**1) Базовые элементы цифровых схем (транзистор, счетчик, регистр, мультиплексор, демультиплексор, компаратор). Принцип работы. Входы и выходы элементов.**

1. Транзистор.

Транзистор имеет два входа и один выход: затвор(верхний), исток и сток соответственно. В зависимости от сигнала на верхнем входе на выход транзистора будет подаваться, либо не подаваться сигнал. Существует два типа транзисторов: p-типа и n-типа. транзистор p-типа (обозначенный кружком на линии затвора) передаёт значение, когда на затворе 0, а транзистор n-типа (без кружка) передаёт значение, когда на затворе 1.

http://www.cburch.com/logisim/docs/2.7/en/img-libs/trans0.png http://www.cburch.com/logisim/docs/2.7/en/img-libs/trans1.png

p-типа n-Типа

1. Счетчик

Имеет один вход(тактовый) и один выход. Когда сигнал на тактовом входе переключается с 0 на 1, то в счетчике значение увеличивается на 1 (начиная с нуля). Максимальное значение счетчика определяется формулой: 2 в степени того, скольки битный счетчик, минус единица. Когда счетчик доходит до максимального значения, он снова начинает отсчет с нуля.

http://www.cburch.com/logisim/docs/2.7/en/img-libs/counter.png

1. Регистр

Имеет два входа и один выход. Когда сигнал на тактовом входе сменяется с 0 на единицу, регистр запоминает значение на левом входе. На выход подается запомненное значение.

http://www.cburch.com/logisim/docs/2.7/en/img-libs/register.png

1. Мультиплексор

Имеет входа снизу. Правый нижний для питания, левый - управляющий вход. Количество входов слева рассчитывается по формуле: 2 в степени того, скольки битный мультиплексор. На выход подается значение с того входа, номер которого подается на управляющем входе.

http://www.cburch.com/logisim/docs/2.7/en/img-libs/mux.png

1. Демультиплексор

Имеет один вход слева и два входа снизу. Левый нижний для питания, правый - управляющий вход. Количество выходов рассчитывается по формуле: 2 в степени того, скольки битный демультиплексор. С входа значение подается на тот выход, номер которого подается на управляющий вход.

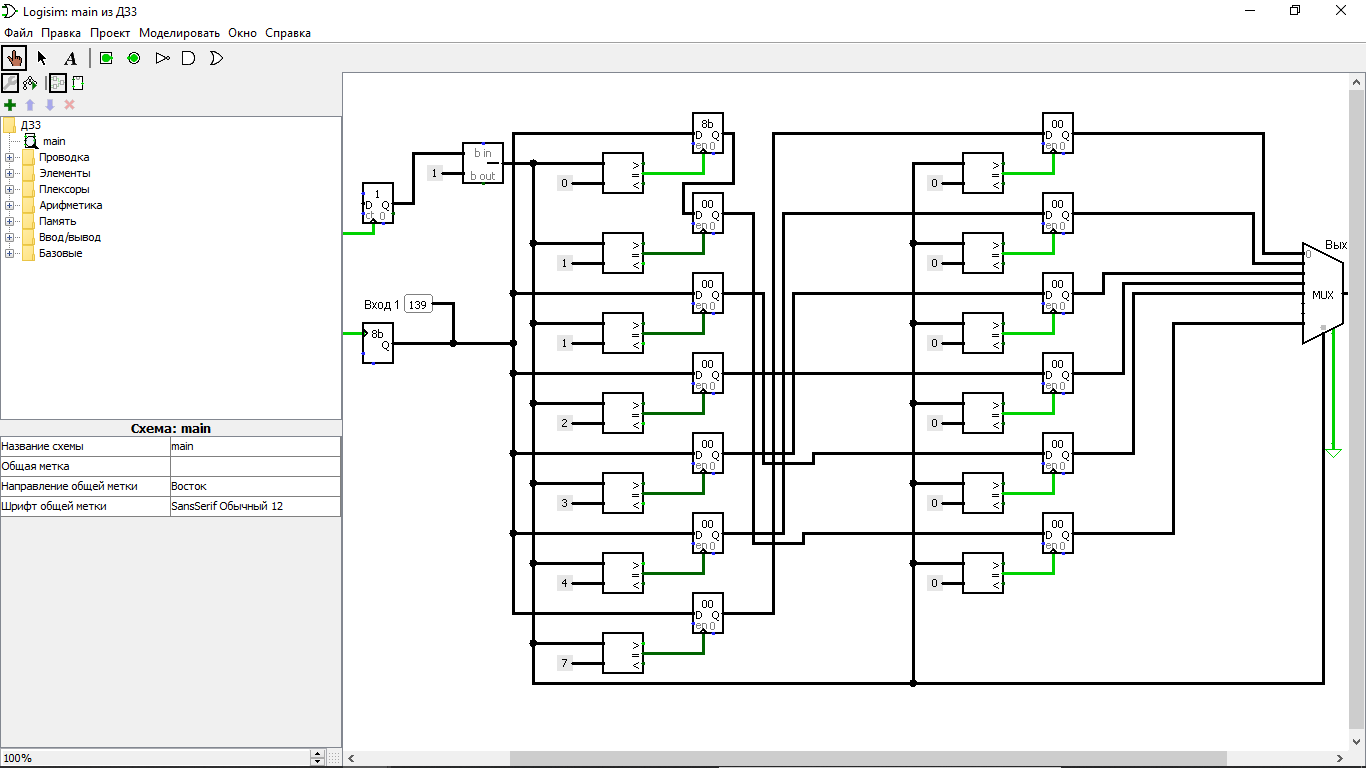
http://www.cburch.com/logisim/docs/2.7/en/img-libs/demux.png

1. Компаратор

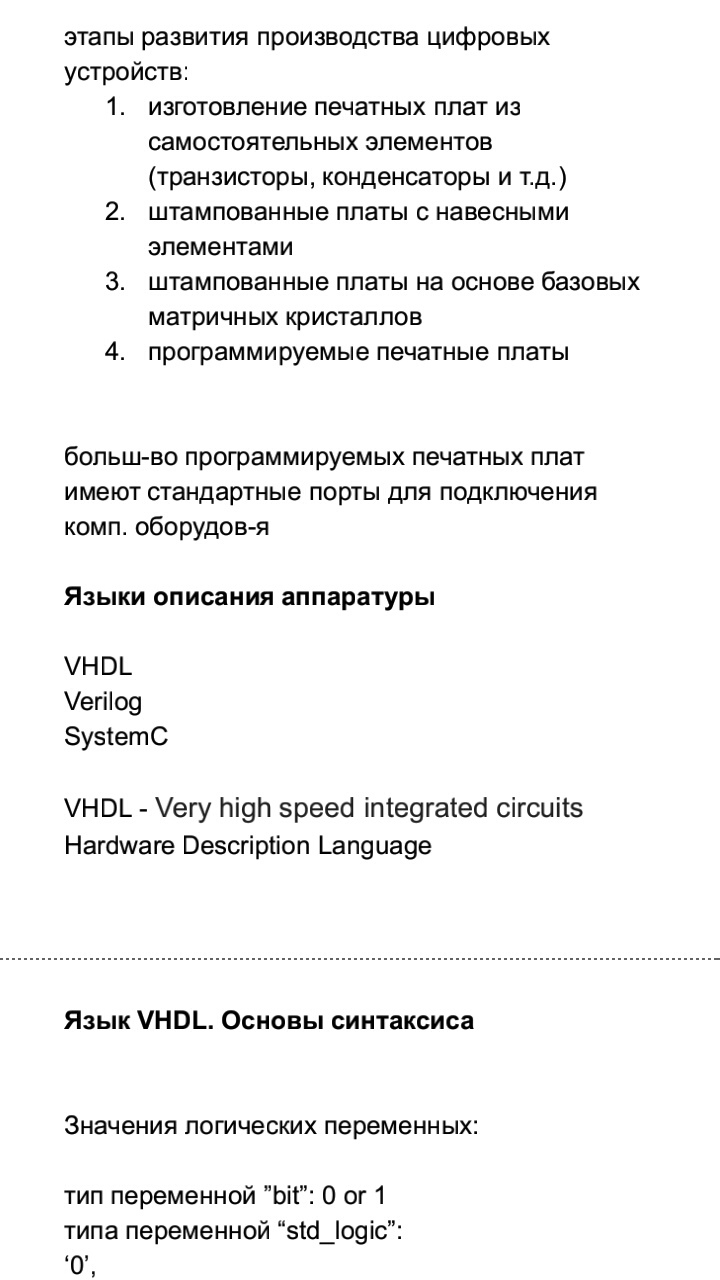
Сравнивает значения. Имеет два входа. Левый верхний – то, что сравнивают, нижний – то, с чем сравнивают. Также имеет три выхода. Единица может подаваться лишь на один из них, в зависимости от результата сравнения значений на входах. Если значение на верхнем входе больше значения на нижнем, то единица подается на самый верхний выход. Если значения на входах равны, то единица подается на выход посередине. Если значение на верхнем входе меньше значения на нижнем, то единица подается на нижний выход.

http://www.cburch.com/logisim/docs/2.7/en/img-libs/comparator.png

**2) Разработка логической схемы блочного шифрования 8-ми битного сигнала методом перестановок.**

****

Для начала нам нужны 8 регистров(по одному регистру на каждый бит). Также поставим счетчик(3 бита, чтобы считал от 0 до 7(столько значений, скольки битный сигнал мы шифруем)), чтобы делать все последовательно, в зависимости от нужного такта после, после счетчика ставим вычитатель и к нижнему входу константой подаем единицу(т.к. у нас первое переключение счетчика идет с 0 в 1, и нулевой такт при первом запуске оказывается выброшенным), а также ставим генератор случайных чисел – он будет генерировать значения, которые мы будем загружать в регистры. Счетчик и генератор случайных чисел подключаем к одному тактовому генератору. Генератор случайных чисел подключаем к каждому из 8 регистров. Но в таком случае все регистры будут запоминать одинаковые значения с генератора, чтобы этого не произошло, добавляем 8 компараторов – по одному к каждому регистру. Средние выходы компараторов(знак «равно») соединяем с вычитателем. На верхний вход компараторов подаем значения с вычитателя, а на нижний вход подаем константы от 0 до 7 последовательно. Так в зависимости от номера такта определенный регистр будет запоминать определенное значение. Затем добавляем еще 8 регистров и еще 8 компараторов, к ним подключенных. Выходы первых 8 регистров соединяем со входами вторых 8 регистров в зависимости от того, как мы шифруем сигнал. На верхние входы компараторов подаем сигнал с вычитателя, на нижние – константами подаем нули. В нулевой такт сигнал зашифруется. Но в таком случае регистр, в который записывается значение при нулевом такте, будет работать некорректно, т.к. в один такт ему нужно и записать сигнал и отдать его на следующий регистр. Чтобы этого не произошло, на первом регистре меняем значение константы на единицу, а также добавляем еще один регистр и компаратор. В нулевой такт этот регистр будет запоминать значение, а в первый такт наш первый регистр будет принимать это значение. И в заключение ставим мультиплексор и последовательно к его входам подключаем выходы регистров. К управляющему входу подключаем наш вычитатель. Таким образом последовательно выводим зашифрованный сигнал.

**3) Программируемые логические интегральные схемы (ПЛИС). История создания. Принцип работы. Примеры реальных плат.**

В программируемых платах накидываем нули и единицы на ячейки памяти возле транзисторов, в результате выполняется нужный алгоритм.

Примеры плат - Zync-7000, Artix-7, Kintex-7

**4) Язык VHDL. Базовая структура описания устройства на языке VHDL. Блоки entity, architecture, component, port map.**

Раздел Entity(для всей схемы) описывает внешний интерфейс схемы, т.е. содержит список сигналов, с помощью которых схема связана с окружающим ее миром. Раздел Architecture описывает, как эта схема функционирует, т.е. описывает то, что у схемы внутри

Между ключевыми словами component и end component описывается интерфейс компонента, а с помощью оператора port map конкретизируется его связь с внешними сигналами.

.

Entity(для компонента) – объявление модуля. Описывает внешний интерфейс модуля. Architecture – архитектура модуля. Описывает внутреннюю реализацию.

Для начала нужно подключить библиотеки. После их подключения задаем блок entity для всей нашей схемы. В ней описываем все входы и выходы, которые присутствуют в нашей схеме.  
Затем идет блок architecture для всей схемы. В ней объявляются блоки component.

component N -объявляем блок  
 port(A: in bit; -прописываем входы, указываем их тип данных  
 Y: out bit); -и выходы, указываем их тип данных  
end component; -закрываем блок

Сколько типов компонентов, столько и блоков component. И также объявляются сигналы.

Затем следует begin и блоки port map

N1\_1: N port map(s7,s8); - указываем имя компонента, затем тип компонента(может быть несколько компонентов с разными именами, но одного типа) и в скобках прописываем входящие и выходящие сигналы.

После чего закрываем architecture всего проекта: end Behavioral.

Затем прописываем блоки entity и architecture для каждого компонента.

entity N is  
 port(A: in bit;  
 Y: out bit);  
end N;

architecture N\_Arch of N is  
begin  
Y<= not A after 1 ns;  
end N\_Arch;

**5) Язык VHDL. Типы данных на языке VHDL. Параметры объявления и присвоения значений сигналов и переменных.**

Основными типами данных в языке VHDL являются bit(‘1’,’0’), bit\_vector(последовательность значений типа bit), integer - целые числа, real – дробные числа, Boolean – логическое true или false  
character – один символ, string – строка,std\_logic  
std\_logic: (‘U’, ‘X’, ‘0’, ‘1’, ‘Z’, ‘W’, ‘L’, ‘H’, ‘-’) Значения для синтеза: • ‘0’, ‘1’ – тоже что и в типе bit – обыкновенные ‘0’ и ‘1’ • ‘Z’ – состояние с высоким импедансом • ‘X’ – неизвестное Значения для моделирования: • ‘L’, ‘H’ – слабый ‘0’, слабая ‘1’ (имеется в виду слабый ток) • ‘W’ –слабое неизвестное • ‘U’ – неинициализированное • ‘-’ – неважное значение

Объявление и присвоение переменных

Variable x1: bit;  
x1:=’0’;  
variable x2: bit\_vector (7 downto 0);  
x2:=”00000000”;  
variable x3: integer;  
x3:=0;

Объявление и присвоение сигналов. Сигналы нужно создавать когда(1) проводник не  
подключается напрямую ни к одному из входов и выходов(2)проводник подключен к какому-нибудь выходу, но дополнительно этот сигнал подается и на другой компонент(3)сигнал одного компонента подается сразу на несколько выходов устройства.

Signal s1: bit;  
s1<=’0’;  
signal s2: bit\_vector (7 downto 0);  
s2<=”00000000”;  
signal s3: integer;  
s3<=0;

Когда проводник подключен к какому-нибудь выходу, но этот сигнал подается и на другой компонент, то присваиваем выходу этот сигнал: y<=s1;

**6) Язык VHDL. Создание процессов для описания функциональной модели устройства. Блоки process. Их параметры. Пример блока process и инструкцией if… then… else.**

Оператор процесс является поведенческим описанием. При использовании оператора process каждый выход рассчитывается последовательностью команд, как в языках программирования.  
В скобках оператора process пишутся те сигналы/переменные, при изменении которых выполняется данный процесс.

SimulationProcess: process

begin

xx1<='0';  
xx2<='0';  
xx3<='0';  
xx4<='0';

for i in 1 to 16 loop  
if((i-1)mod 16<8) then xx1<='0'; else xx1<='1'; end if;  
if((i-1)mod 8<4) then xx2<='0'; else xx2<='1'; end if;  
if((i-1)mod 4<2) then xx3<='0'; else xx3<='1'; end if;  
if((i-1)mod 2<1) then xx4<='0'; else xx4<='1'; end if;  
wait for 50 ns;  
end loop;  
xx1<='0';  
xx2<='0';  
xx3<='0';  
xx4<='0';  
wait;  
end process;

**7)Математическое моделирование в программе FreeMat. Создание переменных, массивов, двухмерных массивов, и их заполнение и использование. Способы заполнения массивов последовательностями чисел (заполнение нулями, единицами, случайными числами и 2 способа заполнения последовательностью чисел) с примерами.**

Freemat -свободно распространяемая программа для математического моделирования и обработки данных.Matlab.

Объявление переменной a: a = 2; Также переменная может объявиться самостоятельно. Когда мы к примеру просто складываем 2 числа. В таком случае фримат создаст переменную ans(answer) и присвоит ей результат вычисления. Переменная, по сути, и является числом, и выполнять операции над ней мы можем как с обычным числом a=a/2; a=1; b = 2+a; b=3.

Возведение переменной в степень производится таким образом b.^2; b=25;

Объявление массива: mas=[0 1 2]. Одномерный массив с соответствующими значениями.

Можем вставлять переменную в массив: mas=[0 1 2 b]; mas=0123

Многомерный массив:matr= [0 1 2 3; 0 1 2 3]. Двумерный массив.

0123  
0123

Можем вставлять массивы друг в друга:matr=[0 1 2 3; 0 1 2 3;mas]

0123  
0123  
0123

Заполнение нулями: matr = zeros(4,4);  
Заполнение единицами: matr = ones(4,4);  
Заполнение случайными числами:matr = rand(4,4);  
Заполнение последовательности чисел 1: x=0:1:10; x = 0 1 2 3 4 5 6 7 8 9 10;  
Заполнение последовательности чисел 2: x=linspace(0,10,10); x = 0 1 2 3 4 5 6 7 8 9 10;

Linspace(минимальное значение, максимальное значение, количество значений)

Пример: построим данную матрицу с помощью цикла

0111210  
0113110  
0141110  
0511110

a=zeros(4,7); - создаем матрицу из 4 строчек и 7 столбцов, состоящий из нулей  
b=[0 1 1 1 2 1 0]; -создаем одномерный массив с соответствующими значениями  
a(1,1:7)=b; - вставляем наш массив в матрицу на 1 строку с 1 по 7 столбцы  
i=1  
c=2  
b(1,5-i)=c+i; | - заменяем значение в первой строке 5-i столбце на c+i  
b(1,6-i)=1; | цикл -меняем следующее значение после того, которое заменили, на 1  
a(1+i,1:7)=b; | -вставляем массив b в следующую строку с 1 по 7 столбцы  
i=i+1; | -увеличиваем i на 1

**8) Математическое моделирование в программе FreeMat. Построение графика функции с указанием параметров графика, осей, координатной сетки.**

Зададим несколько переменных с рядами значений  
x= 0:1:10;  
y=x\*2;  
plot(x,y); - строим график по соответствующим значениям x и y  
axis([minx maxx miny maxy]); - задание масштаба  
grid on ; - создаем координатную сетку  
axis equal; - и делаем эту сетку всегда симметричной вне зависимости от масштаба

**9)Математическое моделирование в программе FreeMat. Построение нескольких графиков функций в одних осях, на разных осях в одном окне, в разных окнах.**

Зададим несколько переменных с рядами значений  
x= 0:1:10;  
y=x\*2;  
x1= 0:2:20;  
y1=sin(x1);  
plot(x,y,x1,y1); - строим два графика на одной плоскости  
Для построения графика на разных осях в одном окне, нужно задать сколько плоскостей будет.  
  
subplot(1,2,1); -Задаем , что будет 1 строка, 2 столбца, и обращаемся к первому месту для графика  
plot(x,y); - и на этом месте строим график по значениям x и y  
subplot(1,2,2); - обращаемся ко второму месту для графика  
plot(x1,y1); - строим в нем второй график

Для построения графиках в разных окнах, нужно для начала эти окна создать  
fugure; - создаем новое окно  
plot(x,y); - строим в нем график  
figure; - создаем второе окно  
plot(x1,y1); строим второй график  
Итого – 2 окна, 2 графика