一中间语言转换为可重构算子可执行文件包括步骤：

[0032] 根据所述第一中间语言在可重构算子阵列上进行布局布线；

[0033] 对布局布线后的可重构算子阵列进行仿真与验证；

[0034] 将经过验证后的输出的文件转换为可重构算子可执行文件。

[0035] 进一步地，根据所述第一中间语言再可重构算子阵列上进行布局布线包括步骤：

[0036] 根据所述第一中间语言的每条语句中的特定数字电路结构确定其在可重构算子阵列中的实施位置；

[0037] 再根据所述第一中间语言的各个语句中各特定数字电路结构之间的连接关系，确定完成各特定的数字电路结构之间的连接关系所消耗的电路资源情况。

[0038] 进一步地，所述第一中间语言的每一条语句包括功能信息和连接信息，则将所述第一中间语言转换为相应的硬件描述文件包括步骤：

[0039] 主要读取所述第一中间语言的功能信息和连接信息；

[0040] 根据所述功能信息查找硬件描述语言模型库，得到相应的硬件描述语言描述；

[0041] 结合所述连接信息，生成相应的硬件描述语言文件。

[0042] 本发明的有益效果是：本发明的高级综合方法，包括输入描述数字电路的高级语言输入文件，并对其进行综合约束设置；再根据所述数字电路的高级语言输入文件和综合约束，生成第一中间语言，所述第一中间语言的每一条语句代表一个特定的数字电路结构及各数字电路结构间的连接关系；然后将所述第一中间语言转换为可重构算子可执行文件，并下载至所述可重构算子阵列中实施，或者将所述第一中间语言转换为对应的硬件描述文件，并根据所述硬件描述文件生成相应的数字电路。本发明的高级综合方法通过生成第一中间语言，再由该第一中间语言生成可重构算子阵列可执行文件，从而把高级语言输入文件所描述的系统下载到可重构算子阵列上实施，或者根据该第一中间语言生成硬件描述文件，从而把高级语言输入文件所描述的系统下载到FPGA或ASIC上实施，即本发明的高级综合方法的输出具有多目标性，且灵活性、可执行性高、执行效率高。

[0043] 另一方面，本发明的高级综合方法，通过将高级输入文件转换为抽象层次高的第三中间语言，再由该第三中间语言转换为抽象层次较低的第二中间语言，该第二中间语言接近于模块化的数字电路描述，再由该第二中间语言转换为第一中间语言，最后由该第一中间语言直接转换为硬件描述语言或可重构算子阵列。本发明采用模块化思想的第一中间语言、第二中间语言和第三中间语言作为转换过程的中间表示形式，并着重描述具有一定功能模块的特性与模块间的连接关系和通信关系。这种高级综合过程通过模块特性、模块间的连接关系和模块间的通信机制来调度高级语言输入文件所描述的系统，从而生成的可重构算子阵列配置信息或者数字电路信息灵活、可执行性高，且执行效率高。

附图说明

[0044] 图 1 为一种可重构算子阵列的结构示意图；

[0045] 图 2 为本发明的集成电路的高级综合方法的一种实施例的流程图；

[0046] 图 3 为本发明的集成电路的高级综合方法中生成第一中间语言的一种实施例的流程图；

[0047] 图 4 为本发明的集成电路的高级综合方法中生成的第三中间语言的语言结构的一种实施例的示意图；

[0048] 图 5 为本发明的集成电路的高级综合方法中生成的第三中间语言的语句块的组成结构的一种实施例的示意图；

[0049] 图 6a 和图 6b 分别为本发明的高级语言输入文件的一种实施例，和由高级语言输入文件对应生成的第三中间语言的一种实施例；

[0050] 图 7 为本发明的对高级语言输入文件进行预综合的一种实施例的流程图；

[0051] 图 8 为本发明的集成电路的高级综合方法中生成的抽象语法树的总结构的一种实施例的示意图；

[0052] 图 9 为图本发明的集成电路的高级综合方法中生成的第三节点的子节点的组成结构的一种实施例的示意图；

[0053] 图 10 为本发明的集成电路的高级综合方法中生成第二中间语言的一种实施例的流程图；

[0054] 图 11 为本发明的集成电路的高级综合方法中生成的抽象语法树的一种具体实施例的结构示意图；

[0055] 图 12 为本发明的集成电路的高级综合方法中对生成的第一中间语言进行仿真的一种实施例的流程图；

[0056] 图 13a 和图 13b 分别为本发明的集成电路的高级综合方法中生成的第二中间语言描述文件的一种实施例，和由该第二中间语言描述文件生成的第一中间语言描述文件的一种实施例；

[0057] 图 14 为本发明的集成电路的高级综合方法中对生成的第一中间语言进行仿真的一种实施例的流程图；

[0058] 图 15 为本发明的集成电路的高级综合方法中生成可重构算子阵列可执行文件的一种实施例的流程图；

[0059] 图 16 为本发明的集成电路的高级综合方法中生成硬件描述语言文件的一种实施例的流程图；

[0060] 图 17a 和图 17b 分别为本发明的集成电路的高级综合方法生成的第一中间语言描述文件的一种实施例，和由该第一中间语言描述文件转换为硬件描述文件的一种实施例。

具体实施方式

[0061] 下面通过具体实施方式结合附图对本发明作进一步详细说明。

[0062] 请参考图 2，本实施方式的集成电路的高级综合方法，包括步骤：

[0063] S1，输入描述数字电路的高级语言输入文件。

[0064] 本实施方式中的高级语言输入文件为描述数字电路情况的输入文件，其可以为 C

语言或者其他高级语言，如 Java，C++，Matlab 等。

[0065] S2，进行综合约束设置。

[0066] 本实施方式中的综合约束以文件输入或者键盘输入等人机界面实现。

[0067] 本实施方式通过综合约束设置约束了系统设计输出的运算速度、功率消耗、实际使用的电路面积、各类电路资源相对比例、目标可重构算子阵列的硬件结构、输入文件语言格式、电路资源使用策略。其中，电路资源使用策略是指在实现同样功能的情况下，在多种不同的电路资源使用方案中的选择策略，不同的方案间使用的电路模块类型和不同电路模块类型之间的比例都不一样。

[0068] S3，根据数字电路的高级语言输入文件和综合约束，生成第一中间语言。

[0069] 本实施方中的第一中间语言的每一条语句代表一个特定的数字电路结构及各数字电路结构间的连接关系。

[0070] 请参考图 3，本实施方式中，步骤 S3 中由高级语言输入文件生成第一中间语言包括步骤：

[0071] S31，将输入的高级语言输入文件和综合约束进行预综合，生成第三中间语言。即将高级语言输入文件转换为由带有列表接口的语句块或者带有列表接口的嵌套语句块构成的主函数，即第三中间语言。

[0072] 本实施方式中的抽象句法树是一个树型的数据存储结构，它包含数字电路情况的高级语言输入文件所描述的语句块信息、语句块之间接口信息与输入文件所描述的数字电路系统输入输出接口信息。

[0073] 请参考图 4，为本实施方式的第三中间语言的语言结构图，最顶层的描述是主函数块 301，主函数 301 完成了对高级语言输入文件所描述系统的描述。该主函数 301 中包含了各子语句块 302，高层的语句块 302 可以由底层的语句块 302 组成，其中，最底层子语句块 302 由各语句 303 或各类控制结构 304 组成。

[0074] 当然，本实施方式中的语句块 302 之间还可互相嵌套，如图 5 所示。

[0075] 本实施方式中的语句 303 可以是常数赋值语句，即把一个常数赋值给一个变量；该语句 303 也可以是运算赋值语句，即把某些变量运算的结果赋值给一个变量。

[0076] 在本实施方式的第三中间语言一种具体实施例中：

[0077]

```
const1=2;
const2=4;
if（ip1>ip2） #info（op2 ； op3 ； op1 ； ip3，ip4 ； ip3，ip4 ； ip3，ip4）
  {
    op1=ip3+ip4;
    op2=ip3*ip4*const1;
  }
else
  {
    op1=ip3-ip4;
    op3=ip3+ip4*const2;
  }
```

[0078] 其中“const2 ＝ 4 ； ”即为常数赋值语句 303 ；“op1 ＝ ip3+ip4 ； ”即为运算赋值语句 303。

[0079] 请参考图 4，本实施方式中的控制结构 304 分为 ：分支结构、循环结构和复用结构。其中分支结构由 if 语句块实现，循环结构由 while 语句块实现，复用结构由函数实现。该控制结构 304 主要包括控制结构的关键字，控制结构的执行体，该执行体描述控制结构的功能，以及列表接口 305，列表接口 305 由变量或数组以一定的集合组成。请参考图 5，本实施方式中的列表接口 305 用于描述语句块和语句块之间的接口信息，语句块之间通过该列表接口 305 进行通信。

[0080] 本实施方式中的循环结构由第三中间语言实现的一种具体实施例中 ：

[0081]

```
while（i==const）#info（a，i； b，const； const； i）
    {
```

[0082]

```
        a=b+i;
        i=i+c_1;
    }
```

[0083] 其中，while 为关键字，代表循环 ；“{a ＝ b+i ；i ＝ i+c_1 ；} ”即为该控制结构 304 的执行体，描述了该控制结构 304 的功能 ；“#info(a,i ;b,const ;const ;i)”即为该控制结构 304 的列表接口。

[0084] 请参考图 6a，为输入高级语言输入文件的一种实施例的节选，其中 op1，op2，op3，wr_tmp1，wr_tmp2 等变量均为全局变量，数组 a_ar[]、b_ar[] 均为全局数组。请参考图 7，对图 6a 所示的输入高级语言文件进行预综合，从而转换为第三中间语言的一种具体实施

例包括步骤：

[0085] S701、检测该高级语言输入文件。

[0086] S702、当检测到上述的子函数 testFun 后，检测该子函数 testFun 中被读取的数组，即出现在“＝”的右边的数组，并根据该数组相应生成列表接口的第一个元素 a_ar。

[0087] S703，检测该子函数 testFun 中被写入的数组，即出现在”＝”左边的数组，并根据该数据生成列表接口的第二个元素 b_ar。

[0088] S704，检测被赋予子函数 testFun 的返回值的变量，并根据该变量生成列表接口的第三个元素 op1。

[0089] S705，检测该子函数 testFun 中，完成写入或读取数组操作的变量，并根据该变量生成相应的列表接口的第四个元素 wr_tmp1，wr_tmp2，x，y。

[0090] S706，生成对应的第三中间语言描述文件，如图 6b 所示

[0091] S32，将生成的第三中间语言进行综合前端，综合生成抽象语法树。

[0092] 请参考图 8，为本实施方式中的抽象语法树 AST 的总结构，包括整个程序的根节点 program801 ；第一节点 802 的子节点存储了步骤 S1 中高级语言输入文件所描述系统的输入输出信息 ；第二节点 803 存储了步骤 S1 中高级语言输入文件中的全局变量或数组的声明信息，全局变量或数组是指在所有函数中都有效的变量或数组 ；第三节点 804 存储了步骤 S1 中高级语言输入文件中的所有函数的信息。

[0093] 请参考图 9，其中第三节点 804 是由多个节点 901 组成，节点 901 的第一子节点 902 包含了函数返回值的类型信息，第二子节点 903 包含了函数名称信息，第三子节点 904 包含了函数形参信息，第四子节点 905 和第六子节点 907 包含了这个函数语句块与其他语句块接口信息，第五子节点 906 包含了函数体语句块的信息。

[0094] S33，遍历上述抽象语法树，并按照预定的综合策略，将该抽象语法树转换为第二中间语言。

[0095] 本实施方式中的预定的综合策略包括综合后输出系统的运算速度、功率消耗、实际使用的电路面积、各类电路资源相对比例、目标可重构算子阵列的硬件结构、输入文件语言格式、电路资源使用策略。

[0096] 本实施方式中的第二中间语言的每一条语句都代表一个功能模块与此功能模块与其他模块的连接关系。

[0097] 请参考图 10，本实施方式中步骤 S33 包括步骤：

[0098] S331，读取 AST 树的每一个节点，从而得到每一个节点的信息。

[0099] S332，收集 AST 中与函数语句块相关的信息，并进行存储。

[0100] S333，根据读取的节点信息和收集的函数信息，检查 AST 生成是否正确，如是，则进行步骤 S334，否则，系统停止，抛出异常警告用户。

[0101] S334，根据读取的节点信息发射第二中间语言。

[0102] S335，输出第二中间语言。

[0103] 本实施方式中同类型的节点信息对应了固定发射了一条或几条第二中间语言或一条第二中间语言的某部分。

[0104] 请参考图 11，本实施方式中由 AST 转换为对应的第二中间语言的一种具体实施例中，AST 的节点 1301 对应“SUB8”，节点 1302 对应“port8@205”，节点 1303 对应

“port8@177”，节点 1304 对应“port8@238”，所以发射的第二中间语言的语句为：

[0105] M33 :SUBB port8@177，port8@238。

[0106] 本实施方式中由于第三中间语言的语言描述抽象层次高，类似于高级语言描述，而第二中间语言的语言描述抽象层次较低，更接近于模块化的数字电路描述，且第二中间语言的每一条语句都代表一个功能模块与此功能模块与其他模块的连接关系，因此通过将第三中间语言转换为第二中间语言来降低系统输入文件的抽象层次，以便于后面将系统输入文件转换为相应的可重构算子阵列可执行文件或者相应的硬件描述。

[0107] S34，将第二中间语言进行综合后端，生成第一中间语言，并进行仿真。本实施方式中的第二中间语言的描述抽象层次，虽然低于类似于高级语言的第三中间语言，但是不能直接转换成硬件描述语言或可重构算子阵列可执行文件。本实施方式通过将第二中间语言转换为第一中间语言，即通过降低第二中间语言的抽象层次，从而第一中间语言可以直接转换成硬件描述语言或可重构算子阵列。

[0108] 请参考图 12，本实施方式的将第二中间语言进行综合后端，生成第一中间语言包括步骤：

[0109] S341，分解第二中间语言的运算粒度到更低的层次。

[0110] S342，转换第二中间语言中功能模块到第一中间语言，且功能等效。

[0111] S343，插入对应通信单元。

[0112] 本实施方式中，由于在第二中间语言中描述的抽象层次较高，部分模块连接的通信机制在硬件上无法直接实现，因此需加入一定的通信单元。

[0113] S344，修改模块间连接信号的编号。

[0114] 本实施方式中，由于加入了新的通信单元，各个功能模块间的连接不能直接套用第二中间语言的形式，因此，需要对负责功能模块间连接的信号进行重新编号。

[0115] 请参考图 13a，为本实施方的一种实施例中，通过遍历抽象语法树，生成的第二中间语言描述文件。请参考图 13b，为对图 13a 的第二中间语言描述文件进行综合后端，得到的第三中间语言描述文件。其中，图 13a 的第二中间语言描述文件中的一条 16 位加法语句（即两个十六位数相加）:M1 :AND 16port32@1，port32@2，port32@3，经过降低其粒度，转换为了第三中间语言描述中的两条 8 位加法语句分别为 :M1 :AND port@1_1，port@2_1，port@3_1，M1 :AND port@1_2，port@2_2，port@3_2 ；其中，由于该第二中间语言描述文件中的语句 :M3 :-8CONU-16 port8@5，port6@6，不能够直接拆分，对其进行功能等效转换为第三中间语言描述的语句为 :M3 :BRD port@5_1，port@5_5，port@5_6，M3 :OR port@5_5，port@6_3，port@6_1，M3 :MOVI 127，port@6_4 ；其中，第三中间语言描述中的语句 :M4 :BRD port@9_1，port@9_3，是插入的通信单元，从而把原来第二中间语言中的 port8@9 信号广播成了 2 个信号，使 port8@9 中的数据可以被更多模块使用。请参考图 13a 和图 13b，生成的第三中间语言中的信号编号发生了一定变化。

[0116] 本实施方式中的第一中间语言中每一条语句都代表一个特定的数字电路结构及各数字电路结构间的连接关系。这个特定的数字电路结构既可以通过步骤 S4a 和步骤 S5a 下载到可重构算子阵列上实施，或者由步骤 S4b 和 S5b 后转换成具体的数字电路。

[0117] 请参考图 14 实施方式中对生成的第一中间语言进行仿真包括步骤：

[0118] S341，在仿真系统启动时配置仿真系统运行参数，包括输入待仿真的系统描述文

件、输入仿真数据文件、输出仿真数据文件、仿真系统运行时间等。

[0119] 本实施方式中的待仿真的系统描述文件为由第一语言描述文件。

[0120] S342，读取输入待仿真的系统描述文件，生成系统描述文件中描述的各子模块。

[0121] S343，将输入仿真数据文件中的数据输入到系统的数据存储容器中，等待各子模块启动，这些数据即被释放。

[0122] S344，通过读取输入待仿真的系统描述文件，连接步骤 S342 中所生成的所有子模块。

[0123] S345，启动整个仿真的运行，即启动所有子模块工作。

[0124] S346，输出仿真数据文件中所描述的输出数据到指定文件。

[0125] S4a，将该第一中间语言转换为可重构算子可执行文件，并执行步骤 S5a。

[0126] 请参考图 15，本实施方式的步骤 S4a 包括步骤：

[0127] S4a1，根据第一中间语言在可重构算子阵列上进行布局布线。

[0128] 本实施方式中进行布局布线包括步骤：

[0129] S4a11，根据该第一中间语言的每条语句中的特定数字电路结构确定其在可重构算子阵列中的实施位置。

[0130] S4a12，再根据第一中间语言的各个语句中各特定数字电路结构之间的连接关系，确定完成各特定的数字电路结构之间的连接关系所消耗的电路资源情况。

[0131] 本实施方式中的电路资源主要指硬件连线资源。

[0132] S4a2，对布局布线后的可重构算子阵列进行仿真与验证；

[0133] S4a3，将经过验证后的输出的文件转换为可重构算子可执行文件，并下载实施。

[0134] S4b，将第一中间语言转换为对应的硬件描述文件，执行步骤 S5b。

[0135] 本实施方式中的第一中间语言的每一条语句包括功能信息和连接信息。

[0136] 请参考图 16，本实施方式将第一中间语言转换为硬件描述文件包括步骤：

[0137] S4b1，转换系统从由第一中间语言描述的输入文件中主要读取两部分信息。一部分是第一中间语言中各条语句的功能信息；另一部分是第一中间语言中各条语句的连接信息，即语句间数据交换通路的信息。

[0138] S4b2，在读取到功能信息后，查找硬件描述语言模型库，得到对应功能信息的硬件描述语言描述。

[0139] S4b3，再结合连接信息，即可生成整个系统的硬件描述语言描述。

[0140] S5a，调试可重构算子阵列，并执行步骤 S6。

[0141] S5b，根据该硬件描述文件生成相应的数字电路。

[0142] S6，输出结果。

[0143] 请参考图 17a，为本实施方式的一种实施例中，由第二中间语言经过综合后端，生成的第一中间语言描述文件的节选。请参考图 17b，为将图 17a 所示的第一中间语言转换成硬件描述文件后的 verilog。其中硬件描述文件中的功能模块 alu_operator u_alu_1 和 alu_operator u_alu_2 对应第一中间语言 ADD 模块和 NOT 模块，第一中间语言中的连接关系在硬件描述文件中转化为了 reg 信号和 wire 信号（见 verilog 程序的开头部分），硬件描述文件中通过对两个模块输入不同的信号完成等同于第一中间描述的不同硬件功能的实现。

[0144] 本发明的高级综合方法通过综合生成第一中间语言，再由第一中间语言生成相应的可重构算子可执行文件或者硬件描述文件，从而输出具有多目标的特性，即既可以通过生成可重构算子阵列可执行文件把高级语言输入文件所描述的系统下载到可重构算子阵列上实施，又可以通过生成硬件描述文件把输入文件所描述的系统下载到 FPGA 或 ASIC 上实施。

[0145] 另一方面，本发明的高级综合方法，通过将高级输入文件转换为抽象层次高的第三中间语言，再由该第三中间语言转换为抽象层次较低的第二中间语言，该第二中间语言接近于模块化的数字电路描述，再由该第二中间语言转换为第一中间语言，最后由该第一中间语言直接转换为硬件描述语言或可重构算子阵列。本发明采用模块化思想的第一中间语言、第二中间语言和第三中间语言作为转换过程的中间表示形式，并着重描述具有一定功能模块的特性与模块间的连接关系和通信关系。这种高级综合过程通过模块特性、模块间的连接关系和模块间的通信机制来调度高级语言输入文件所描述的系统，从而生成的可重构算子阵列配置信息或者数字电路信息灵活、可执行性高，且执行效率高。

[0146] 以上内容是结合具体的实施方式对本发明所作的进一步详细说明，不能认定本发明的具体实施只局限于这些说明。对于本发明所属技术领域的普通技术人员来说，在不脱离本发明构思的前提下，还可以做出若干简单推演或替换，都应当视为属于本发明的保护范围。

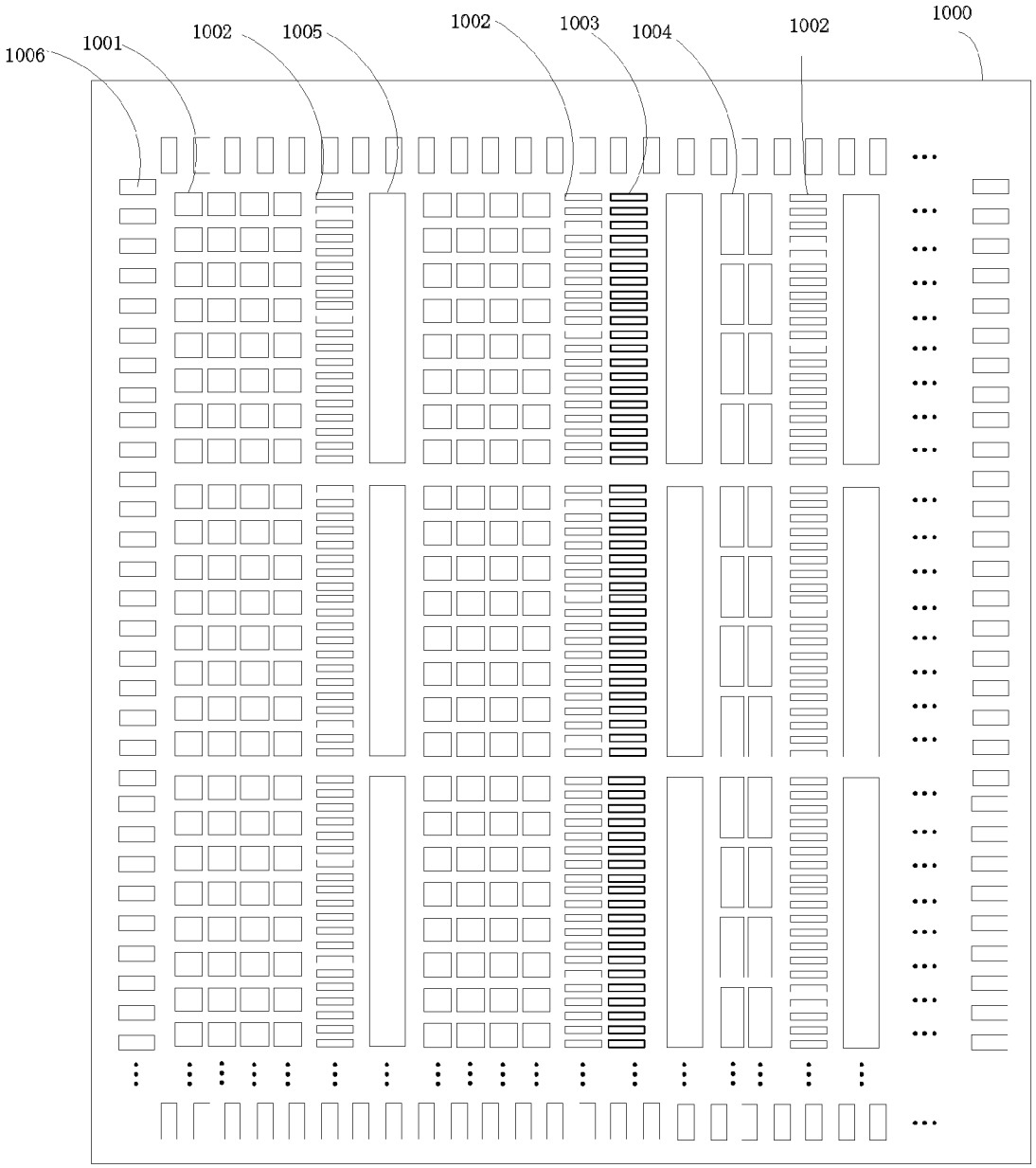


图 1

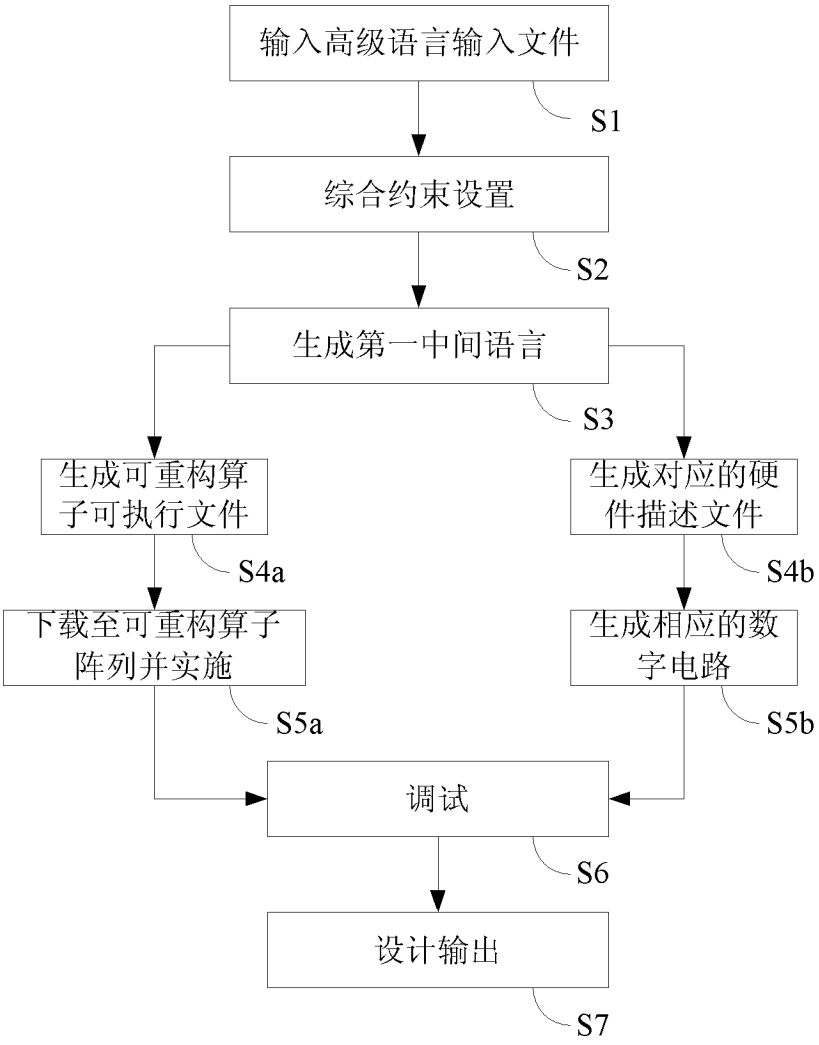

输入高级语言输入文件
S1
综合约束设置
S2
生成第一中间语言
S3
生成可重构算子可执行文件
S4a
生成对应的硬件描述文件
S4b
下载至可重构算子阵列并实施
S5a
生成相应的数字电路
S5b
调试
S6
设计输出
S7

图 2

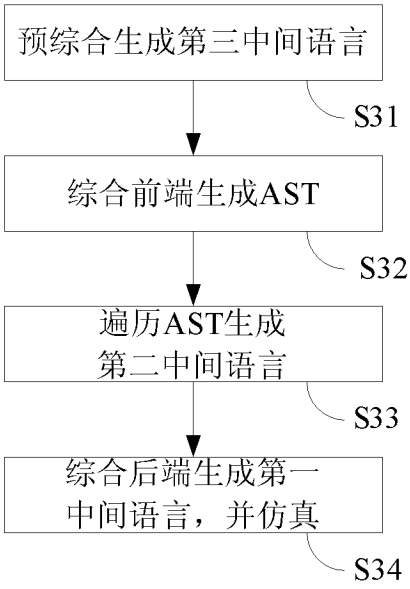

预综合生成第三中间语言
S31
综合前端生成AST
S32
遍历AST生成
第二中间语言
S33
综合后端生成第一
中间语言，并仿真
S34

图 3

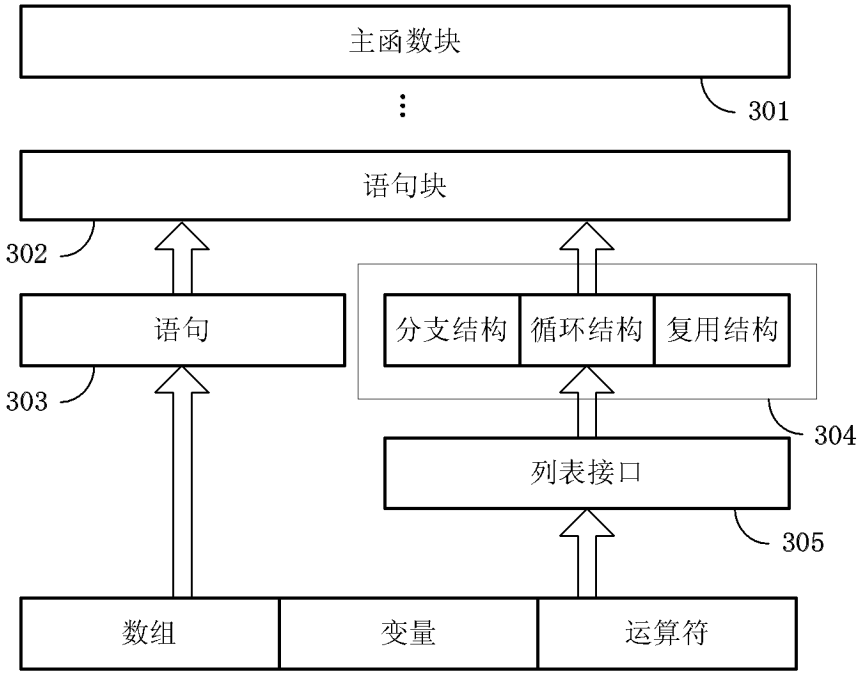

主函数块
301
语句块
302
语句
303
分支结构
循环结构
复用结构
304
列表接口
305
数组
变量
运算符

图 4

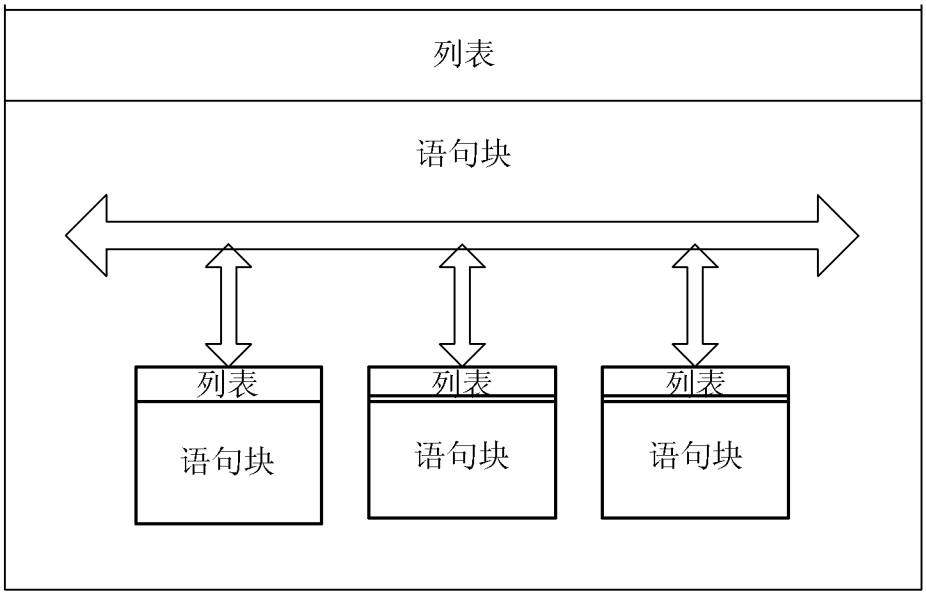


图 5

```
.....
int main() {
    op1=testFun(a,b);
    op2=testFun2(a,b,op1);
    op3=op1+op2;
}
int testFun(int x,int y) {
        int z,rd_tmp1,rd_tmp2;
        wr_tmp1=x+y;
        wr_tmp2=x-y;
        b_ar[x]=wr_tmp1;
        b_ar[y]=wr_tmp2;
        rd_tmp1=a_ar[x];
        rd_tmp2=a_ar[y];
       z=x+y+rd_tmp1-rd_tmp2;
        return z;
}
.....
```

图 6a

```
int main() {
    op1=testFun(a,b);
    op2=testFun2(a,b,op1);
    op3=op1+op2;
}
int testFun(int x,int y) #info(a_ar;b_ar;op1;wr_tmp1,wr_tmp2,x,y) {
        int z,rd_tmp1,rd_tmp2;
        wr_tmp1=x+y;
        wr_tmp2=x-y;
        b_ar[x]=wr_tmp1;
        b_ar[y]=wr_tmp2;
        rd_tmp1=a_ar[x];
        rd_tmp2=a_ar[y];
        z=x+y+rd_tmp1-rd_tmp2;
        return z;
}
```

图 6b

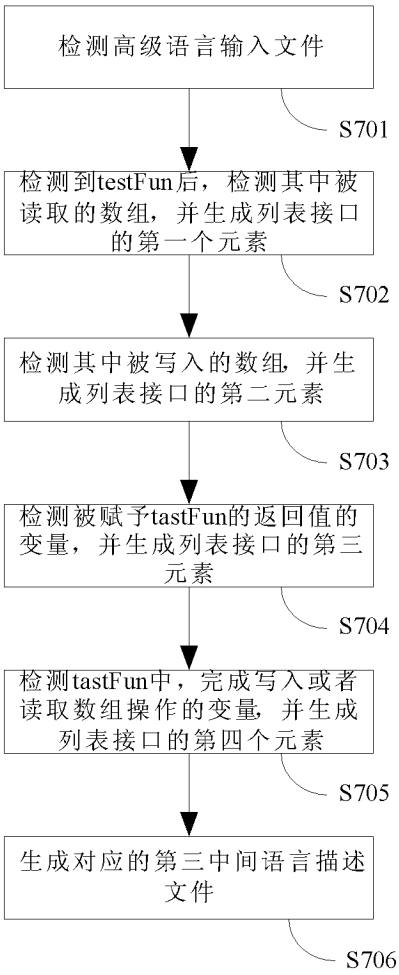

检测高级语言输入文件
S701
检测到testFun后，检测其中被读取的数组，并生成列表接口的第一个元素
S702
检测其中被写入的数组，并生成列表接口的第二元素
S703
检测被赋予tastFun的返回值的变量，并生成列表接口的第三元素
S704
检测tastFun中，完成写入或者读取数组操作的变量，并生成列表接口的第四个元素
S705
生成对应的第三中间语言描述文件
S706

图 7

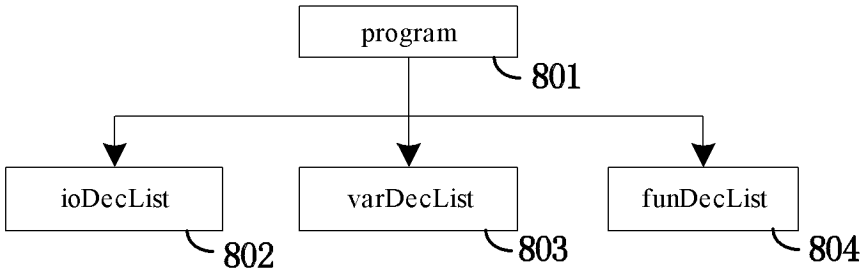

program
801
ioDecList
802
varDecList
803
funDecList
804

图 8

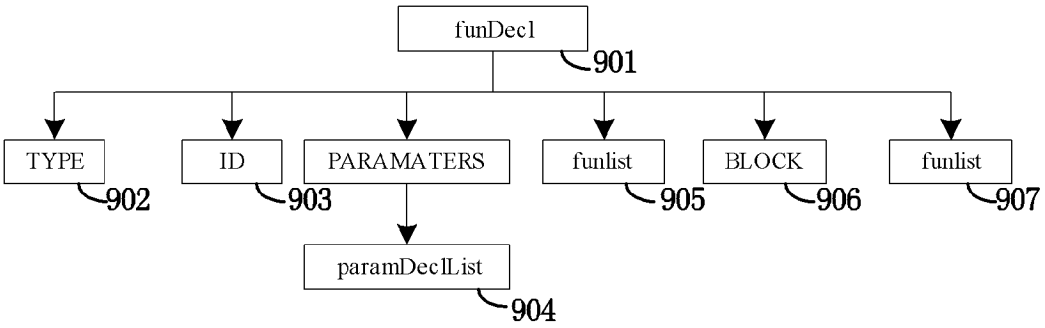

funDecl
901
TYPE
902
ID
903
PARAMATERS
paramDeclList
904
funlist
905
BLOCK
906
funlist
907

图 9

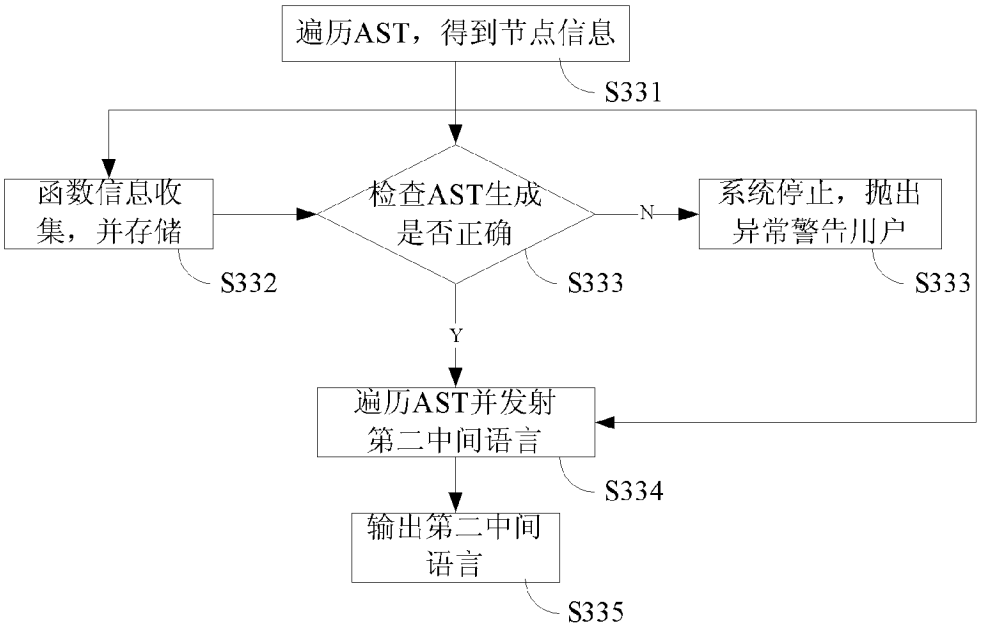

遍历AST，得到节点信息
S331
函数信息收集，并存储
S332
检查AST生成是否正确
S333
N
系统停止，抛出异常警告用户
S333
Y
遍历AST并发射第二中间语言
S334
输出第二中间语言
S335

图 10

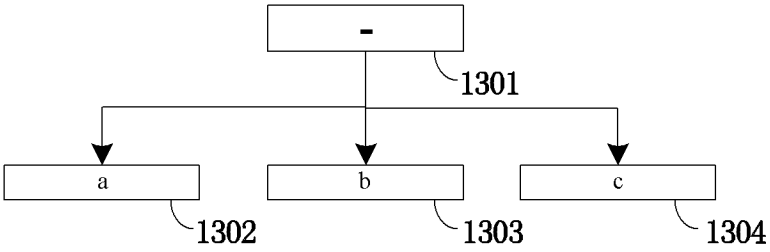

-
1301
a
1302
b
1303
c
1304

图 11

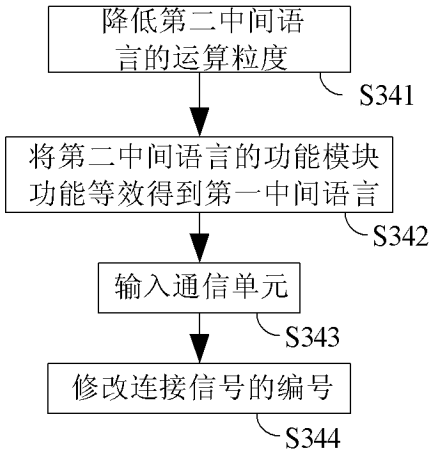


图 12

```
1 M1:  AND 16 port32@1,port32@2,port32@3
2 M2:  MOVI-16-16129, port32@8
3 M3:  -8CONU-16 port8@5, port16@6
4 M4:  BK16 port16@7, port@16@8, port@9
5 M5:  ADD16 port16@10, port16@11, port16@12
6 M6:  SUBU32 port32@13, port32@14, port32@15
7 M7:  NEG8 port8@19, port8@20
8 M8:  CLR8 port8@21, port8@22, port8@23
9 M9:  EQL 16 port16@24, port16@25, port8@26
```

图 13a

```
1 M1:  AND port@1_1,port@2_1,port@3_1
2 M1:  AND port@1_2,port@2_2,port@3_2
3 M2:  MOV1 255,  port@1_1
4 M2:  MOVI -64,,  port@4_2
5 M3:  MOVI 128,,  port@6_3
6 M3:  BRD port@5_1,port@5_5,port@5_6
7 M3:  OR port@5_5, port@6_3, port@6_1
8 M3:  MOVI 127,  port@6_4
9 M3:  OR port@5_6,port@6_4,port@6_2
10M4:  BRD port@9_1,port@9_2,port@9_3
11M4:  BK port@7_1, port@8_1, port@9_2
12M4:  BK port@7_2, port@8_2, port@9_3
13M5   :   ADDUC   port@10_1,   port@11_1,c@0_1,
port@12_1,c@12_3
14M5:  ADD C port@10_2, port@11_2,c@12_3, port@12_2,c@0_4
15M6   :   SUBUC   port@13_1,   port@14_1,c@0_1,
port@15_1,c@15_5
16M6   :   SUBUC   port@13_2,   port@14_2,c@15_5,
port@15_2,c@15_6
17M6   :   SUBUC   port@13_3,   port@14_3,c@15_6,
port@15_3,c@15_7
18M6   :   SUBUC   port@13_4,   port@14_4,c@15_7,
port@15_4,c@0_8
19M7:  MOVI   0,  port@19_2
20M7:  SUB port@19_2, port@19_1,c@0_3,port@20_1
21M8:  CLR port@21_1, port@22_1, port@23_1
22M9:  EQL port@24_1, port@25_1, port@26_2
23M9:  EQL port@24_2, port@25_2, port@26_3
24M9:  AND port@26_2, port@26_3, port@26_1
```

图 13b

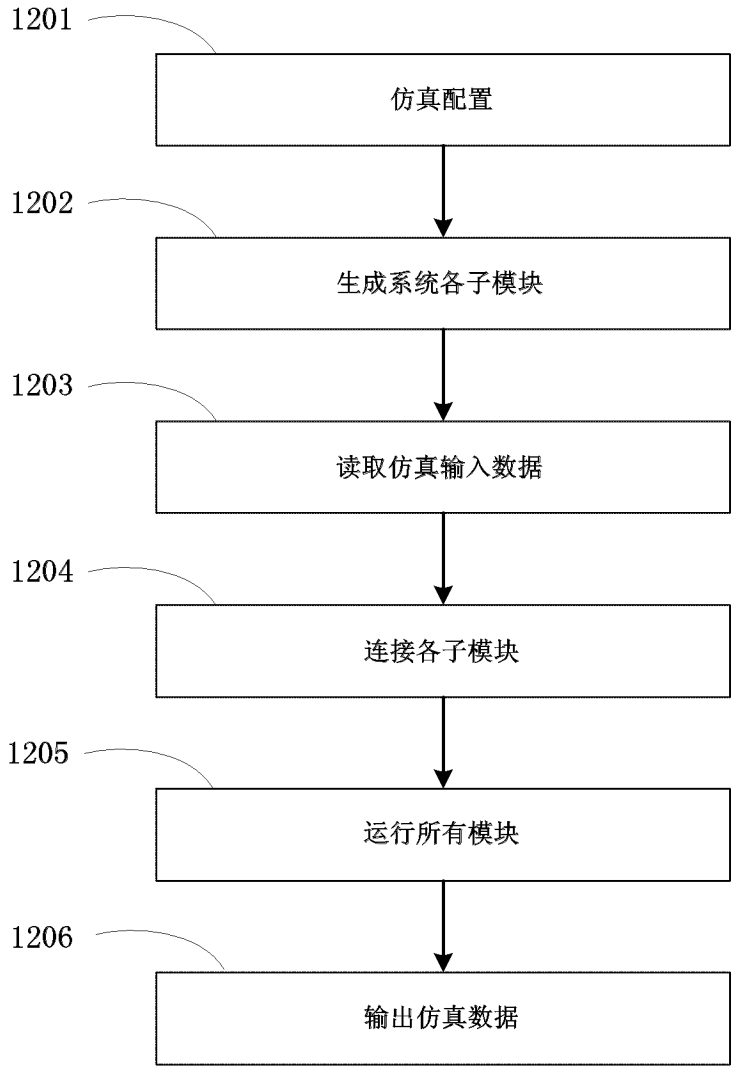

1201
仿真配置
1202
生成系统各子模块
1203
读取仿真输入数据
1204
连接各子模块
1205
运行所有模块
1206
输出仿真数据

图 14

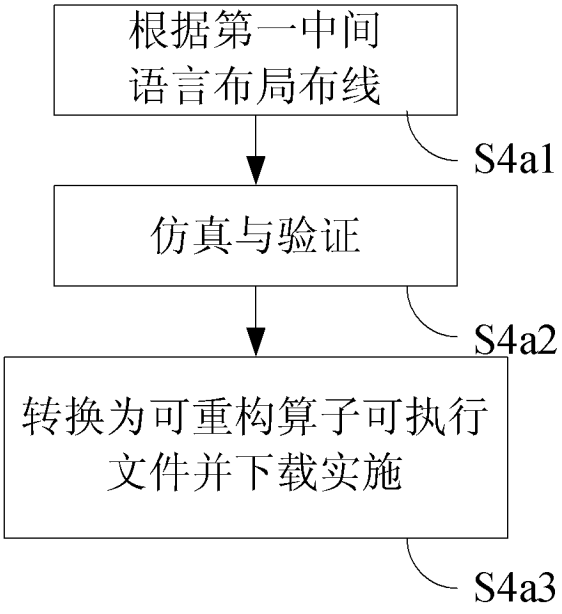

根据第一中间
语言布局布线
S4a1
仿真与验证
S4a2
转换为可重构算子可执行
文件并下载实施
S4a3

图 15

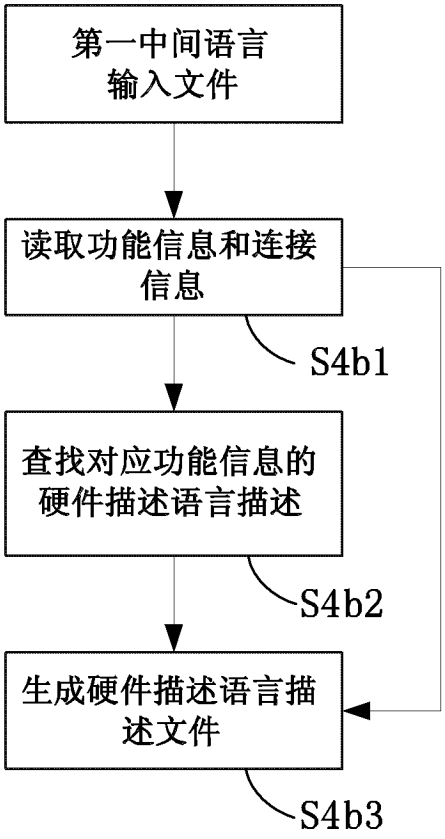

第一中间语言
输入文件
读取功能信息和连接
信息
S4b1
查找对应功能信息的
硬件描述语言描述
S4b2
生成硬件描述语言描
述文件
S4b3

图 16

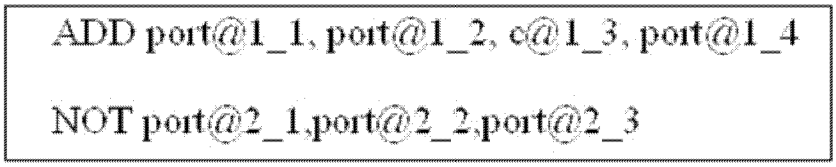


图 17a

```
module top;
reg clk ;
reg config_clk ;
reg rst_n;
reg opcode_alu_1;
reg opcode_alu_2;
wire [7:0] port1_1,port1_2;
wire valid_port1_1,valid_port1_2 ;
wire ack_port1_1,ack_port1_2 ;
wire c1_3 ;
wire valid_c1_3 ;
wire ack_c1_3;
wire [7 :0] port1_4 ;
wire [7:0] port2_1,port2_2;
wire valid_port2_1,valid_port2_2 ;
wire ack_port1_1,ack_port1_2 ;
wire [7 :0] port2_3 ;
wire valid_port2_3 ;
wire ack_port2_3 ;
alu_operator u_alu_1(
                        .clk(clk),
                        .config_clk(config_clk),
                        .rst_n(rst_n),
                        .opcode (opcode_alu_1),
                        .src1(port1_1),
                        .src2(port1_2),
                        .carry_in(c1_3),
                        .dst(port1_4),
                        .valid_src1(valid_port1_1),
.valid_src2(valid_port1_2),
.valid_carry_in(valid_c1_3),
.valid_dst(valid_port1_4),
.ack_src1(ack_port1_1),
.ack_src2(ack_port1_2),
.ack_carry_in(ack_c1_3),
.ack_dst(ack_port1_4)
);
alu_operator u_alu_2(
                        .clk(clk),
                        .config_clk(config_clk),
                        .rst_n(rst_n),
                        .opcode (opcode_alu_2),
                        .src1(port2_1),
                        .src2(port2_2),
                        .dst(port2_3),
                        .valid_src1(valid_port2_1),
.valid_src2(valid_port2_2),
.valid_dst(valid_port2_3),
.ack_src1(ack_port2_1),
.ack_src2(ack_port2_2),
.ack_dst(ack_port2_3)
);
endmodule
```

图 17b