Санкт-Петербургский государственный университет

Кафедра системного программирования

Разработка универсального скрипта моделирования систем цифровой связи в среде Python

Никита Михайлович Фаст, 371 группа

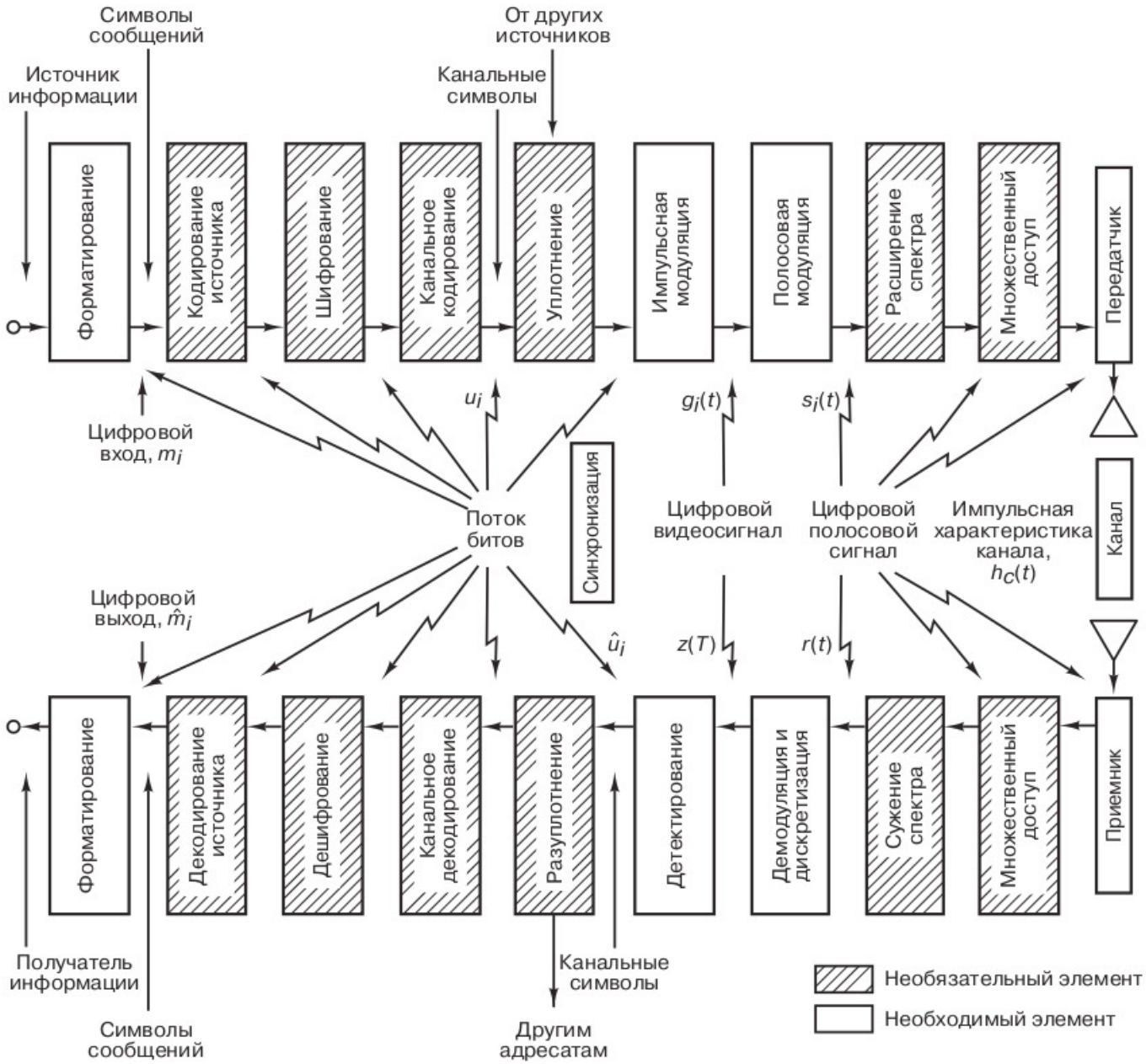
Научный руководитель: к.ф.-м.н. Д.В. Луцив, доцент кафедры СП
Консультант: А.С. Кривоногов, начальник отдела программирования АО “Концерн Гранит”

Санкт-Петербург
2023

Введение

- Требуется проверить, удовлетворяет ли помехоустойчивость разрабатываемой системы цифровой связи требованиям заказчика
- Один из подходов это построить модель, осуществить моделирование на ее основе и получить кривую помехоустойчивости
- Предлагается разработать моделирующую систему с графическим интерфейсом, которая позволит оперативно создать требуемую модель и на ее основе установить помехоустойчивость разрабатываемой системы

Структура системы цифровой связи



Альтернативный подход

Альтернативным методом выяснения помехоустойчивости системы цифровой связи является строгий теоретический вывод формулы, определяющей зависимость между вероятностью появления ошибочного бита P_b и отношением E_b/N_0.
Преимущества:

- быстрота получения результата по готовой формуле
- возможность анализа системы при очень низких значениях вероятности битовой ошибки

Недостатки:

- вывод формул для реальных систем затруднителен
- изменения в системе ведут к необходимости в перерасчете формулы

Существующие решения

- Simulink это графическая среда на основе MATLAB предназначенная для моделирования, симуляции и анализа динамических систем. В данный момент заблокирован. Крайне универсален и из-за этого тяжеловесен. Подход к моделированию в Simulink избыточен для поставленной в работе задачи
- GNU Radio это свободное ПО, предназначенное для разработки систем обработки сигналов. Распространяется по лицензии GPL v3. Поддерживает модули реализованные пользователем. Содержит встроенные инструменты BER Tool и GNU Radio Companion. Умеет генерировать по моделе код на C++ или Python, но не на VHDL. Поддерживает параллельные, но не поддерживает распределенные вычисления

Постановка задачи

Целью работы является реализация моделирующей системы, позволяющей оперативно построить модель и осуществить моделирование, получив в итоге кривую помехоустойчивости

Задачи:

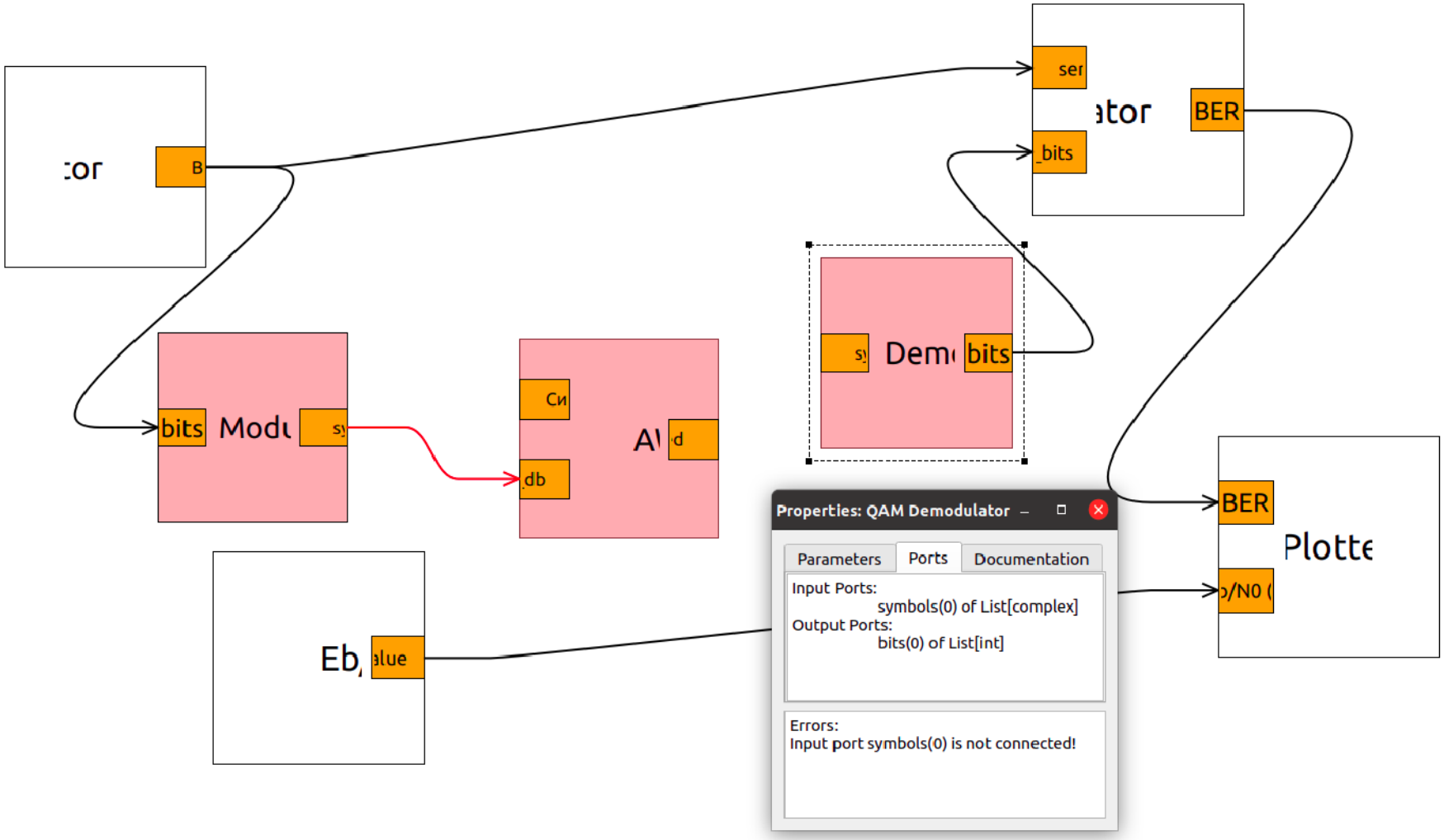
- Разработать архитектуру, позволяющую:
  - поддерживать реализации модулей на Python
  - добавлять в библиотеку пользовательские реализации модулей
  - генерировать код инициализации модели и самого моделирования
  - задействовать для моделирования по крайней мере все ядра одного процессора, а в идеале – все ядра сразу нескольких машин
- Разработать графический интерфейс, который:
  - позволит конструировать модель, комбинируя модули
  - позволит настраивать параметры моделирования
  - будет информировать пользователя о ходе моделирования
- Провести тестирование разработанной моделирующей системы

Дескриптор модуля

```
name = "Binary Generator"
language = "Python"
module_type = 'function'
entry_point = gen_binary_data

output_ports = [
    {
        "label": "Выход",
        "type": List[int]
    },
]
module_parameters = [
    {
        'name': 'bits_num',
        'type': int,
        'has_default_value': True,
        'default_value': 64_000,
        'validator': lambda x: isinstance(x, int) and (0 < x < 1_000_000_000)
    }
]
```

Графическая сцена



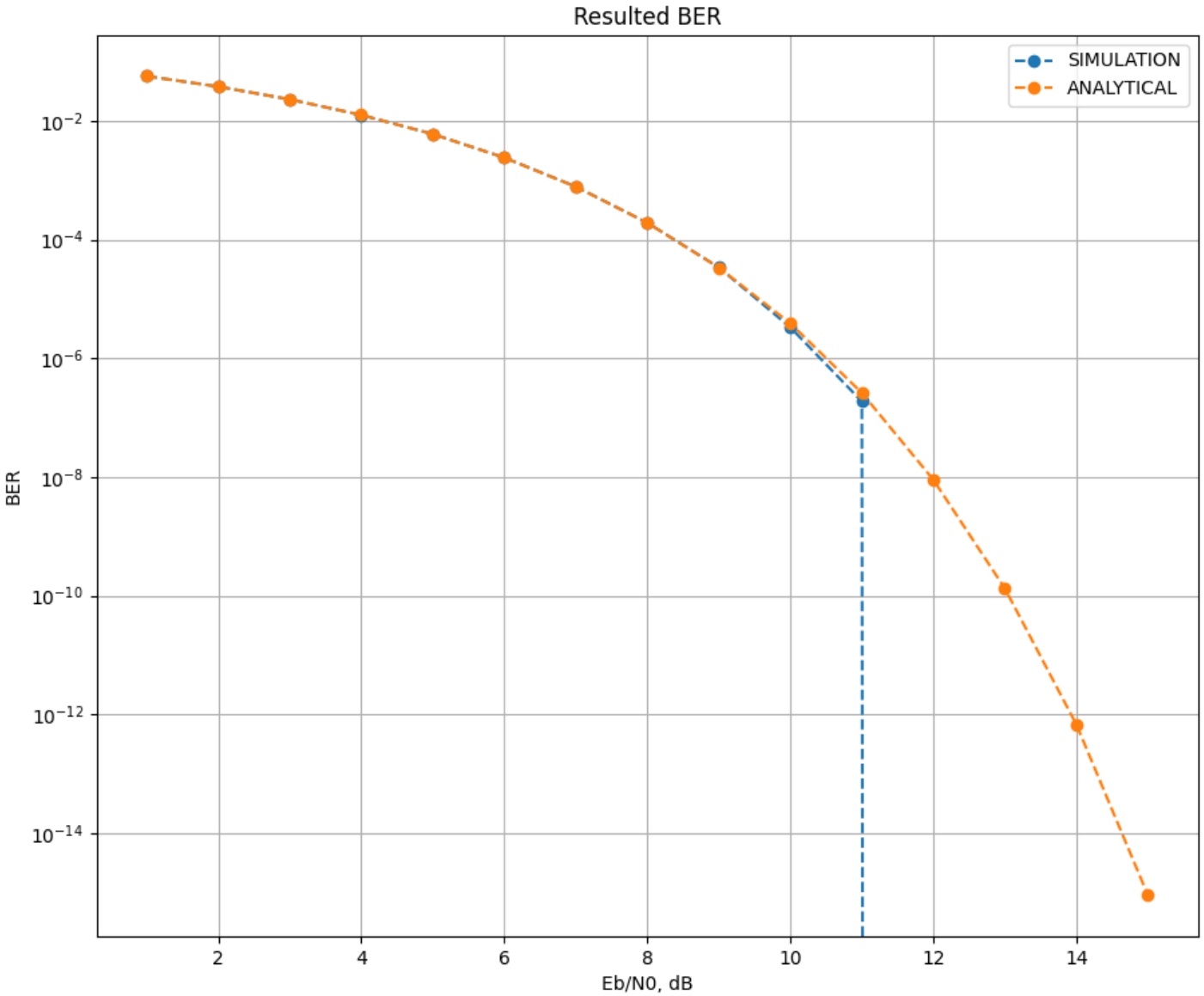
Кодогенерация

- Осуществляется для корректной модели
- Для каждого модуля сериализуем информацию о его исходящих соединениях и выбранных значениях параметров
- Сгенерированный файл подразделяется на 4 части:
  - импорт дескрипторов и вспомогательных классов;
  - конструирование модулей;
  - оборачивание модулей с помощью класса ModuleWrapper;
  - конструирование объекта класса FlowGraph и вызов метода FlowGraph.run();

Моделирование

- Можно выделить для моделирования несколько процессов
- Каждый процесс выполняет несколько итераций моделирования
- В ходе итерации моделирования запускаем модули в порядке, определенном очередью на исполнение:
  - порядок запуска модулей одинаков для всех итераций, поэтому он рассчитывается в конструкторе класса FlowGraph;
  - в очереди на исполнение модули распределены по уровням;
  - начинаем с минимального уровня;
  - когда все модули текущего уровня исполнили, то переходим к следующему уровню;
- Ожидаем завершения всех процессов
- Агрегируем вычисленные значения BER-а и строим график помехоустойчивости

Тестирование



Результаты

По задачам сделано следующее:

- Разработана архитектура, позволяющая:
  - добавлять в библиотеку пользовательские реализации модулей выполненные на Python;
  - осуществлять валидацию параметров, получаемых от пользователя через GUI;
  - используя модель, сформированную через GUI, осуществлять генерацию файла с кодом инициализации модели и кодом выполняющим моделирование;
  - задействовать для моделирования несколько ядер процессора;
- Разработан графический интерфейс, который позволяет:
  - конструировать модель на основе отдельных модулей и того, как они соединены между собой;
  - настраивать параметры отдельных элементов модели и самого моделирования;
  - получать информацию об ошибках в построении модели и ходе моделирования;
- Провели тестирование системы, сравним практические результаты с теоретическими