RTL Mетодология

[Описание](#_aa7wllkqx85t)

[ASM Диаграмa](#_ljsam0yte6zo)

[RT (register transfer) Операции](#_hbh6fqx3bfbi)

[Пример - Fibonacci](#_x07qb37clt4r)

[АSMD диаграма](#_equtndkxako2)

[SystemVerilog code](#_18b2phfgikwv)

[Пример - Най-голям Общ Делител](#_6bybt6no85co)

# Описание

Методика за проектиране на цифрови схеми за обработка на дани.

Разделя описанието на схемата на функционални блокове за преобразуване на данните (комбинационна логика), регистри за съхранение на данните и крайни автомати за управление на последователността на обработка и предаване на данните.



| Хардуер | Диаграма (графичен модел) |
| --- | --- |
| FSM - finite state machine  краен автомат | ASM chart, State diagram |
| FSMD - finite state machine with datapath  краен автомат + схема за обработка на данни (datapath) | ASMD chart = ASM + RT ops |

## ASM Диаграмa



## RT (register transfer) Операции

regi <- f(reg1, … , regN, inputs)

Примери

| reg1 <- const | запис на константа в регистъра reg1 |
| --- | --- |
| reg1 <- reg1 + reg2 | запис на резултат от аритметична операция |
| reg1 <- reg1 << 1 | преместване на един бит в ляво |
| reg1 <- reg1 | запазване състоянието на регистъра |
| reg1 <- input1 | запис на входния сигнал input1 в регистъра reg1 |

# Пример - Fibonacci

Ред на Фибоначи

| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| f(n) | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 |  |

f(n) = f(n-1) + f(n-2)

f(0) = 0

f(1) = 1

## АSMD диаграма

|  | Диаграмата има три състояния:  **IDLE** - схемата е “в покой”. При постъпване на “1” на входа start, автомата зарежда начални стойности в регистрите fn\_1, fn\_2 и n и преминава в състояние OP.  В fn\_2 и fn\_1 се зареждат респективно f(0) и f(1). Регистърът n се зарежда от входа index с номерът на числото от реда на Фибоначи, което трябва да се изчисли.  **OP** - тук се извършват основните изчисления, посредством трите RT операции:  fn\_1 <- fn\_1+fn\_2  fn\_2 <- fn\_1  n <- n-1  Итерациите приключват когато n стане 1 или когато началната му стойност е била 0 (т.е. изчисление на f(0) ).  **DONE** - изработва сигнал за край на операцията (done\_tick) с продължителнист един такт. |
| --- | --- |

## SystemVerilog code

| **module** fibo\_fsmd**(**  **output** **integer** fibo**,**  **output** **logic** rdy**,** done\_tick**,**  **input** start**,** clock**,** reset**,**  **input** **integer** index  **);**    **enum** **logic** **[**1**:**0**]** **{**IDLE**,** OP**,** DONE**}** state**,** state\_next**;**    **integer** n**,** **n\_next;**  **integer** fn\_2**,** fn\_1**,** **fn\_2\_next,** **fn\_1\_next;**    **assign** fibo **=** fn\_1**;**    **always\_ff** **@(posedge** clock**,** **posedge** reset**)**  **if** **(**reset**)** **begin**  state **<=** IDLE**;**  fn\_2 **<=** 0**;**  fn\_1 **<=** 0**;**  n **<=** 0**;**  **end**  **else** **begin**  state **<=** state\_next**;**  fn\_2 **<=** fn\_2\_next**;**  fn\_1 **<=** fn\_1\_next**;**  n **<=** n\_next**;**  **end;**    **always\_comb** **begin**  // default state  **state\_next = state;**  // default values  ready **=** '0**;**  done\_tick **=** '0**;**  fn\_2\_next **=** fn\_2**;**  fn\_1\_next **=** fn\_1**;**  n\_next **=** n**;**    **case** **(**state**)**  IDLE**:** **begin**  rdy**=** '1**;**  **if** **(**start**)** **begin**  **fn\_2\_next =** 0**;**  **fn\_1\_next =** 1**;**  **n\_next =** index**;**  **state\_next =** OP**;**  **end**  **end**  OP**:** **if** **(**n**==**0**)** **begin**  fn\_1\_next **=** 0**;**  state\_next **=** DONE**;**  **end**  **else** **if** **(**n**==**1**)** state\_next **=** DONE**;**  **else** **begin**  **fn\_1\_next =** fn\_2 **+** fn\_1**;**  **fn\_2\_next =** fn\_1**;**  **n\_next =** n **-** 1**;**  **state\_next =** state**;**  **end**  DONE**:** **begin**  done\_tick **=** '1**;**  state\_next **=** IDLE**;**  **end**  **endcase**  **end**  **endmodule** |  |
| --- | --- |

# Пример - Таймер

<https://github.com/vtchoumatchenko/PCIS/tree/master/examples/timer-switch>

# Пример - Светофар

<https://github.com/vtchoumatchenko/PCIS/tree/master/examples/traffic-light>

# Пример - Най-голям Общ Делител

Реализирана е версия на алгоритъмът на Евклид за намиране на най-голям общ делител.

стъпка 1: Докато A > B, A = A - B

стъпка 2: Ако B==0 алгоритъмит приключва, като НОД се намира в А. Иначе, разменяме местата на A и B и се връщаме на стъпка 1.



**SystemVerilog example**

| **module** gcd**(**  **input** clk**,** reset**,** start**,**  **input** **integer** a\_in**,** b\_in**,**  **output** **integer** gcd\_out**,**  **output** **logic** ready**,** done\_tick**);**    **enum** **logic** **[**1**:**0**]** **{**  IDLE **=** 2'b00**,**  CHECK **=** 2'b01**,**  RUN **=** 2'b10**,**  DONE **=** 2'b11**}** state**,** state\_next**;**    **integer** a\_next**,** b\_next**,** a**,** b**;**    **assign** gcd\_out **=** a**;**    **always\_ff** **@** **(posedge** clk**,** **posedge** reset**)**  **if(**reset**)** **begin**  state **<=** IDLE**;**  a **<=** 0**;**  b **<=** 0**;**  **end**  **else** **begin**  state **<=** state\_next**;**  a **<=** a\_next**;**  b **<=** b\_next**;**  **end;**    **always\_comb** **begin**  state\_next **=** state**;**  ready **=** '0**;**  done\_tick **=** '0**;**  a\_next **=** a**;**  b\_next **=** b**;**    **case(**state**)**  IDLE**:** **begin**  ready **=** '1**;**  **if** **(**start**)** **begin**  **assert(**a\_in **|** b\_in**)** **else** $error**(**"gcd(0,0) is undefined"**);**  a\_next **=** a\_in**;**  b\_next **=** b\_in**;**  state\_next **=** CHECK**;**  **end**  **end**  CHECK**:** **if** **(**a**==**0 **||** b**==**0**)** **begin**  a\_next **=** 0**;**  state\_next **=** DONE**;**  **end**  **else** state\_next **=** RUN**;**  RUN**:** **if** **(**b**==**0**)** state\_next **=** DONE**;**  **else** **if** **(**a **>=** b**)** a\_next **=** a **-** b**;**  **else** **begin**  a\_next **=** b**;**  b\_next **=** a**;**  **end**  DONE**:** **begin**  state\_next **=** IDLE**;**  done\_tick **=** '1**;**  **end**  **endcase**  **end;**  **endmodule** |  |
| --- | --- |

**module** gcd\_test**;**

**logic** clk**,** reset**,** ready**,** done\_tick**,** start**;**

**integer** a\_in**,** b\_in**,** gcd\_out**;**

gcd uut**(.\*);**

**localparam** T **=** 100**;**

**initial** **begin**

clk **=** '0**;**

**forever** **#(**T**/**2**)** clk**=~**clk**;**

**end**

**initial** **begin**

reset **=** '1**;**

**#**T reset **=** '0**;**

check\_gcd**(**6**,**15**,**3**);**

**@(posedge** ready**)**check\_gcd**(**0**,**1**,**0**);**

**@(posedge** ready**)**check\_gcd**(**40**,**60**,**20**);**

**@(posedge** ready**)**check\_gcd**(**1**,**0**,**0**);**

**@(posedge** ready**)**check\_gcd**(**128**,**64**,**64**);**

**@(posedge** ready**)**check\_gcd**(**588**,**1352**,**4**);**

**@(posedge** ready**)**check\_gcd**(**0**,**0**,**0**);**

**#**T $finish**;**

**end**

**task** check\_gcd**(input** **integer** a**,** b**,** expected**);**

a\_in **=** a**;** b\_in **=** b**;**

**#**20 start **=** 1**;**

**#(**T**+**10**)** start **=** 0**;**

**@(posedge** done\_tick**)**

**assert** **(**gcd\_out **==** expected**)** $display**(**"gcd(%0d,%0d)=%0d"**,**a\_in**,**b\_in**,**gcd\_out**);**

**else** $error**(**"a=%0d, b=%0d, expected=%0d, actual=%0d"**,**a\_in**,**b\_in**,**expected**,** gcd\_out**);**

**endtask**

**endmodule**

run all

gcd(6,15)=3

gcd(0,1)=0

gcd(40,60)=20

gcd(1,0)=0

gcd(128,64)=64

gcd(588,1352)=4

Error: gcd(0,0) is undefined