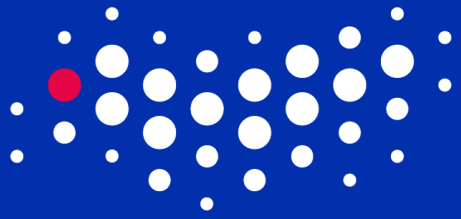


УНИВЕРСИТЕТ ИТМО

Курс «Системы на кристалле»

Лекция 5
Методология HW/SW Codesign

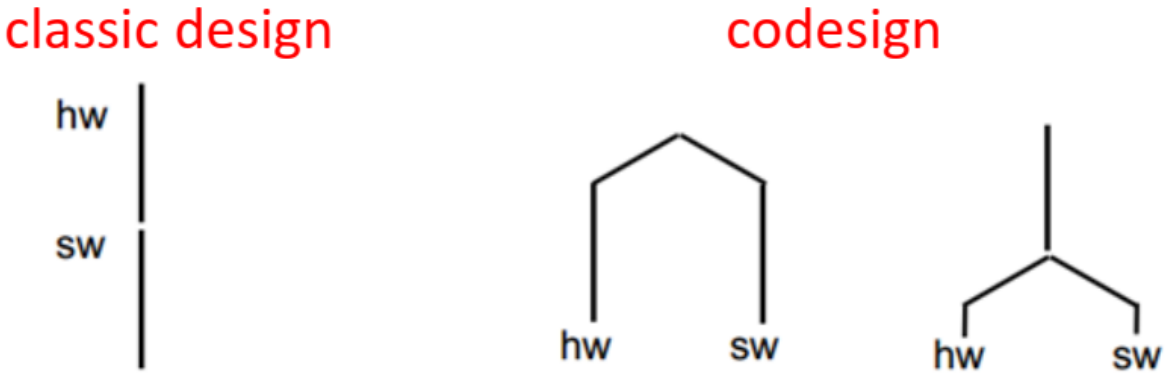
Быковский С.В

e-mail: sergei_bykovskii@itmo.ru

Санкт-Петербург, 2019

Введение

- **HW/SW Codesign** – процесс параллельного и скоординированного проектирования электронных программно-аппаратных систем, который основывается на НЕ зависящем от конечной реализации описании и использует средства автоматизации проектирования.
- При **HW/SW Codesign** используется синергизм аппаратуры и ПО с целью оптимизации и/или удовлетворения проектных ограничений, таких как стоимость, производительность, энергопотребление и время выхода на рынок целевого продукта.



Рекомендуемая литература

- Jurgen Teich. «**Hardware/Software Codesign: The Past, the Present, and Predicting the Future**». 2012.
- Курс лекций Швейцарской высшей технической школы Цюриха по дисциплине **HW/SW Codesign** https://tec.ee.ethz.ch/education/lectures/hardware-software-codesign.html
- Быковский С.В., Горбачев Я.Г., Ключев А.О., Пенской А.В., Платунов А.Е. **Сопряжённое проектирование встраиваемых систем (Hardware/Software Co-Design).** **Часть 2**: Учебное пособие - Санкт-Петербург: Университет ИТМО, 2016. - 105 с. - экз

Цели методологии

Главные цели и намерения методологии HW/SW **Co**design объясняются, используя различные интерпретации слога «**Co**» слова **Co**design:

- **Co**-ordination
- **Co**-ncurrency
- **Co**-rrectness
- **Co**-mplexity

Co-ordination

Координация этапов проектирования междисциплинарных проектных групп:

- Разработчики приложений (application)
- Разработчики операционной системы (middleware)
- Разработчики встроенного системного ПО (firmware)
- Разработчики интегральных схем (integrated chips)
- Разработчики аппаратного обеспечения (hardware)

Co-ncurrency

Одновременная (~~параллельная~~) работа аппаратных и программных разработчиков позволяет избежать узкого места, свойственного классическому потоку проектирования.

В классическом потоке проектирования разработчики ПО ожидают готовности аппаратной платформы.

Codesign позволяет избежать это узкое место с помощью ***ко*симуляции** и ***виртуального прототипирования***.

Co-rrectness

- Верификация не только каждой отдельной подсистемы, но и **ко**верификация работы этих подсистем в составе всей системы.
- Сковозная верификация проекта СнК на каждом уровне и каждом этапе
- Отслеживание зависимости изменений на разных уровнях проектирования - ECO (Engineering change orders) - и проверка соответствия результатов.

Co-mplexity

Методы Codesign’а должны способствовать проектированию сложных **оптимизированных** современных электронных систем, в частности СнК

Типичные оптимизируемые характеристики

- Стоимость
- Энергопотребление
- Производительность

Первое поколение (до 1995 года)

- Синтез из С-подобной функциональной спецификации системы такого вида: ЦП <-> общая шина <-> Ускоритель
- Выделяют 2 отправных точки для синтеза:
  - полностью аппаратная реализация
  - полностью программная реализация
- ЦП простаивает, пока работает ускоритель и наоборот
- Однопроцессорность, однопоточность

Второе поколение (до 2004 года)

- Мультипроцессорность
- Мультипроцессность
- Косимуляция. Пример: одновременная симуляция аппаратуры на уровне регистровых передач (RTL) и программного кода на виртуальной модели ЦП для ускорения процесса

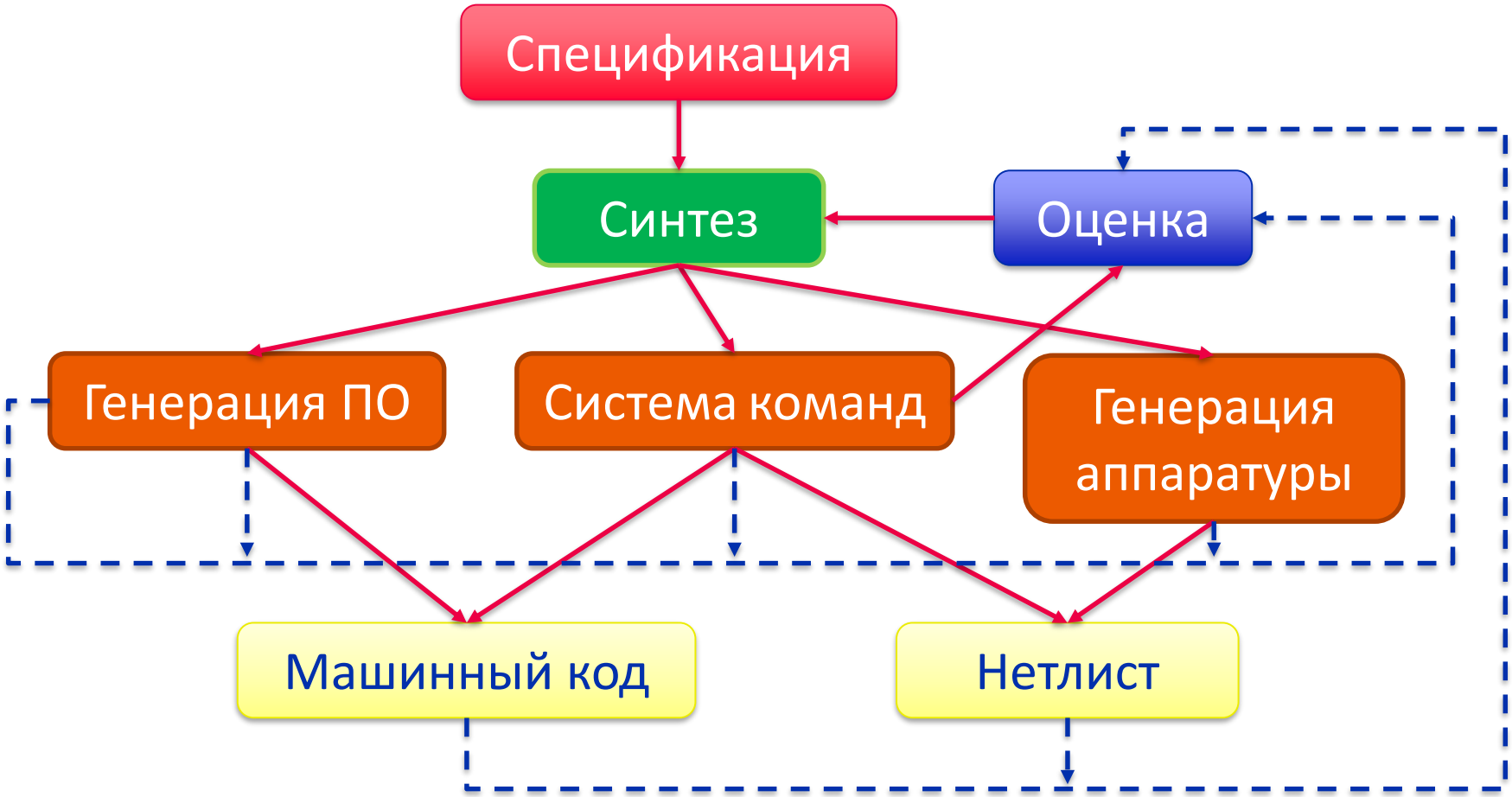
Третье поколение (до 2015 года)

- Целевая платформа становится гетерогенной (может включать различные виды вычислителей: RISC, DSP, ASIP, VLIW, и т.д.), появляются NoC
- Актуальная задача – полностью автоматизированный **межуровневый** синтез.

Четвертое поколение (настоящее время)

- Тесная интеграция аналоговой и цифровой части в СнК.
- Повсеместное внедрение кибер-физических систем.
- Совместное проектирование инструментальной и системной части.
- Перемещение дизайна в runtime фазу. Развитие концепции реконфигурируемых и обучаемых систем.

Типовая схема процессов в Codesign



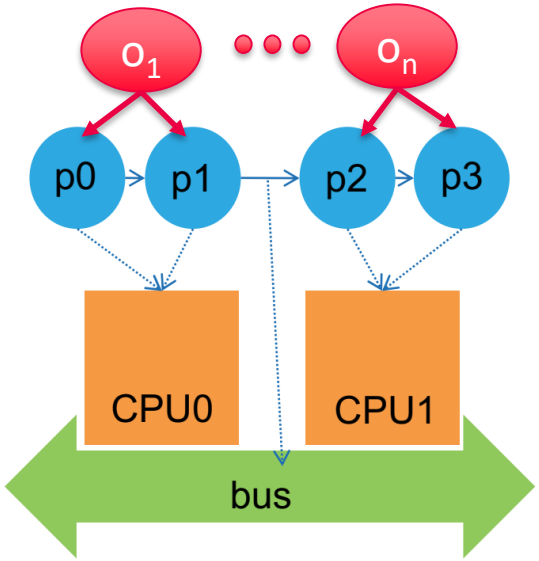
Задачи синтеза системы

- **Allocation** – выбор ресурсов, включающих процессоры, аппаратные IP и их межсоединения. Отсутствующие в библиотеках IP могут синтезироваться.
- **Binding** – связывание функциональности (задач, процессов, функций) с выбранными ресурсами.
- **Scheduling** – диспетчеризация ресурсов между процессами (память, процессорное время и т.п.)
- **Mapping** = binding + scheduling
- **Partitioning** = allocation + binding

Формальное определение задачи разделения (partitioning)

Разбить n объектов O={o1,…,on} на части (партиции) P={p1,…,pm} и привязать их к m блокам, так чтобы:

- p1 V p2 V…V pm = O (все объекты были привязаны)
- pi ∩ pj = { } $\forall i, j: i \neq j$ (объект был привязан один раз)
- Стоимость c(P) была минимизирована



Методы разделения

Точные методы

- Перебор
- Целочисленное линейное программирование

Эвристические методы

- Случайное отображение
- Иерархическая кластеризация

Итеративные методы

- Эволюционные алгоритмы
- Алгоритм Кернигана-Лина
- Алгоритм имитации отжига

Задача целочисленного программирования

Задается цель ⇒ $C = \sum_{x_i \in X} a_i x_i \text{ with } a_i \in \mathbb{R}\ x_i \in \mathbb{N}$

и ограничения ⇒ $\forall j \in J : \sum_{x_i \in X} b_{i,j}\, x_i \geq c_j \text{ with } b_{i,j}, c_j \in \mathbb{R}$

Необходимо минимизировать целевую функцию с учетом ограничений

Пример постановки задачи целочисленного программирования (1)

Имеется 8 задач $\{v_1 \dots v_8\}$ и два процессора p_1, p_2

Каждая задача v_i характеризуется необходимыми ресурсами памяти $c^m_{i,1}$ и $c^m_{i,2}$, а также энергопотребления $c^e_{i,1}$ и $c^e_{i,2}$

на первом и втором процессоре соответственно.

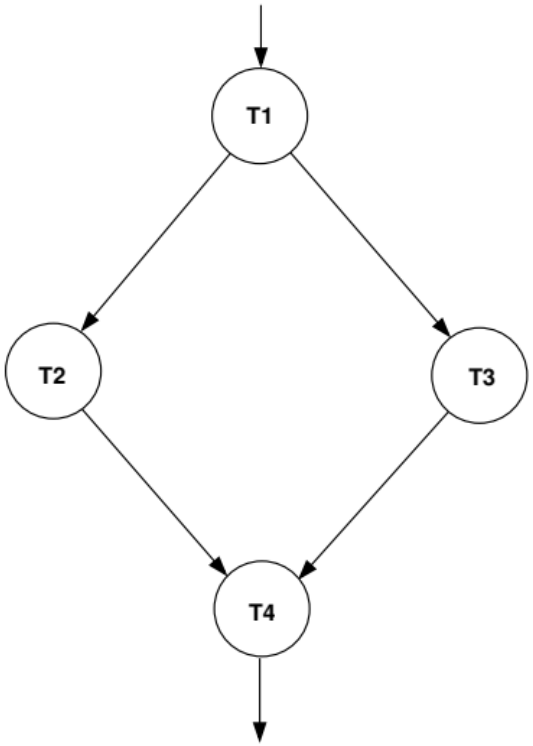
Пример постановки задачи целочисленного программирования (2)

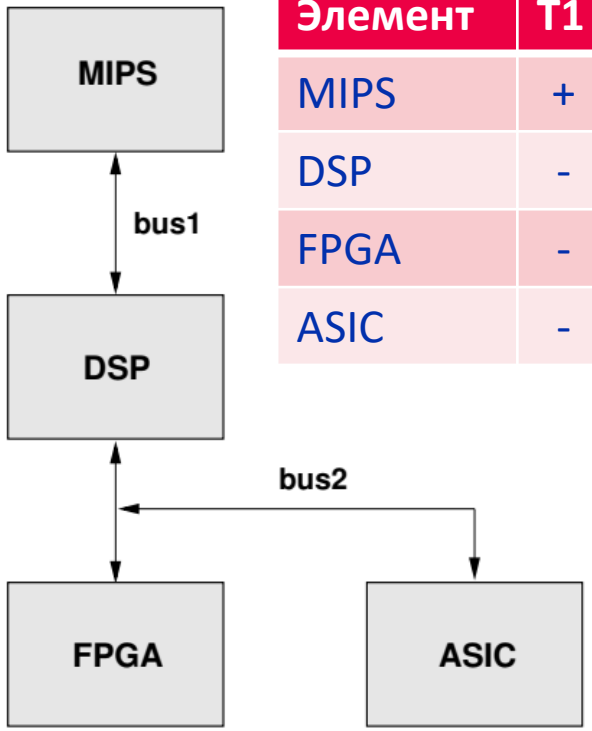
$$min\left\{\sum_{i=1}^{8}\sum_{j=1}^{2} c^{e}_{i,j}\cdot X_{i,j} + \sum_{i=1}^{8}\sum_{j=1}^{2} c^{m}_{i,j}\cdot X_{i,j}\right\}$$

$$\sum_{j=1}^{2} X_{i,j} = 1, i = 1..8$$

$$X_{i,j} \in \{0,1\}$$

Разделение с использованием графа (1)

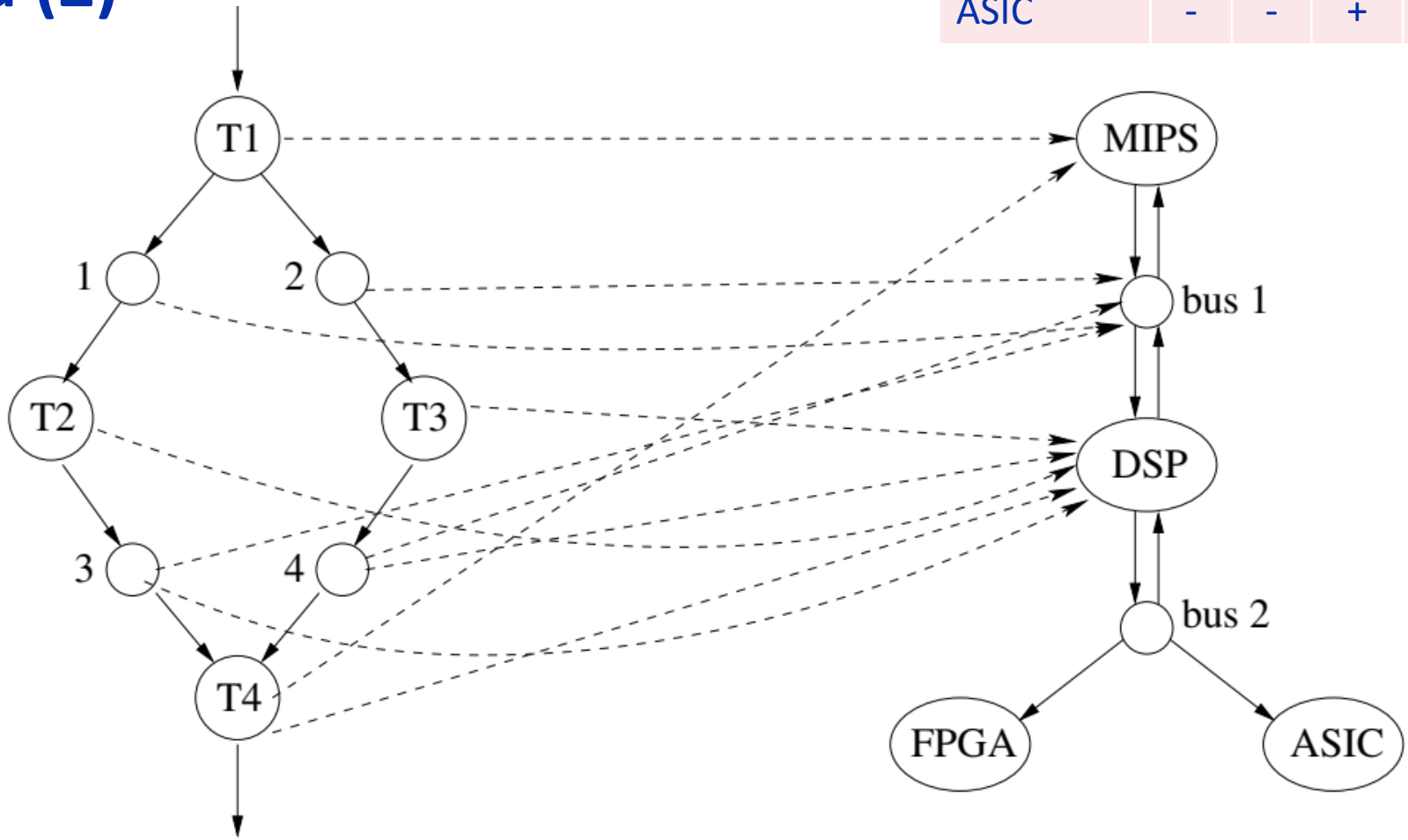




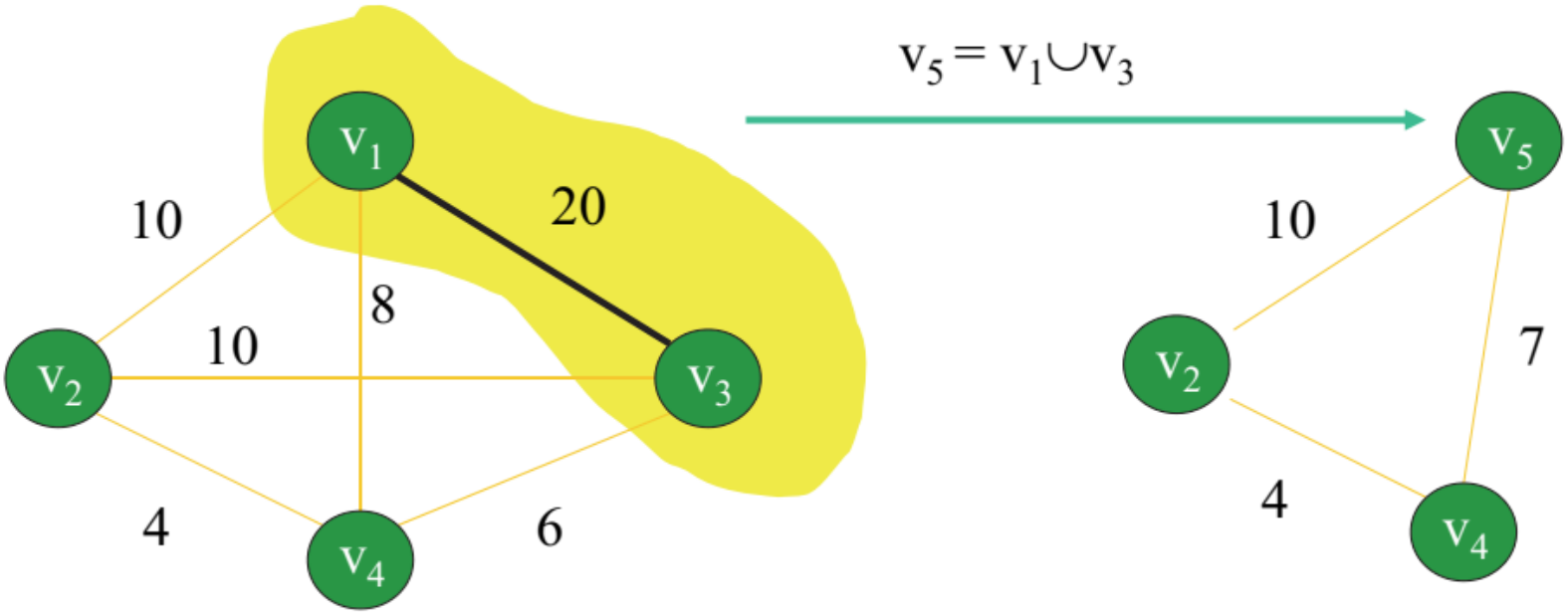
Элемент	T1	T2	T3	T4
MIPS	+	-	-	+
DSP	-	+	+	+
FPGA	-	+	+	-
ASIC	-	-	+	-

Разделение с использованием графа (2)

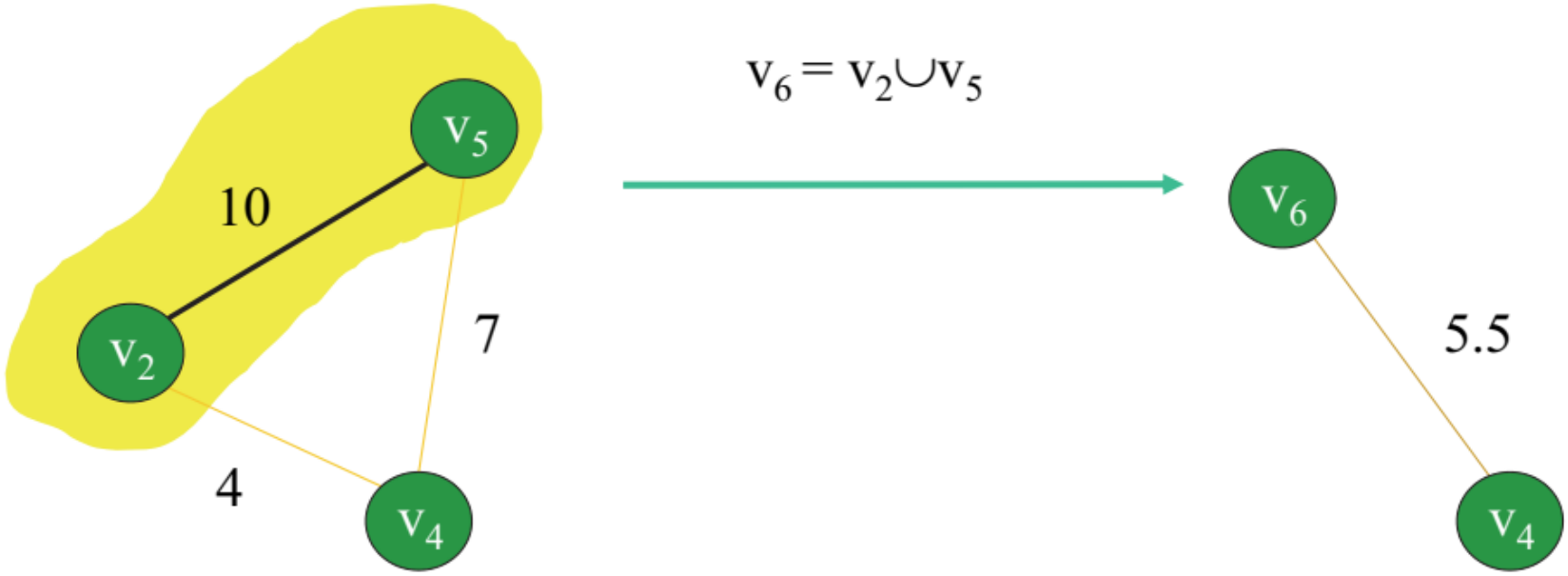
Элемент	T1	T2	T3	T4
MIPS	+	-	-	+
DSP	-	+	+	+
FPGA	-	+	+	-
ASIC	-	-	+	-



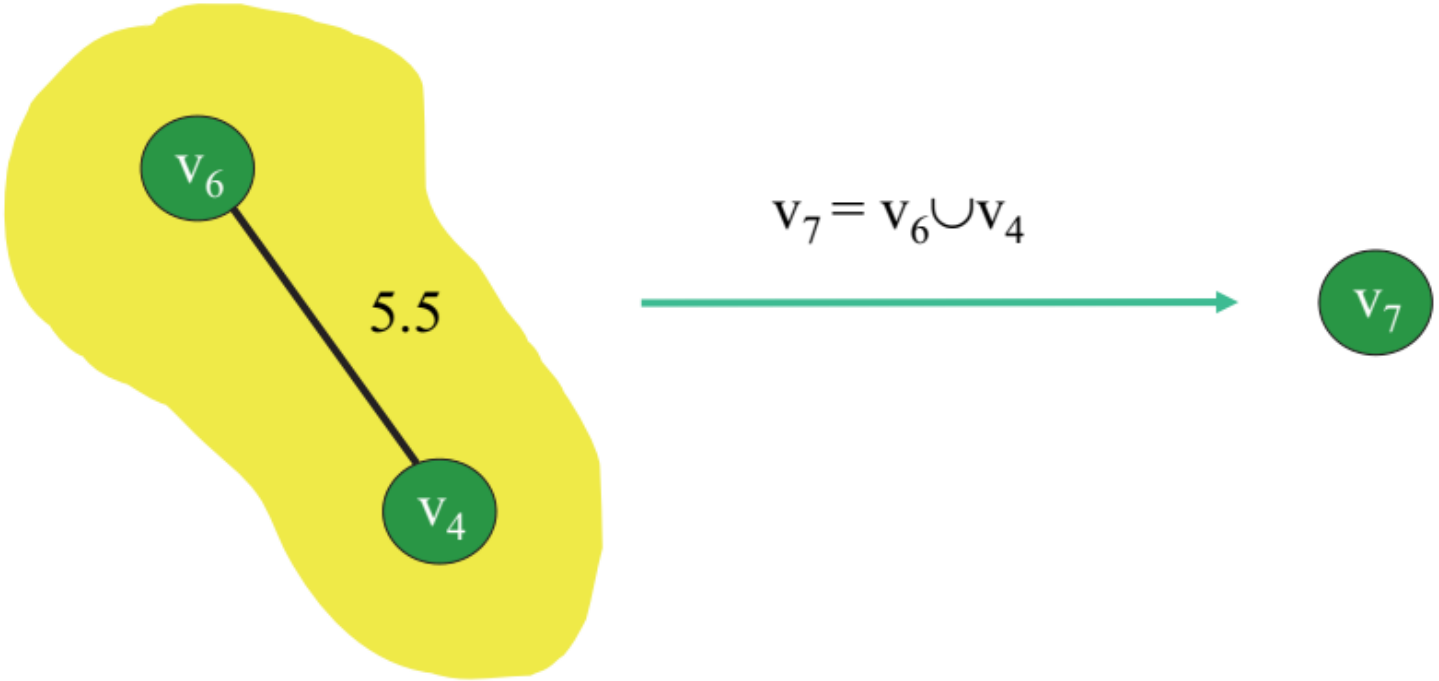
Иерархическая кластеризация (1)



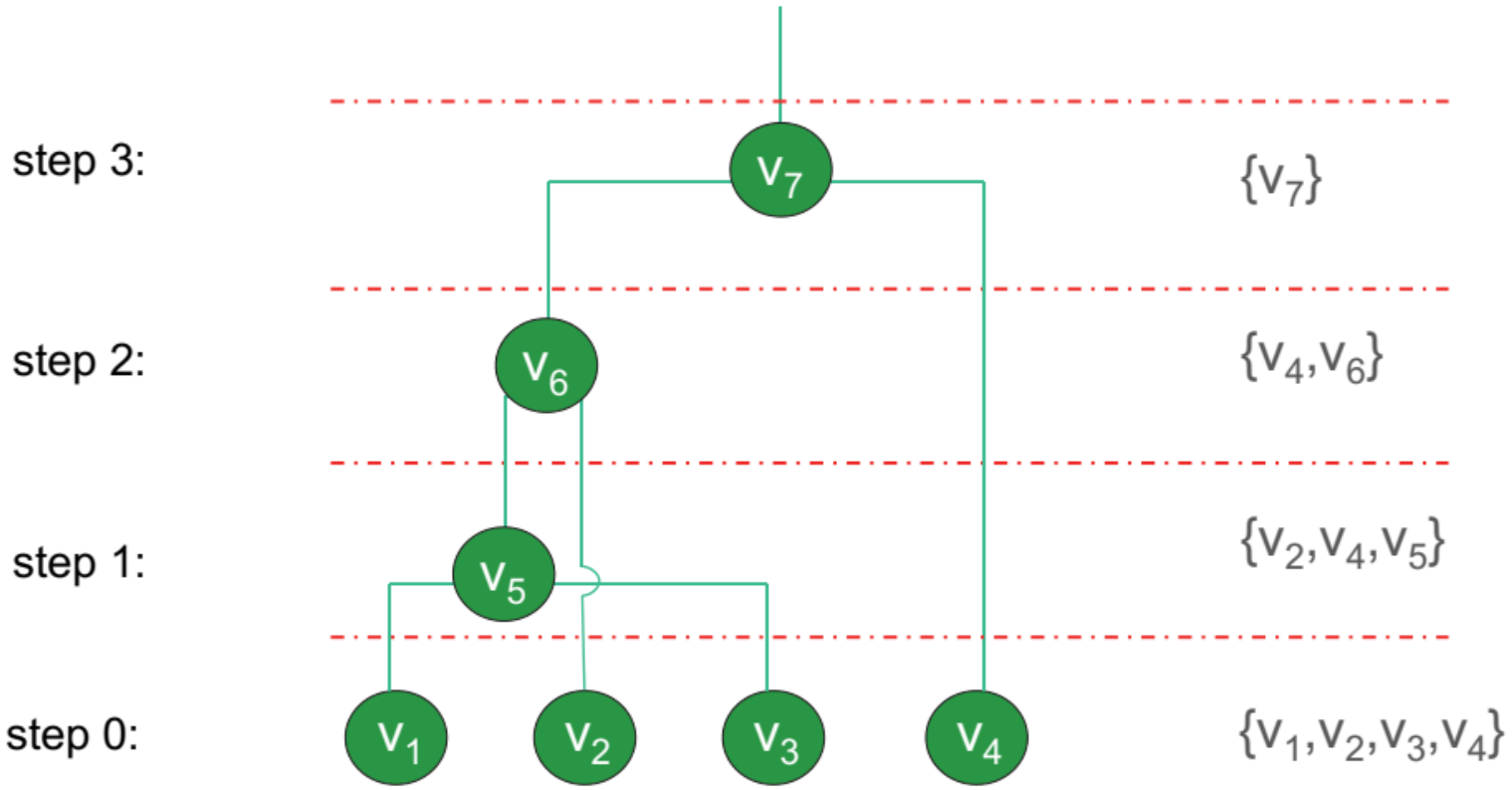
Иерархическая кластеризация (2)



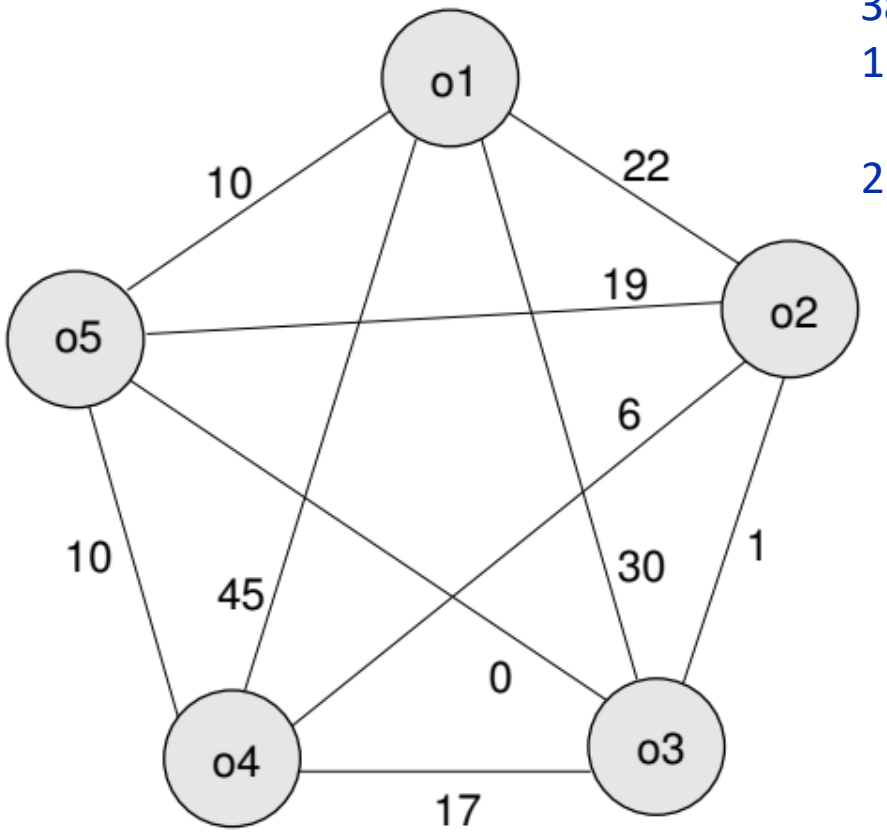
Иерархическая кластеризация (3)



Иерархическая кластеризация (4)



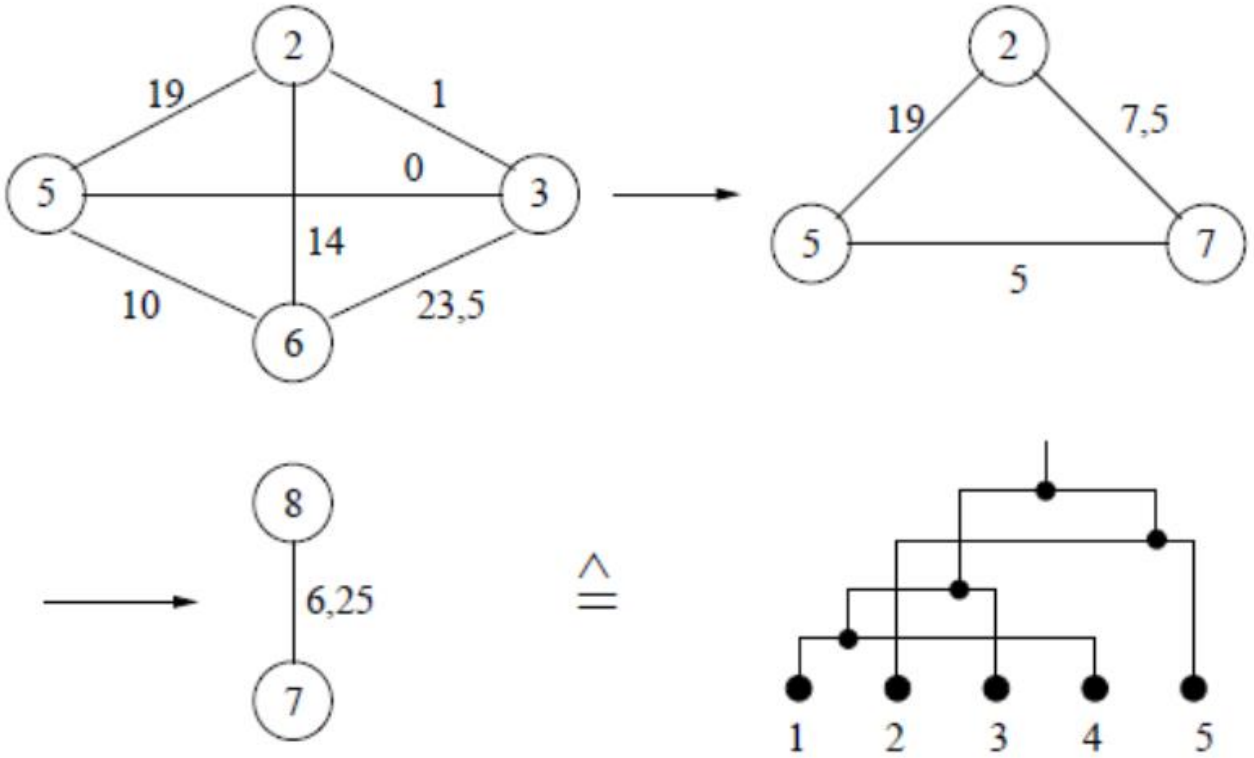
Иерархическая кластеризация (пример)



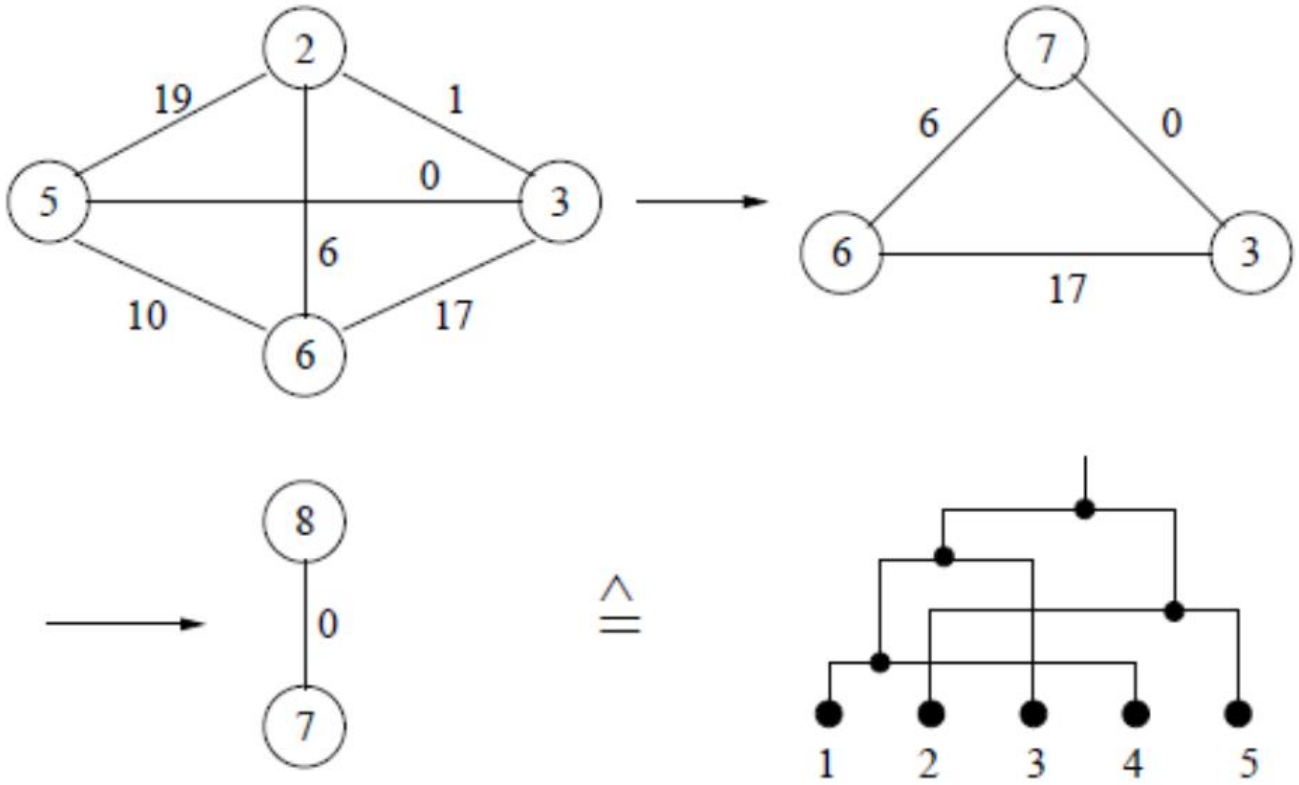
Задачи:

1. Выполнить кластеризацию, используя усреднение весов дуг
2. Выполнить кластеризацию, используя минимальное значение веса дуг

Иерархическая кластеризация (ответ 1)



Иерархическая кластеризация (ответ 2)



Примеры тем для рефератов

- Обзор средств аппаратной эмуляции проекта СнК на ПЛИС
- Обзор средств синтеза LegUp и LeFlow
- Обзор САПР синтеза нейронных сетей на ПЛИС
- Обзор средств формальной верификации, доступных в современных САПР
- Обзор современных HCL (hardware construction language) языков
- Обзор языка BlueSpec. Возможности, преимущества и недостатки
- Обзор ADL (architecture description language) языков
- Архитектура современных GPU. Возможности программирования.
- Обзор технологии HSA
- Обзор языка OpenCL в контексте программирования гетерогенных СнК

УНИВЕРСИТЕТ ИТМО

Спасибо за внимание!

sergei_bykovskii@itmo.ru

Санкт-Петербург, 2019