Билет №3
Пункт №1: Практическая часть (Харрис, упр 4.25)

Упражнение 4.25 Нарисуйте диаграмму состояний конечного автомата, описанного кодом на HDL, привеженным ниже. Автоматы подобного типа используются для предсказания переходов в некоторых микропроцессорах.

SystemVerilog

```
module fsm1(input logic clk, reset,
            input logic taken, back,
            output logic predicttaken);

  logic [4:0] state, nextstate;

  parameter S0 = 5'b00001;
  parameter S1 = 5'b00010;
  parameter S2 = 5'b00100;
  parameter S3 = 5'b01000;
  parameter S4 = 5'b10000;
  always_ff @(posedge clk, posedge reset)
   if (reset) state <= S2;
   else       state <= nextstate;
  always_comb
    case (state)
      S0: if (taken) nextstate = S1;
          else       nextstate = S0;
      S1: if (taken) nextstate = S2;
          else       nextstate = S0;
      S2: if (taken) nextstate = S3;
          else       nextstate = S1;
      S3: if (taken) nextstate = S4;
          else       nextstate = S2;
      S4: if (taken) nextstate = S4;
          else       nextstate = S3;
      default:       nextstate = S2;
    endcase
  assign predicttaken = (state == S4) |
                        (state == S3) |
                        (state == S2 &&
back);
endmodule
```

VHDL

```
library IEEE; use IEEE.STD_LOGIC_1164. all;
entity fsm1 is
  port(clk, reset:    in  STD_LOGIC;
       taken, back:   in  STD_LOGIC;
       predicttaken:  out STD_LOGIC);
end;
architecture synth of fsm1 is
  type statetype is (S0, S1, S2, S3, S4);
  signal state, nextstate: statetype;
begin
  process(clk, reset) begin
    if reset then state <= S2;
    elsif rising_edge(clk) then
      state <= nextstate;
    end if;
  end process;
process(all) begin
   case state is
    when S0 => if taken then
                    nextstate <= S1;
               else nextstate <= S0;
               end if;
    when S1 => if taken then
                    nextstate => S2;
               else nextstate <= S0;
               end if;
    when S2 => if taken then
                    nextstate <= S3;
               else nextstate <= S1;
               end if;
    when S3 => if taken then
                    nextstate <= S4;
               else nextstate <= S2;
               end if;
    when S4 => if taken then
                    nextstate <= S4;
               else nextstate <= S3;
               end if;
    when others => nextstate <= S2;
  end case;
end process;
-- логика выхода
predicttaken <= '1' when
               ((state = S4) or (state = S3) or
               (state = S2 and back = '1'))
  else '0';
end;
```

Ответ из решебника:

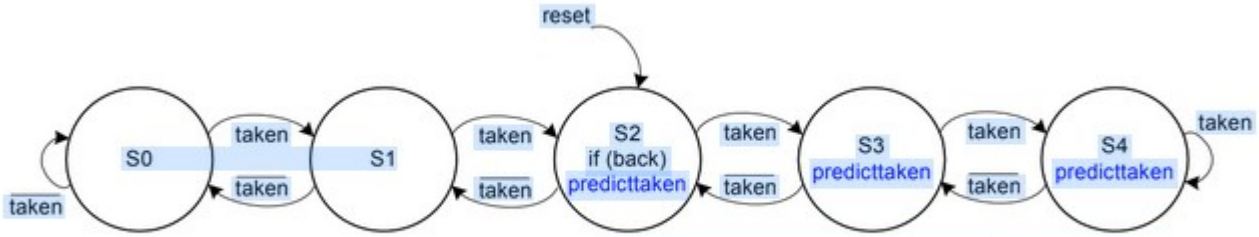

FIGURE 4.1 State transition diagram for Exercise 4.25

Пункт №2:

Разработайте конечный автомат, который принимает последовательность битов (один бит за раз) и выполняет над ними операцию преобразования в дополнительный код. Он имеет два входа, Start и A, и один выход Q. Двоичное число произвольной длины подается на вход A, начиная с младшего разряда. Соответствующий выходной бит появляется на том же цикле на выходе Q. Вход Start устанавливается на один цикл для инициализации конечного автомата перед поступлением младшего бита.

Ответ из решебника:

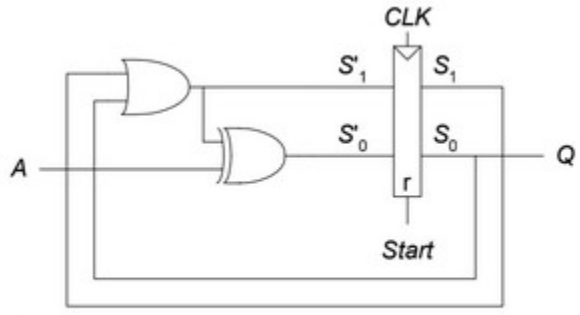

FIGURE 3.18 Finite state machine hardware for Question 3.2

это и есть автомат, ниже приведена диаграмма и таблица переходов(не думаю, что будут обязательны для сдачи вопроса)

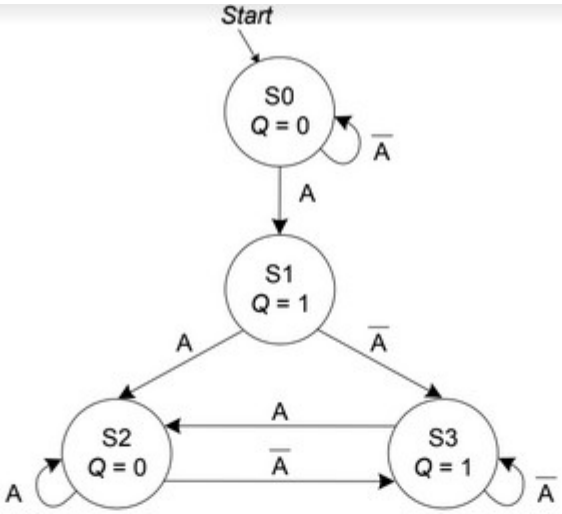

FIGURE 3.17 State transition diagram for Question 3.2

current state $s_{1:0}$	input a	next state $s'_{1:0}$
00	0	00
00	1	01
01	0	11
01	1	10
10	0	11
10	1	10
11	0	11
11	1	10

TABLE 3.22 State transition table for Question 3.2

$$S'_1 = S_1 + S_0$$

$$S'_0 = A \oplus (S_1 + S_0)$$

$$Q = S_0$$

Пункт №3:

Иерархия памяти. Анализ производительности систем памяти.

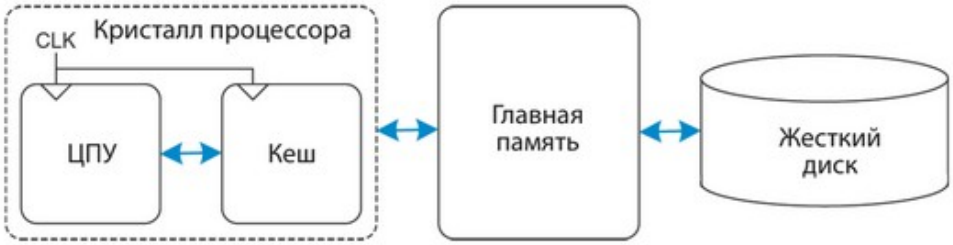

Рис. 8.3 Типичная иерархия памяти

Технология	Цена за ГБ	Время доступа (нс)	Пропускная способность (ГБ/с)
SRAM	$100	0,2–3	100+
DRAM	$3	10–50	30
SSD	$0,10	20 000	0,05–3
HDD	$0,03	5 000 000	0,001–0,1

Рис. 8.4 Компоненты иерархии памяти и их характеристики на 2021 год

Пикча из харриса стр587

Память ЭВМ должна иметь большую информационную емкость, малое время обращения (высокое быстродействие), высокую надежность и низкую стоимость. Но с увеличением емкости снижается быстродействие и растет стоимость. Деление памяти на ОЗУ и ПЗУ не снимает это противоречие полностью, так как различие в быстродействии процессора, ОЗУ и ПЗУ очень велико. Поэтому обмен информацией производится через дополнительные буферные устройства, то есть память ЭВМ имеет иерархическую многоуровневую структуру(как показано на пикче). Чем больше быстродействие

ЗУ, тем выше стоимость хранения 1 байта, тем меньшую емкость имеет ЗУ.

Микропроцессорная память – высокоскоростная память небольшой емкости, входящая в МП и используемая АЛУ для хранения операндов и промежуточных результатов вычислений. КЭШ-память – это буферная, не доступная для пользователя память, автоматически используемая компьютером для ускорения операций с информацией, хранящейся в медленно действующих запоминающих устройствах. Для ускорения операций с основной памятью организуется регистровая КЭШ-память внутри микропроцессора (КЭШ-память первого уровня) или вне микропроцессора на материнской плате (КЭШ-память второго уровня); для ускорения операций с дисковой памятью организуется КЭШ-память на ячейках электронной памяти.

Внутренняя память состоит из ПЗУ (ROM – Read Only Memory) и ОЗУ(оперативка) (RAM – Random Access Memory – память с произвольным доступом). ПЗУ состоит из установленных на материнской плате микросхем и используется для хранения неизменяемой информации: загрузочных программ операционной системы (ОС), программ тестирования устройств компьютера и некоторых драйверов базовой системы ввода-вывода (BIOS – Base Input-Output System) и др. Из ПЗУ можно только считывать информацию, емкость ПЗУ – сотни Кбайт(ваш жесткий диск). Это энергонезависимая память, – при отключении ЭВМ информация сохраняется.

При рассказе можете упомянуть, что еще бывает внешняя память (флешки, диски и прочее)
Если смотреть на пикчу(пирамиду), то видна струткура памяти. Когда процессору необходиом получить данные из памяти он проверяет все по порядку: регистры процессора → кеш-память → главная память → виртуальная память.

Чем дальше от процессора, тем меньше скорость получения данных, или же пропускная способность, Соответственно, повышается время доступа(больше чем в десятки раз). НЕ ЗАБУДЬТЕ, ЧТО КЭШ ПАМЯТЬ МНОГОУРОВНЕВАЯ. На самом деле будет достаточно рассказать про ту пикчу, что в начале ответа, но будет ваще офигенно, если сможете рассказать, что написал тут выше. Там как раз структура, зависимость цена-скорость.

https://works.doklad.ru/view/2eccBmiOF8Q.html
это ссылка откуда инфу брал, для тех, кто что-то не понял

там немного другие названия и по-другому показана пирамида (лучше смотрите на ту, что я скинул с харриса)