

# МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное  
учреждение высшего образования «Национальный  
исследовательский университет  
«Московский институт электронной техники»

## **Исследование и разработка методики и алгоритма генерации виртуального аппаратного обеспечения по спецификации**

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

**Научный руководитель:** докт. ф-м. наук, доц. Кононова А.И.  
**Соискатель:** магистрант. гр. ПИН-22М Уманский А.А.

Москва, 2022

## Проблемная ситуация

При создании прикладного ПО для специализированного аппаратного обеспечения дорого обеспечивать разработчиков самим аппаратным обеспечением.

### **Причины сложившейся ситуации:**

- ▶ производство экземпляров аппаратного обеспечения в условиях санкций и дефицита полупроводников стала дорогой;
- ▶ простаивание программистов, пока происходит производство и доставка аппаратного обеспечения;
- ▶ трудоемкость создания собственного виртуального аппаратного обеспечения.

# Пример специализированного аппаратного обеспечения

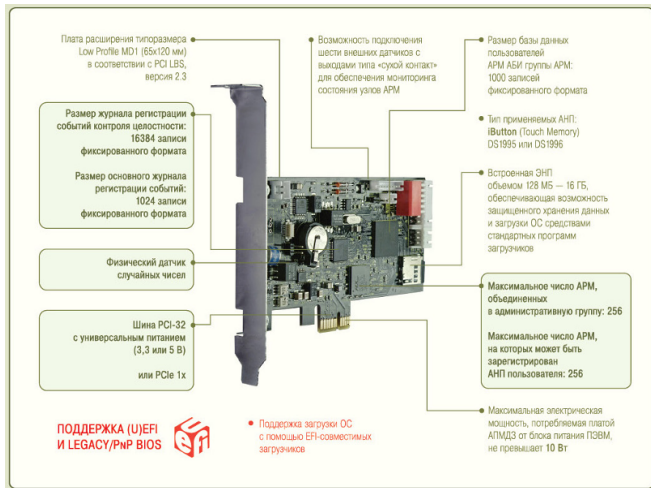


Рис. 1: Аппаратно-программный модуль доверенной загрузки Максим-M1

## Цель и задачи диссертации

**Цель:** снижение трудоемкости создания виртуальных устройств.

**Задачи:**

- ▶ аналитический обзор существующих методов создания виртуального аппаратного обеспечения;
- ▶ формализация задачи создания виртуального аппаратного обеспечения;
- ▶ создание методики и алгоритма генерации виртуального аппаратного обеспечения на основе его спецификации;
- ▶ разработка лингвистического аппарата (семантика, синтаксис) языка для создания программ по генерации виртуального аппаратного обеспечения;
- ▶ выбор метрики оценки эффективности;
- ▶ оценка эффективности.

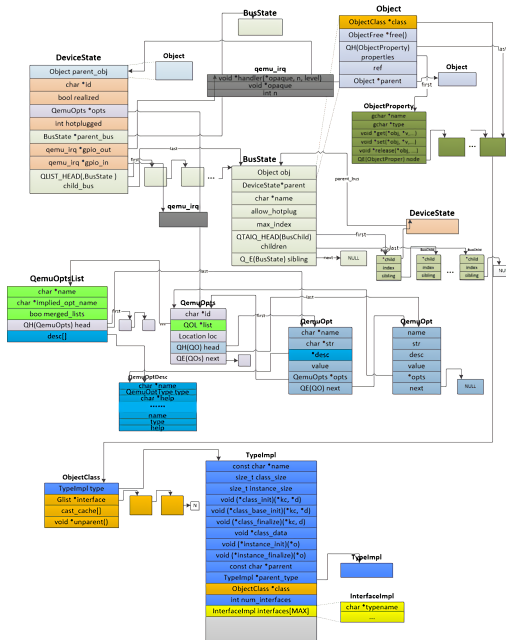
## На защиту выносятся

- ▶ формализованное представление задачи создания виртуального аппаратного обеспечения;
- ▶ формализованное представление алгоритма и методики генерации виртуального аппаратного обеспечения;
- ▶ лингвистический аппарат (синтаксис, семантика) языка для создания программ по генерации виртуального аппаратного обеспечения;
- ▶ метрики оценки эффективности;
- ▶ экспериментальные результаты применения генератора аппаратного обеспечения.

# Анализ существующих методов создания виртуального аппаратного обеспечения

Метод	Особенности	Недостатки
Создание stub-симулятора	Требует создания интерфейсов-адапторов в прикладном ПО	Приходится создавать интерфейсы-адапторы для каждого разрабатываемого ПО
Использование записи работы АО	Быстрый метод, не требует специальных знаний о внутреннем устройстве АО	<ul style="list-style-type: none"><li>● Взаимодействие ПО с аппаратным обеспечением ограничивается заранее записанными сценариями</li><li>● Количество записей очень быстро разрастается</li><li>● Зачастую записи снимаются только с корректных сценариев использования</li></ul>
Использование эмулятора QEMU	<ul style="list-style-type: none"><li>● Готовая инфраструктура для создания виртуального АО</li><li>● Постоянная поддержка эмулятора силами сообщества</li></ul>	<ul style="list-style-type: none"><li>● Необходимость написания виртуального АО на низкоуровневом языке</li><li>● Необходимость обучения объектной системе QEMU (QOM)</li></ul>

# Объектная модель QEMU (QOM)



## Формализация задачи создания виртуального аппаратного обеспечения

Время разработки виртуального аппаратного обеспечения можно выразить формулой

$$T = L + D + C + R \quad (1)$$

- ▶  $T$  – общее время разработки виртуального аппаратного обеспечения;
- ▶  $L$  – время анализа QOM для реализации виртуального аппаратного обеспечения;
- ▶  $D$  – описание устройства в терминах QOM;
- ▶  $C$  – программирование логики устройства;
- ▶  $R$  – тестирование и отладка.

Порог вхождения в QOM для программиста высок, из-за чего  $L + D > C + R$ . В данном исследовании стоит задача уменьшения  $L$  и  $D$ .



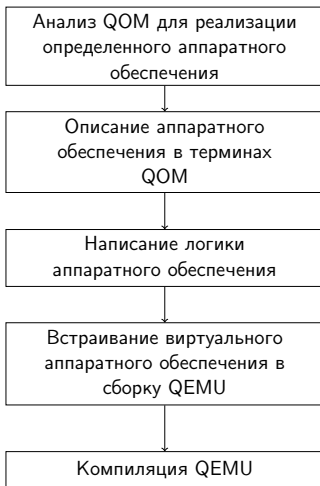
## Формализация задачи создания виртуального аппаратного обеспечения

Пусть задан ориентированный взвешенный граф  $G = (V, E)$ , где  $V$  – коммиты в системе контроля версий, а  $E$  – время, затраченное программистом на формирование коммита. Общее время, затраченное на формирование цепочки коммитов рассчитывается по формуле:

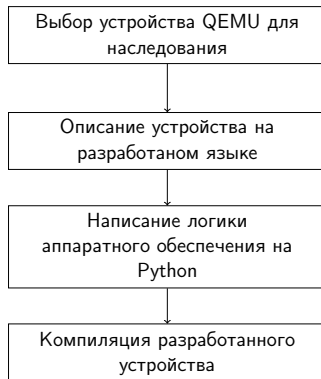
$$\sum_{i=1}^n f(e_{i,i+1}) \quad (2)$$

Где  $f$  – весовая функция, отображающая ребра в их веса:  
 $f : E \rightarrow \mathbb{R}$

# Методика создания виртуального аппаратного обеспечения

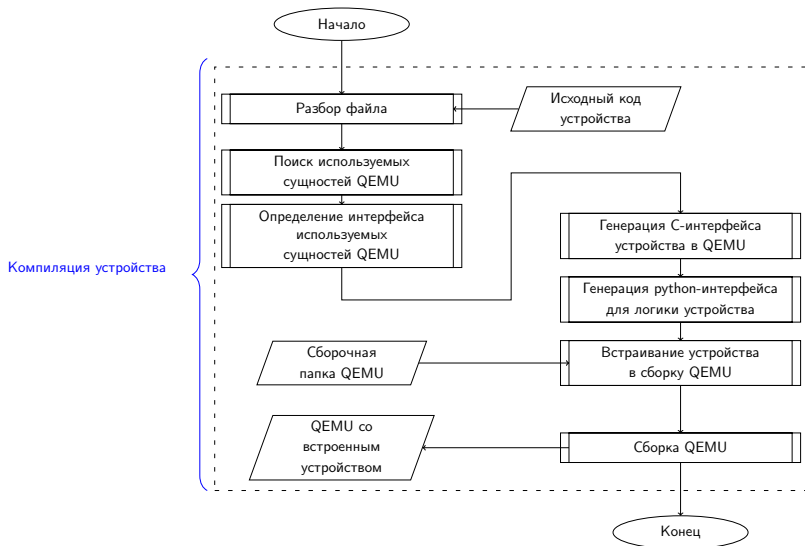


Классическая методика



Разработанная методика

# Алгоритм создания виртуального аппаратного обеспечения



## Грамматика языка QPyDev

$\langle letter \rangle$	::= 'a' ... 'z'   'A' ... 'Z';
$\langle digit \rangle$	::= '0' ... '9' ;
$\langle symbol \rangle$	::= \x20 ... \x7E ; (* любой печатный символ, согласно кодам ASCII *)
$\langle const\ value \rangle$	::= <digit>   ' ' { <symbol> } ' ' ;
$\langle identifier \rangle$	::= <letter> [{ <letter>   <digit>   ' _ ' } ] ;
$\langle block\ start \rangle$	::= " ;
$\langle block\ end \rangle$	::= " ;
$\langle field \rangle$	::= <identifier> '=' <identifier>   <block> ;
$\langle block \rangle$	::= <block start> <field> [{ ' ' <field> } ] <block end>;
$\langle device\ definition \rangle$	::= '# ' <identifier>;
$\langle device\ class\ inheritance \rangle$	::= '(' <identifier> ':' <identifier> [{ ' ' <identifier> } ] ')';
$\langle device\ class\ block \rangle$	::= <device class inheritance> <block>;
$\langle bind\ block \rangle$	::= '@bind' <block>;
$\langle python\ block \rangle$	::= '@py' <block>;
$\langle program \rangle$	::= <device definition> <device class block> <bind block> <python block>;

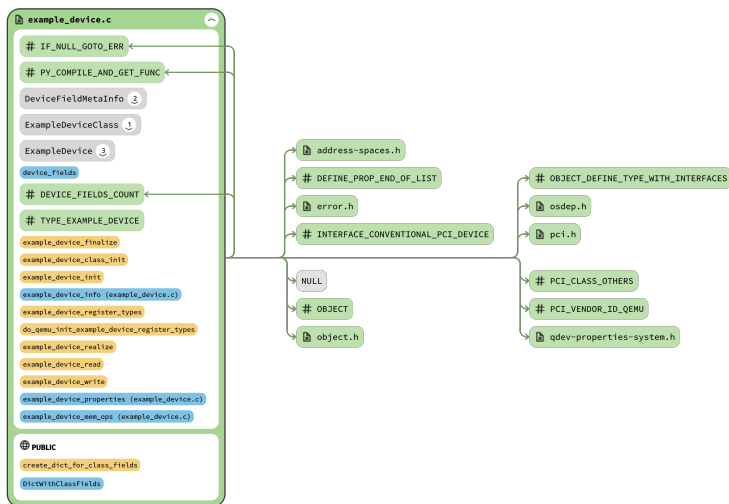
# Денотационная семантика языка QPyDev I

Математическое описание	Значение
$[[assignment]](x,y) = \lambda x.y$	Операция присваивания значения $y$ переменной $x$
$[[terminate]](m) =$ Завершение работы компилятора	Терминирование компилятора с сообщением $m$
$[[if]](c,e_1,e_2) = \begin{cases} e_1, & \text{Если } c = true \\ e_2, & \text{Если } c \neq true \end{cases}$	Условное исполнение. Если условие $c$ истинно, то выполняется $e_1$ , иначе $e_2$
$[[throw\ error]](c,e) = if(c,e_g,terminate)$	Создание и бросание исключения при ложном условии $c$
$[[lookup]](o) = [[throw\ error]](o \in Q, o)$	Поиск объекта $o$ в множестве объектов QEMU $Q$ . В случае, если объект не найден, генерируется исключение
$[[< device\ definition >]](i) = lookup(i)$	Поиск указанного класса устройства в объектах QEMU

# Денотационная семантика языка QPyDev II

$[[< device\ class\ inheritance >]](i_1, \dots, i_n) = lookup(i_1) \wedge \dots \wedge lookup(i_n)$	<p>Поиск указанного класса для наследования и интерфейсов в объектах QEMU. Для успешного завершения должны быть найдены все объекты</p>
$[[< field >]](v_1, v_2) = [[throw\ error]](v_1 \in Q \wedge v_2 \in C \cup Q, assignment(v_1, v_2))$	<p>Присваивание полям значений при условии, что <math>v_1</math> принадлежит множеству объектов QEMU, а <math>v_2</math> множеству констант или множеству объектов QEMU</p>
$[[< block >]](f_1, \dots, f_n) = field(f_1) \wedge \dots \wedge field(f_n)$	<p>Присваивание связанных с одной сущностью полей</p>
$[[< pythonblock >]](b) = assignment(B, B)$	<p>Инициализация специального поля с Python-логикой</p>

# Программная реализация виртуального аппаратного обеспечения с помощью QPyDev



## Выбор метрики оценки эффективности

Основные метрики эффективности разработанного языка:

- ▶ время разработки виртуального аппаратного обеспечения (в человеко-часах);
- ▶ быстродействие сгенерированного виртуального аппаратного обеспечения.

Экспериментальное устройство выполняет задачу сжатия JPEG-картинки. Данная задача легко поддается измерению, так как:

- ▶ легко выбрать сложность входных данных – это размер изображения;
- ▶ возможна векторизация этапов алгоритма;
- ▶ возможно добавить разные подходы к обработке изображения:
  - ▶ вызов подпрограммы;
  - ▶ отправка данных по сети;
  - ▶ реализация алгоритма устройства.



## Оценка эффективности

Таблица 3: Сравнение эффективности разработки и производительности виртуальных устройств реализующих алгоритм сжатия JPEG картинки

Метрика	Разработка с нуля		Использование библиотеки	
	С устройство	Python устройство	С устройство	Python устройство
Время разработки в человеко-часах ( $T$ )	100	50	35	10
Время сжатия (сек.)	3.9	18.5	1.9	2.8

▶  $L = 20$

▶  $D = 4$

▶  $C = 65$

▶  $R = 11$

▶  $L = 4$

▶  $D = 1$

▶  $C = 38$

▶  $R = 7$

▶  $L = 20$

▶  $D = 4$

▶  $C = 8$

▶  $R = 3$

▶  $L = 4$

▶  $D = 1$

▶  $C = 3$

▶  $R = 2$

## Основные результаты диссертационной работы

- ▶ проведен аналитический обзор существующих методов создания виртуального аппаратного обеспечения;
- ▶ формализована задача создания виртуального аппаратного обеспечения;
- ▶ созданы методика и алгоритм генерации виртуального аппаратного обеспечения на основе его спецификации;
- ▶ разработан лингвистический аппарат (семантика, синтаксис) языка для создания программ по генерации виртуального аппаратного обеспечения;
- ▶ проведены эксперименты, результатом которых явилось сокращение времени разработки виртуального аппаратного обеспечения в 2 раза по сравнению с классическим подходом, тогда как производительность устройства упала всего в 1.5 раза.

Спасибо за внимание!