Задания студентам на практику по программированию.

Функции препроцессора для исходного кода на языке Verilog/SystemVerilog. Выполняются в виде скриптов на языке Python. В зависимости от задачи, при запуске скрипта могут указываться как отдельные параметры, так и использоваться конфигурационные файлы. Для конфигурационных файлов рекомендуется использование формата JSON.

1. Удаление комментариев из исходных кодов.
    a. Удаление всех комментариев типа // и /* */
    b. Выборочное удаление комментариев. Удаление кириллических символов и прочих, не входящих в стандартный набор \0-\127.
    c. Выборочное удаление комментариев по minus списку. Удаление только совпадающих с шаблоном (шаблон использует регулярные выражения).
    d. Выборочное удаление комментариев по plus списку. Удаление всех комментариев, кроме тех, что содержат ключевые слова из списка (регулярные выражения).
2. Обработка ifdef. Выполнение препроцессинга исходного кода. Список путей, по которым искать файлы для include задается внешним параметром. Значения для define параметров могут задаваться как в самом коде, так и внешним параметром при вызове скрипта. Внешний параметр define имеет более высокий приоритет, чем определение в тексте.
    a. Включение в код всех `include файлов
    b. Исключение всех `ifdef - `endif веток, для которых условие не выполняется.
3. Обфусцирование кода. (Замена идентификаторов на случайные сгенерированные имена)
    a. Найти в коде все идентификаторы. Заменить идентификаторы на случайные имена, сохранив таблицу соответствия.
    b. Выполнить обфусцирование только выбранного класса идентификаторов (input/output/inout, wire, reg, module, instance, parameter).
    c. Выполнить обфусцирование кода, кроме портов ввода вывода заданного модуля.
    d. Выполнить обфусцирование кода в пределах операторных скобок //pragma protect on - //pragma protect off
4. Восстановление обфусцированного кода
    a. Восстановить исходный код из обфусцированного, используя таблицу соответствия.
    b. Частично восстановить исходный код из обфусцированного только для выбранного класса идентификаторов (input/output/inout, wire, reg, module, instance, parameter).
    c. Частично восстановить исходный код из обфусцированного только для портов ввода вывода заданного модуля.
5. Чтение иерархии проекта
    a. Для проекта из нескольких вложенных модулей восстановить структуру их вызовов. Сохранить иерархические имена всех экземпляров модулей в файле отчета.
    b. Для проекта из нескольких вложенных модулей сохранить в файле иерархические пути ко всем объектам, с указанием типа объекта (reg, net, instance, port).
    c. Разбиение по файлам. Если в файле содержится несколько модулей, разложить исходный код по отдельным файлам, имя файла совпадает с именем модуля.

В качестве примера иерархического кода на SystemVerilog можно использовать проект Syntacore SCR1 https://github.com/syntacore/scr1/tree/master/src.
Стандарт SystemVerilog https://www.francisz.cn/download/IEEE_Standard_1800-2012%20SystemVerilog.pdf
Допускается использование готовых Python библиотек для работы с SystemVerilog кодом, однако, в учебных целях рекомендуется выполнять задание без использования сторонних библиотек.