Создание методики и алгоритма генерации виртуального аппаратного обеспечения по реальным характеристикам.

Диссертация на соискание степени магистра по направлению 09.04.04«Программная инженерия»

Научный руководитель: канд. техн. наук, доц. А. И. Кононова
Соискатель: магистрант. гр. ПИН-22М Александр Александрович Уманский

Объект исследования:

Существующие методики создания виртуального аппаратного обеспечения.

Предмет исследования:

Программное обеспечение.

Проблемная ситуация

Высокая трудоемкость и дороговизна создания прикладного ПО, ориентированного на использование конкретного физического устройства.

Причины сложившейся ситуации:

- невозможность обеспечить всех разработчиков прикладного ПО физическими устройствами;
- отсутствие автоматического или полуавтоматического ПО для эмуляции физических устройств;
- неизбежное появление программных ошибок в "рукописном"устройстве;

Цели и задачи диссертации

Цель: создание методики и алгоритма генерации виртуального аппаратного обеспечения на основе его характеристик.
Задачи:

- аналитический обзор существующих методов создания виртуального аппаратного обеспечения;
- анализ существующих подходов к эмуляции аппаратного обеспечения;
- анализ актуальности существующих подходов к разработке прикладного ПО при отсутствии аппаратного обеспечения;
- декомпозиция поставленной задачи для создания методики и алгоритма генерации виртуального аппаратного обеспечения;
- разработка лингвистического аппарата (семантика, синтаксис) языка для создания программ по генерации виртуального аппаратного обеспечения;

Положения, выносимые на защиту

1. Формализованное представление алгоритма генерации виртуального аппаратного обеспечения;
2. Алгоритма генерации виртуального аппаратного обеспечения;
3. Программная реализация разработанных методики и алгоритма;
4. Лингвистический аппарат (синтаксис, семантика) языка для создания программ по генерации виртуального аппаратного обеспечения;

Анализ существующих средств макетирования реальных устройств

Свойства \ Название программы	umockdev	Verilator	GHDL
Кроссплатформенность	Да	Да	Да
Открытость исходного кода	Да	Да	Да
Бесплатность	Да	Да	Да
Анализ языков описания аппаратуры	Нет	Да	Да
Создание виртуального устройства в системе	Есть воспроизведение поведения	Нет	Нет
Графический интерфейс	Нет	Нет	Нет

Формализованное представление

Описание устройства это абстракное синтаксическое дерево T на разрабатываемом языке L.
$P\colon T \to F(L)$ – отображение, определяющее преобразование языка L в язык C.

Методика создания полнофункционального виртуального устройства

Из описания аппаратуры:

1. С помощью Verilator или GDHL генерируется модель устройства;
2. Из модели удаляется все, кроме общения устройства с внешним миром и обработки этой информации;
3. Полученная модель встраивается как новое устройство в эмулятор QEMU;

Полуавтоматически:

1. С помощью разработанного языка описываются интерфейсы и характеристики устройства;
2. Компилятор для разработанного языка преобразует данное описание в код на языке C;
3. Пользователь заполняет в полученном файле функции обработки данных;
4. Данный файл с описанием устройства на языке C встраивается как новое устройство в эмулятор QEMU;

Программная реализация

```
#include "qemu/osdep.h"
#include "qapi/error.h"
#include "qom/object.h"
/*[[[cog
    import json as j
    import main as m

    with open(m.DEV_SCHEMA_FILE, 'r') as sf:
        SCHEMA = j.load(sf)

    device_header = SCHEMA["parent"]["header"]
    cog.outl(f'#include "{device_header}"')
   ]]]*/
/*[[[end]]]*/
#include "exec/address-spaces.h"
#include "hw/qdev-properties-system.h"

#include <Python.h>

#define IF_NULL_GOTO_ERR(VAR, BODY) \
    BODY; \
    if (!VAR){ \
        goto err; \
    }

#define PY_COMPILE_AND_GET_FUNC(FUNC_NAME, PY_CODE, PY_COMPILED, PY_MODULE, PY_FUNC) \
    if (!PY_COMPILED){ \
        IF_NULL_GOTO_ERR(PY_COMPILED, \
                         PY_COMPILED = Py_CompileString(PY_CODE, FUNC_NAME ".py", Py_single_input)) \
    } \
    IF_NULL_GOTO_ERR(PY_MODULE, \
                     PY_MODULE = PyImport_ExecCodeModule(FUNC_NAME "module", PY_COMPILED)) \
    IF_NULL_GOTO_ERR(PY_FUNC, \
                     PY_FUNC = PyObject_GetAttrString(PY_MODULE, FUNC_NAME))
```

Выбор метрики оценки эффективности

Основные метрики эффективности:

1. Время разработки виртуального устройства (в человеко-часах)
2. Качество сгенерированного кода на языке C

Основные результаты диссертационной работы

1. Проведен аналитический обзор существующих методов создания виртуального аппаратного обеспечения;
2. Проведен анализ существующих подходов к эмуляции аппаратного обеспечения;
3. Проведен анализ актуальности существующих подходов к разработке прикладного ПО при отсутствии аппаратного обеспечения;
4. Произведена декомпозиция поставленной задачи для создания методики и алгоритма генерации виртуального аппаратного обеспечения;
5. Разработан лингвистический аппарата (семантика, синтаксис) языка для создания программ по генерации виртуального аппаратного обеспечения;
6. Разработан лингвистический аппарата (семантика, синтаксис) языка для создания программ по генерации виртуального аппаратного обеспечения;

Спасибо за внимание!

Ссылки I