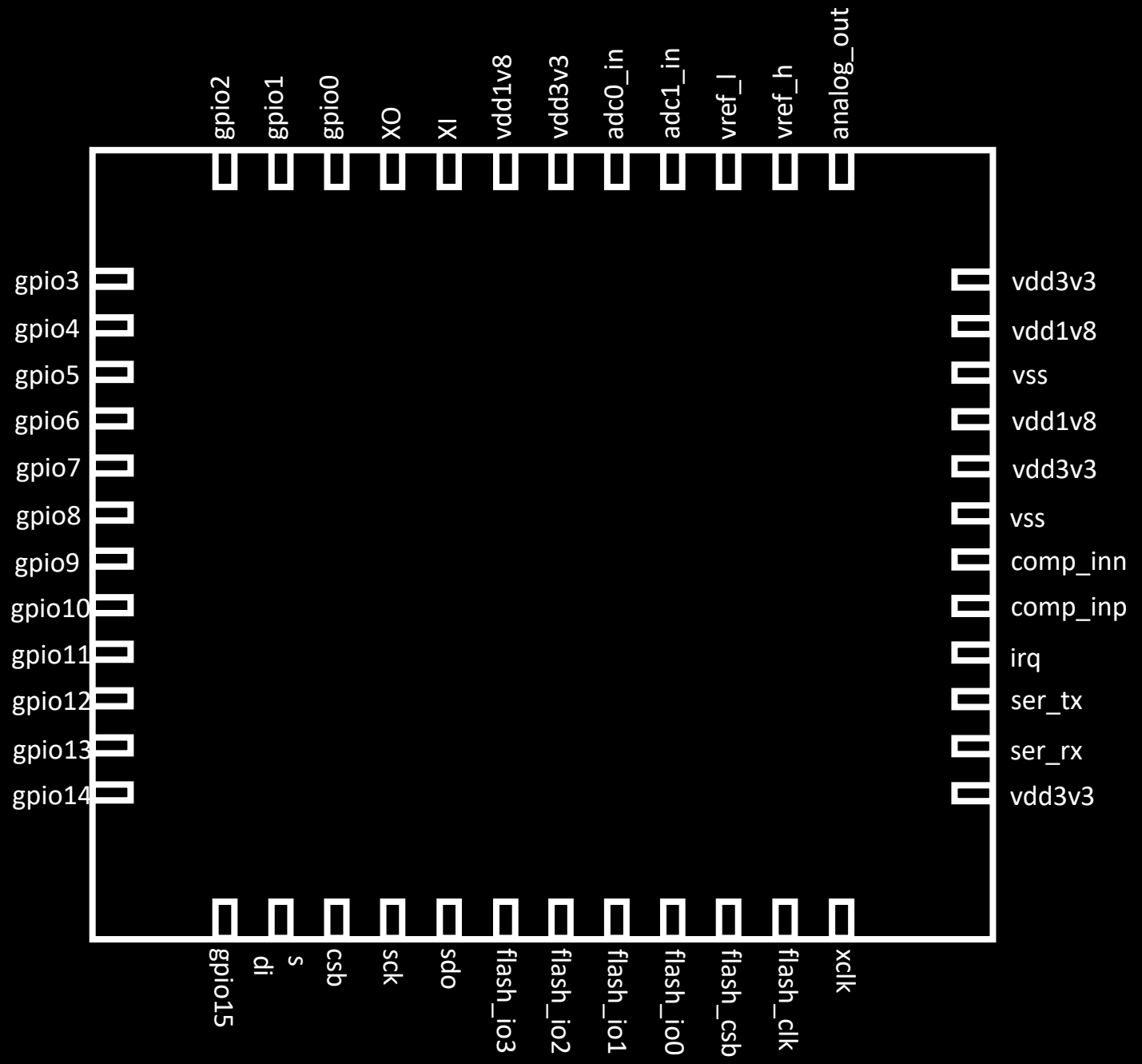
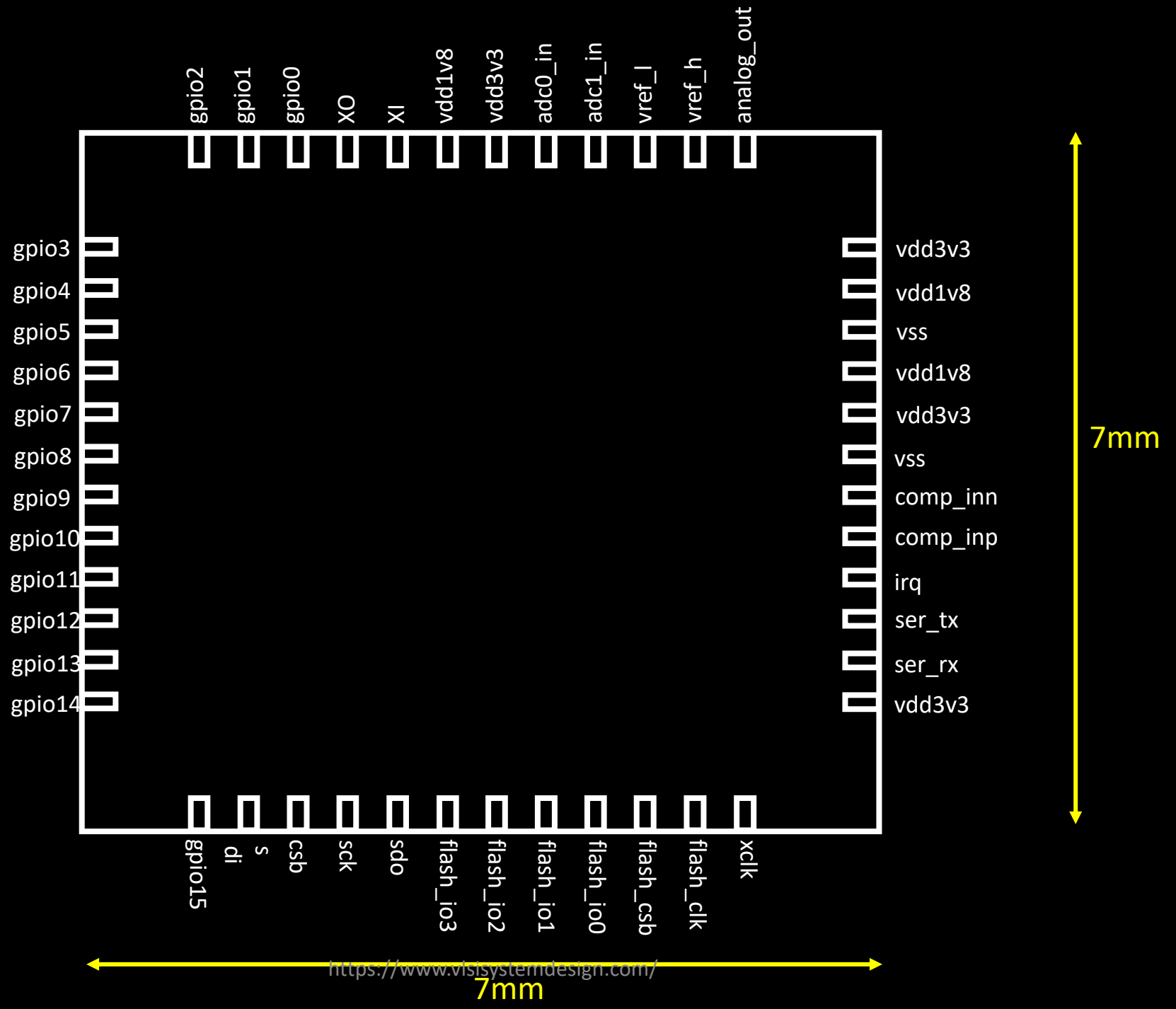
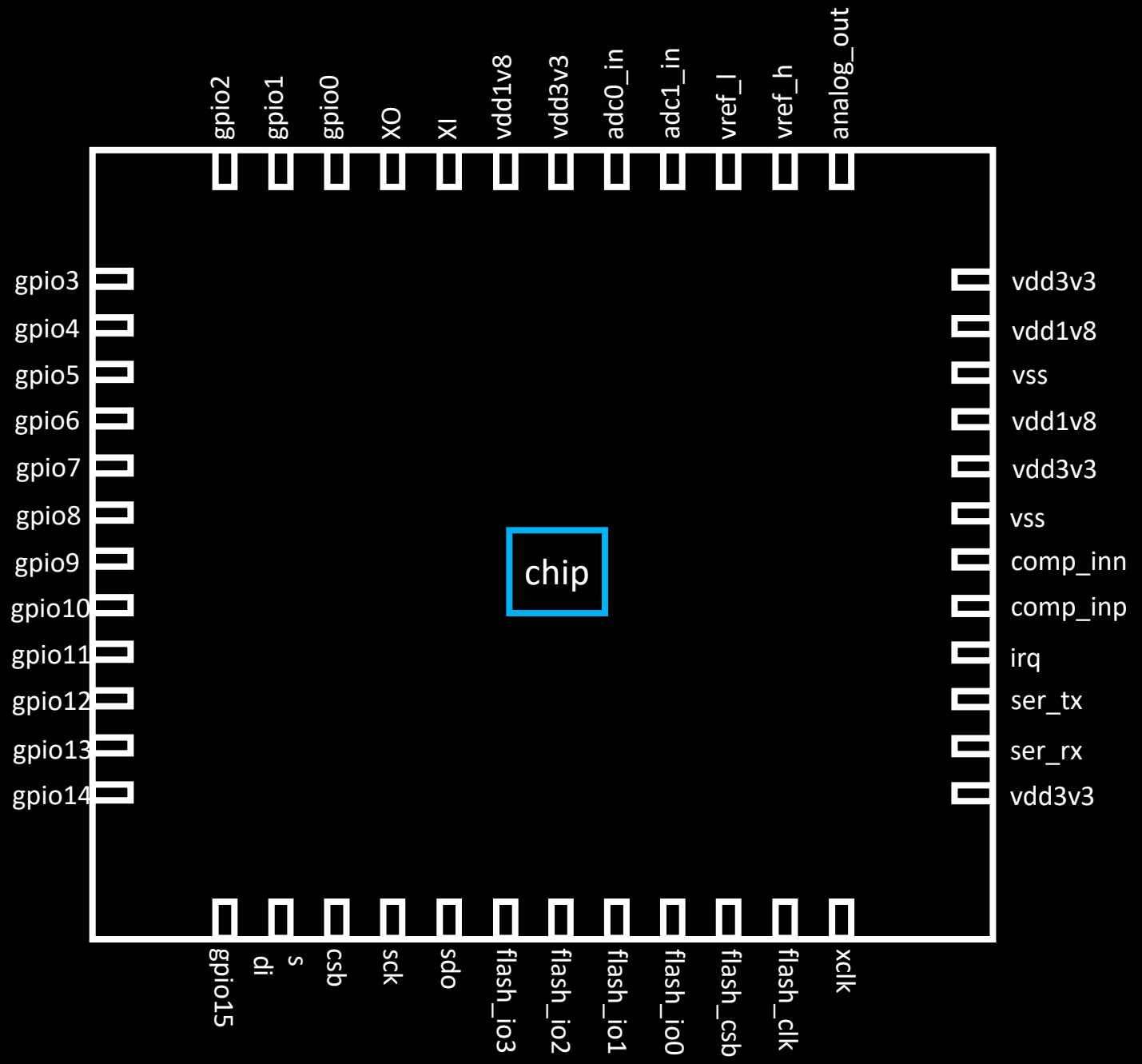
Opensource EDA tool installation

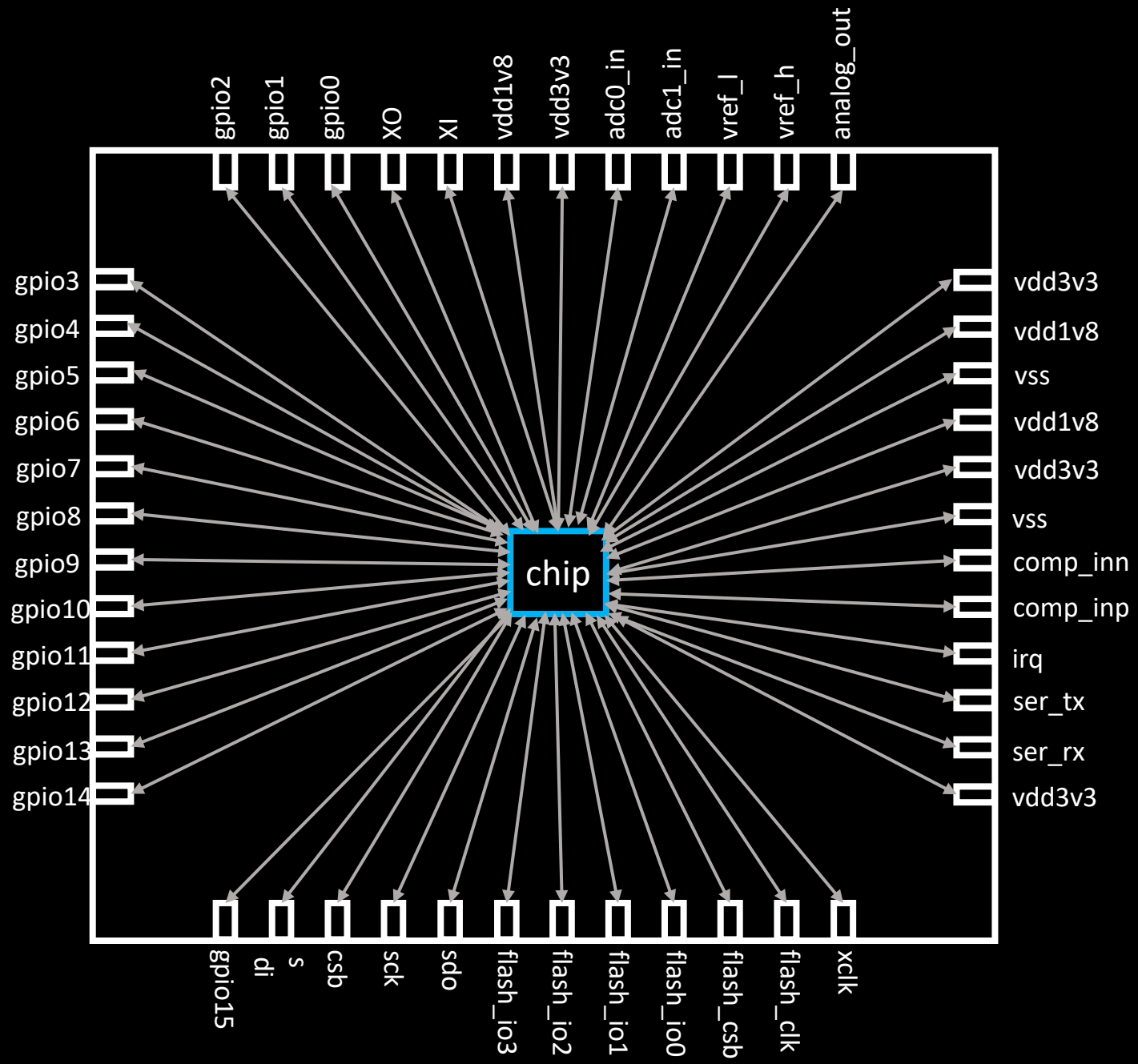
Let's start from 'chip'

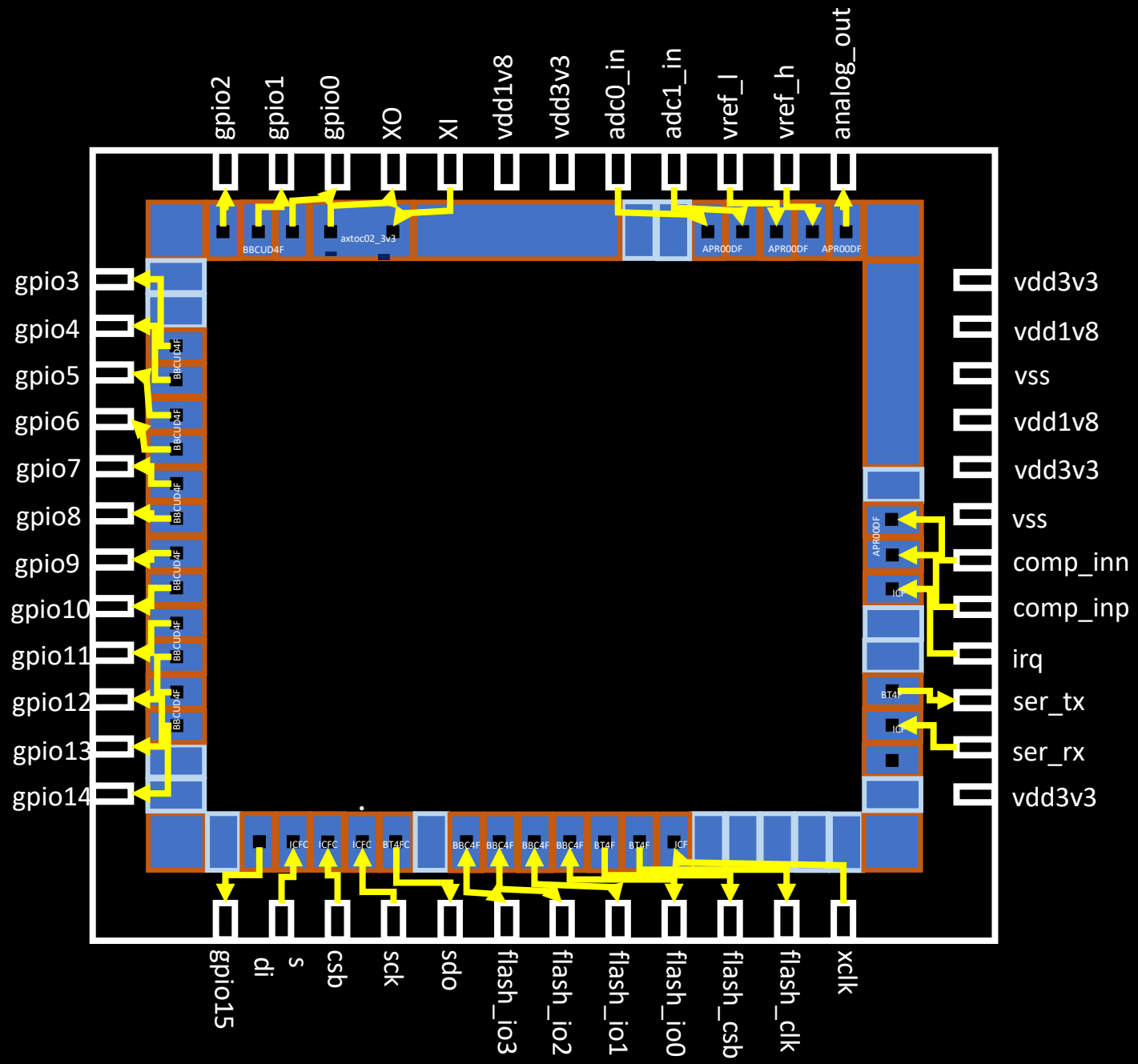


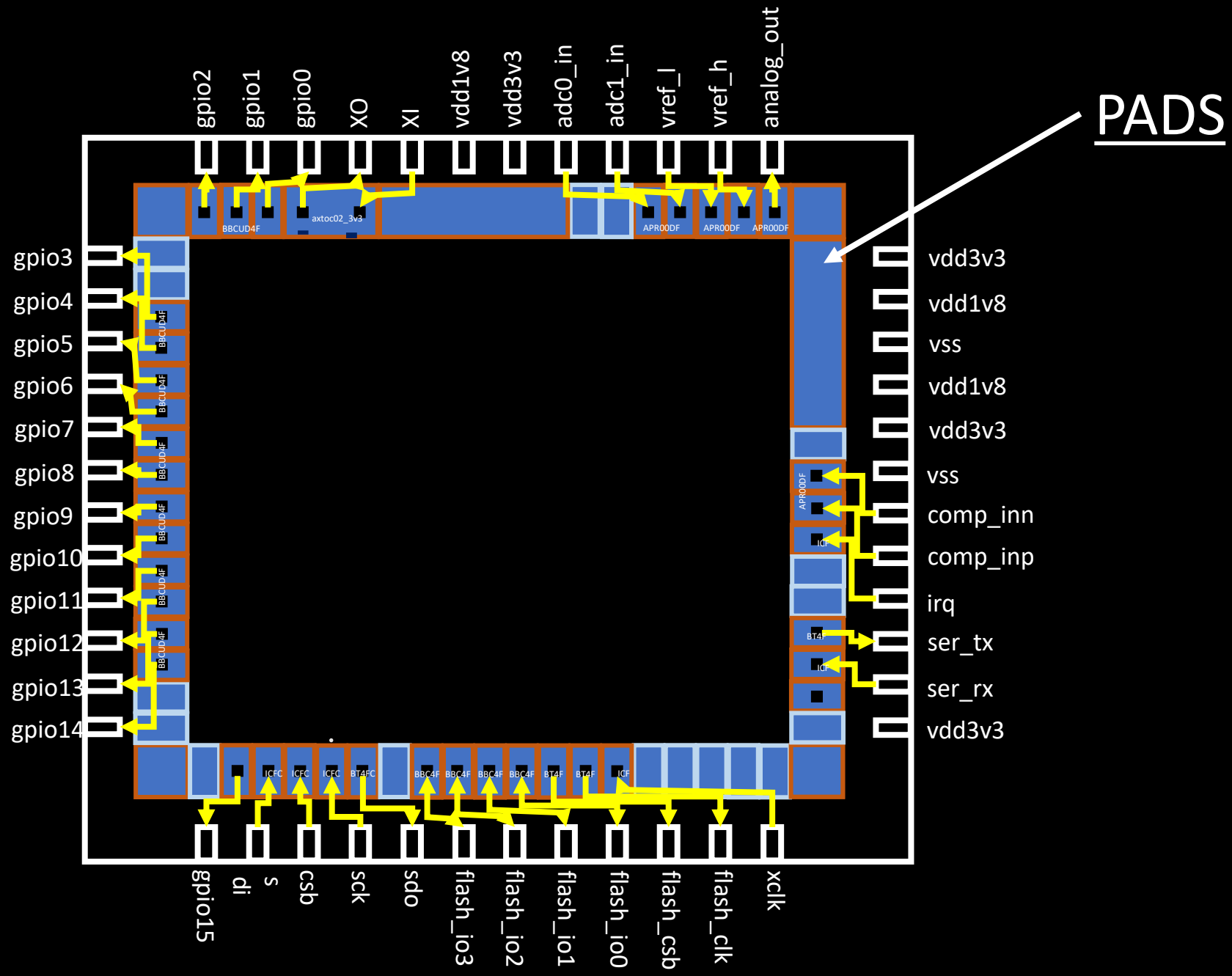
Package : QFN-48
Quad Flat No-leads



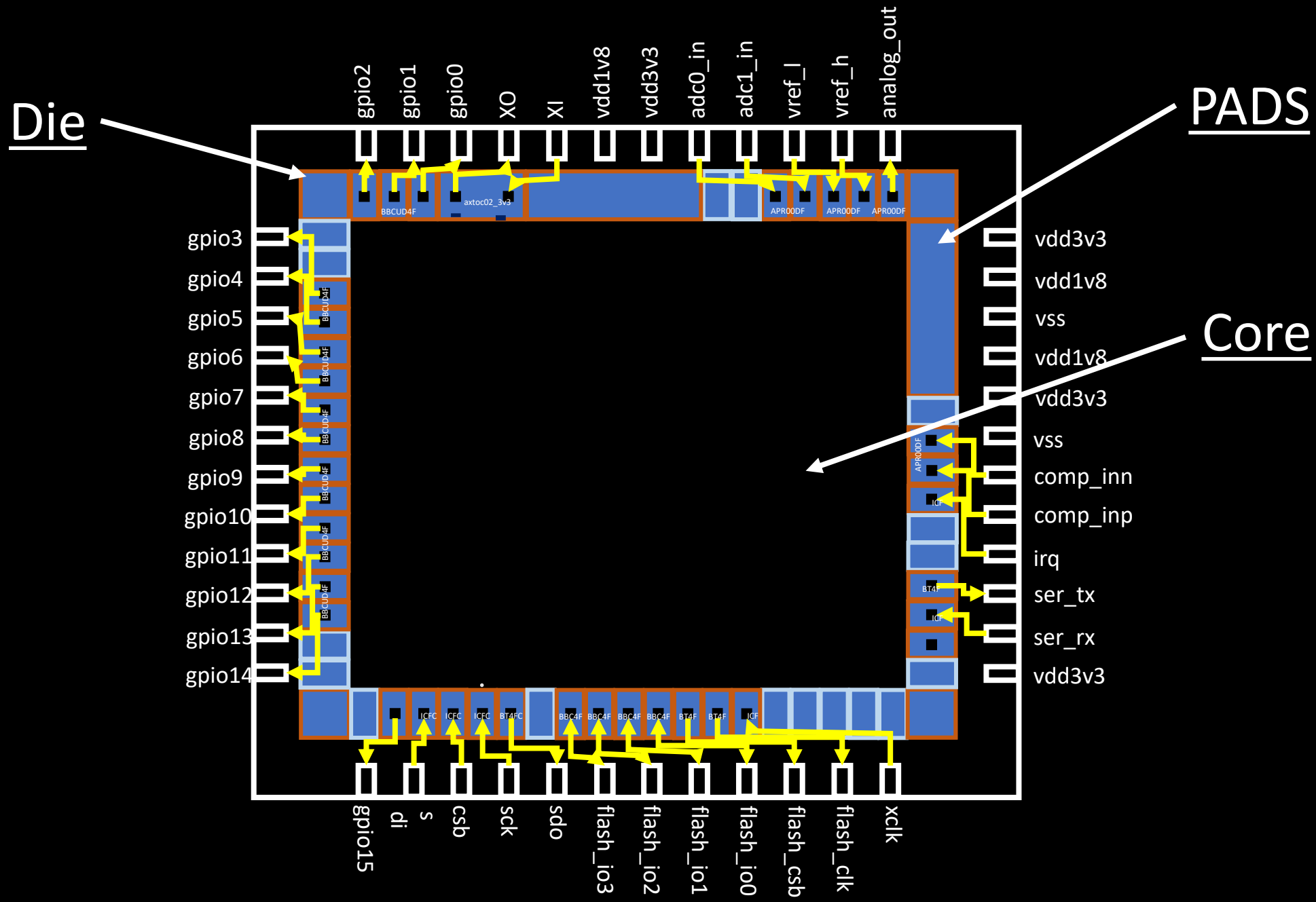



gpio2
gpio1
gpio0
XO
XI
vdd1v8
vdd3v3
adc0_in
adc1_in
vref_l
vref_h
analog_out
gpio3
gpio4
gpio5
gpio6
gpio7
gpio8
gpio9
gpio10
gpio11
gpio12
gpio13
gpio14
chip
vdd3v3
vdd1v8
vss
vdd1v8
vdd3v3
vss
comp_inn
comp_inp
irq
ser_tx
ser_rx
vdd3v3
gpio15
s
di
csb
sck
sdo
flash_io3
flash_io2
flash_io1
flash_io0
flash_csb
flash_clk
xclk

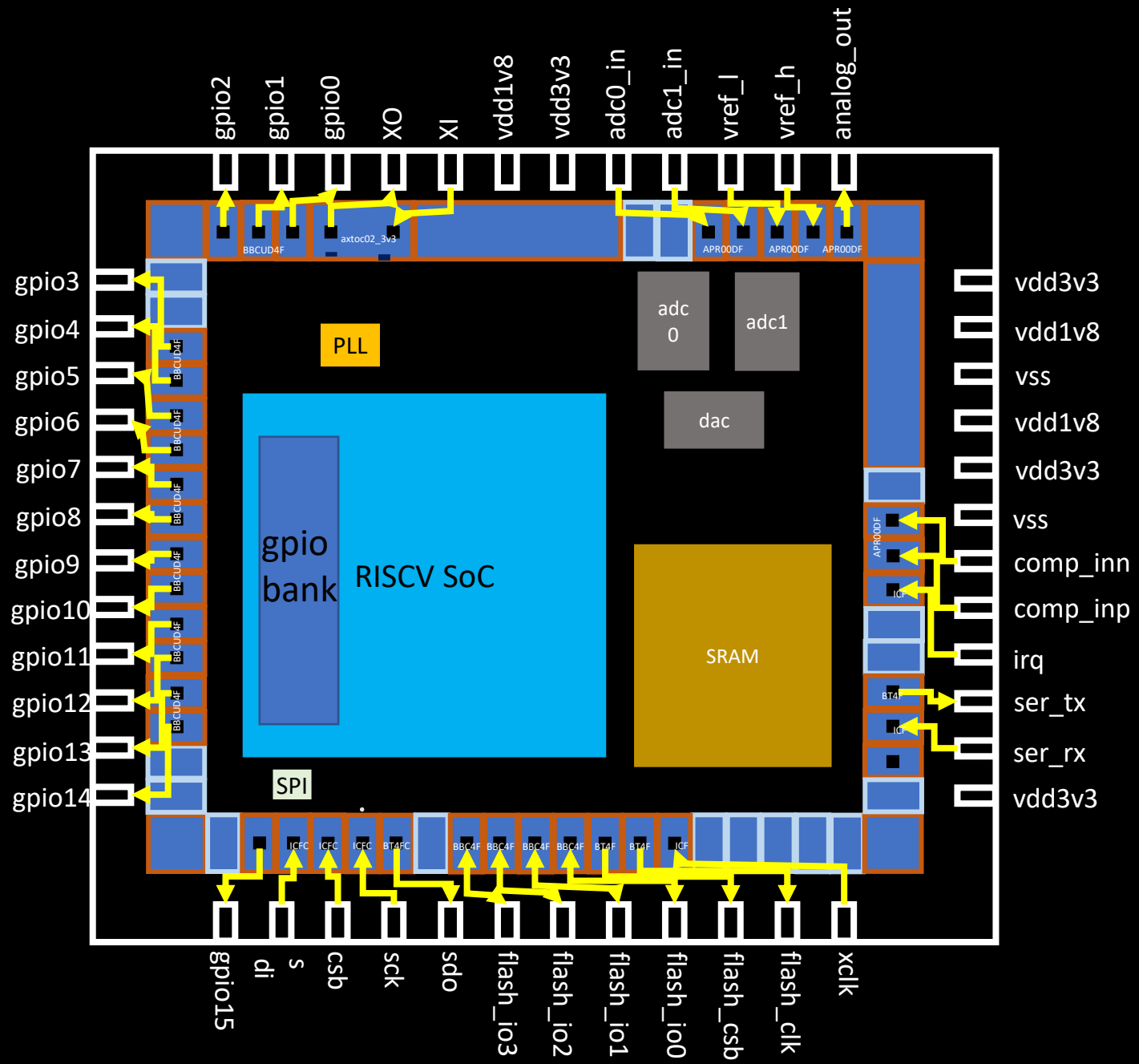




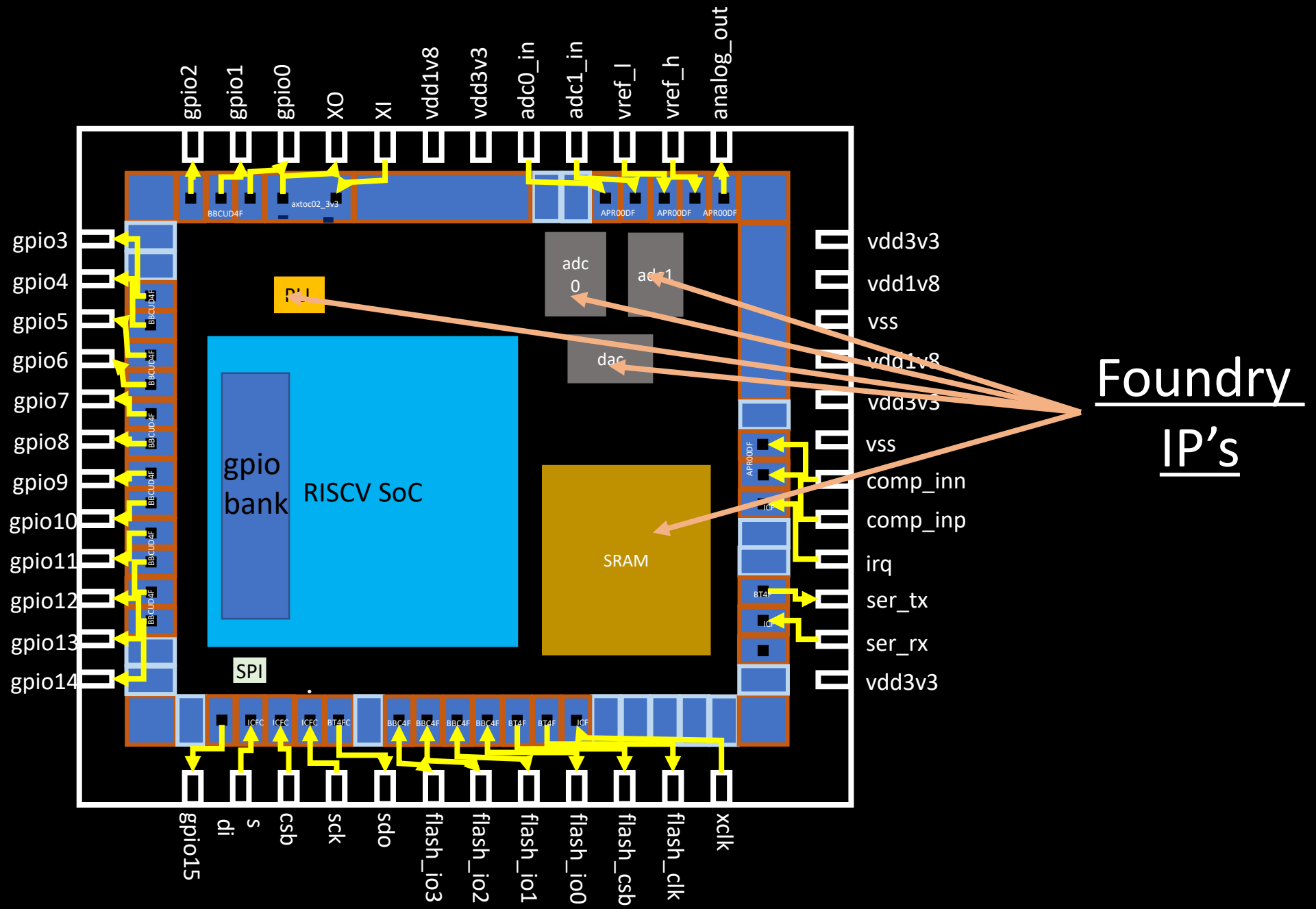
https://www.vlsisystemdesign.com/



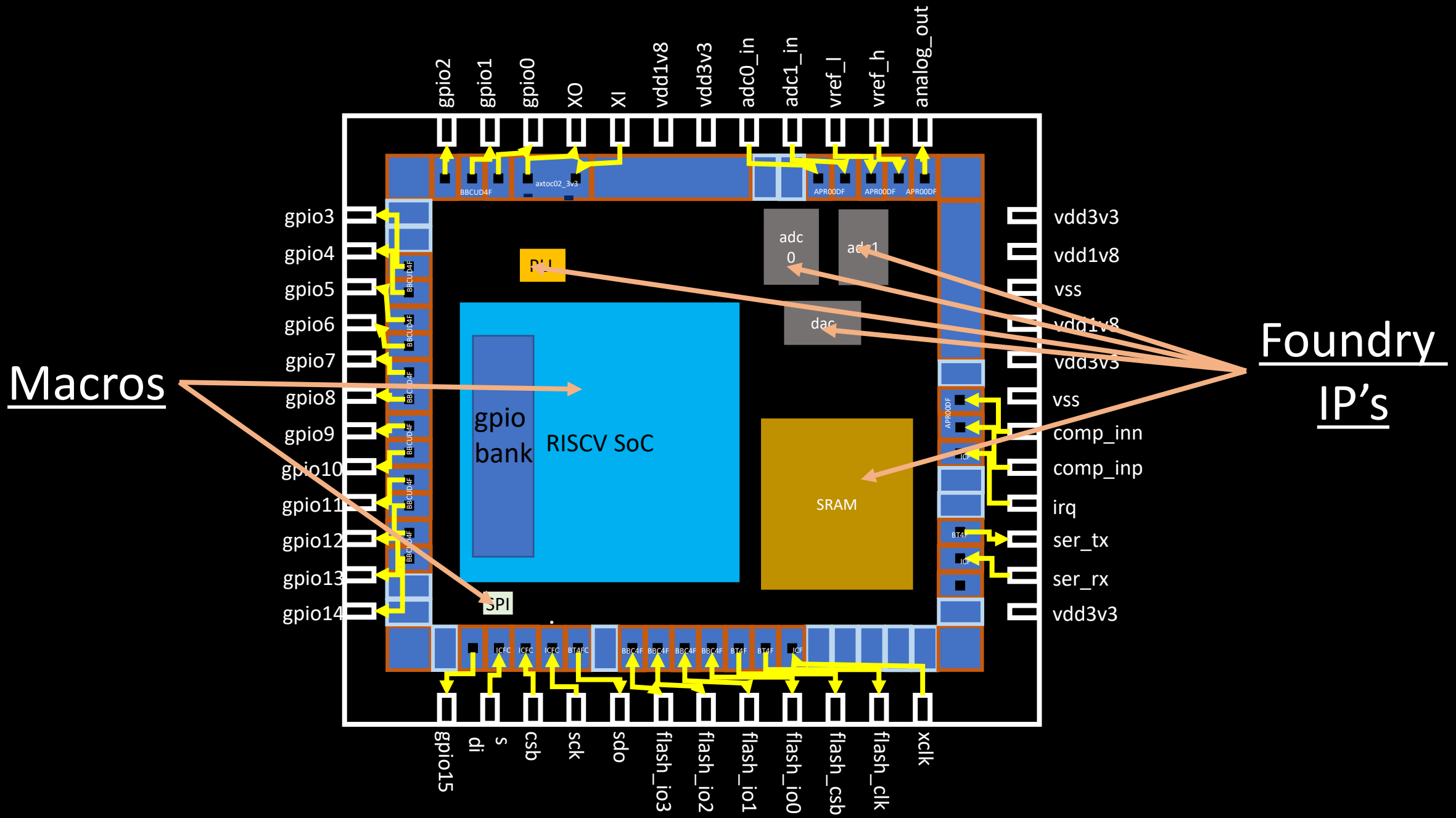
https://www.vlsisystemdesign.com/



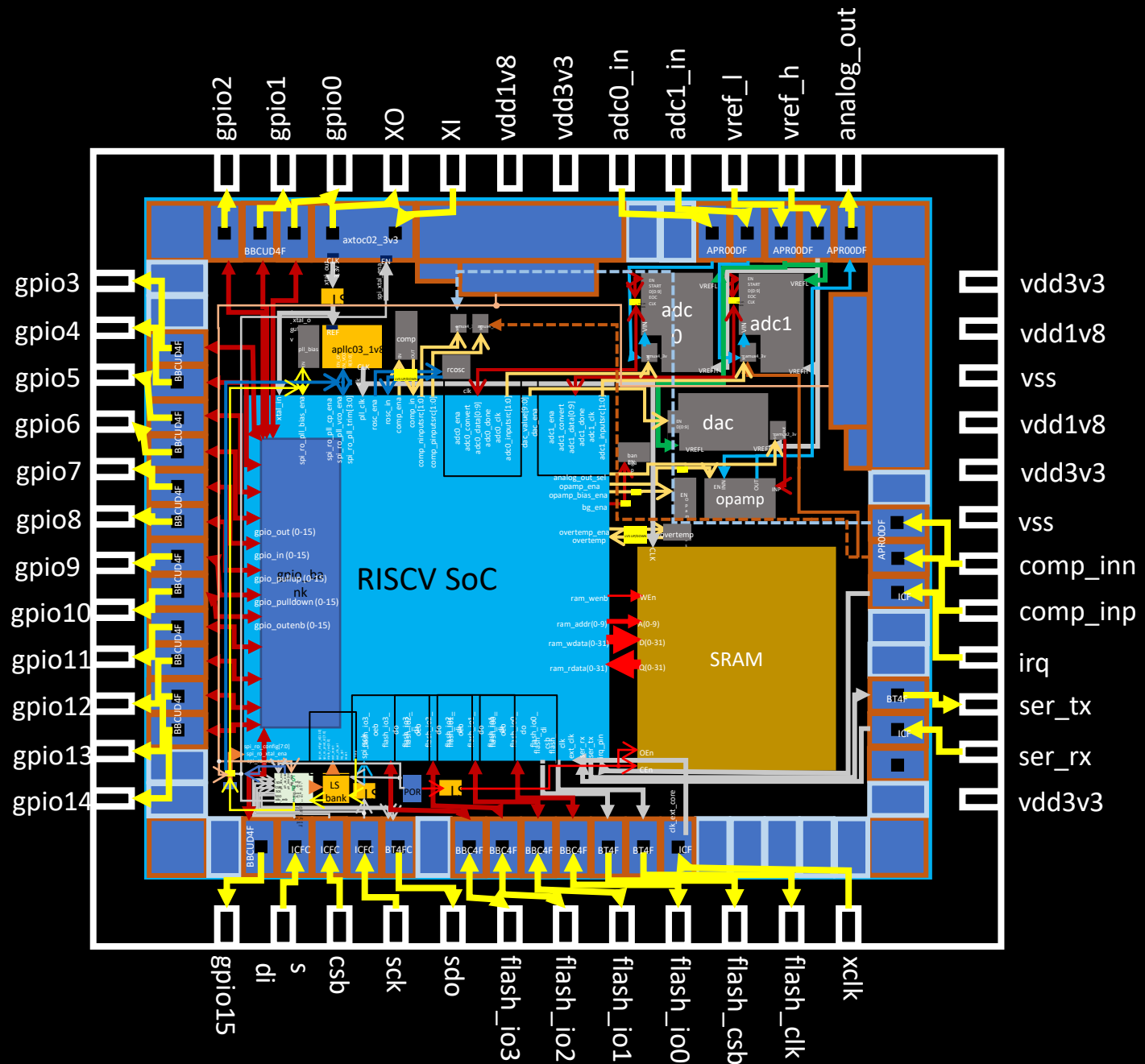
https://www.vlsisystemdesign.com/



https://www.vlsisystemdesign.com/

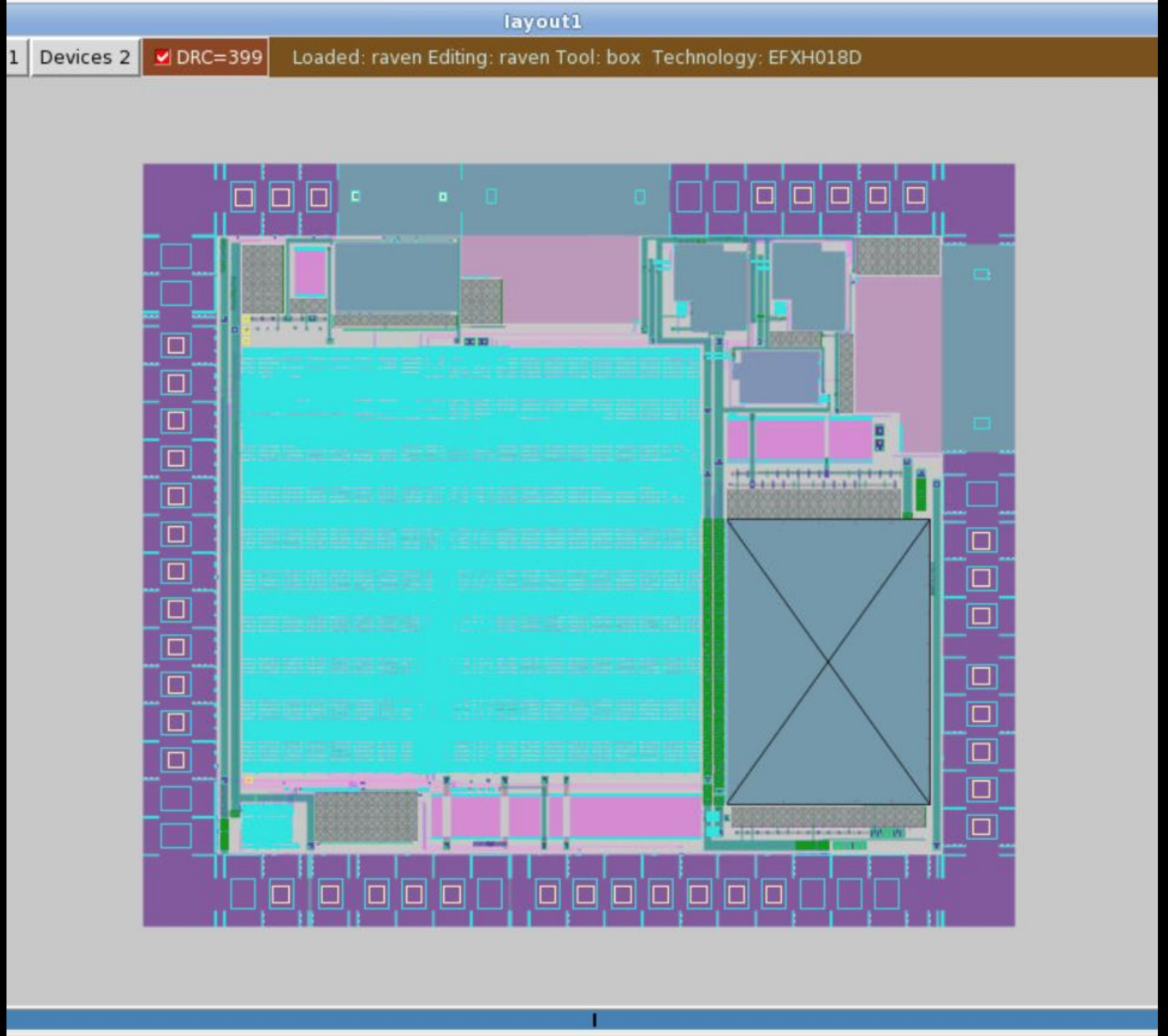


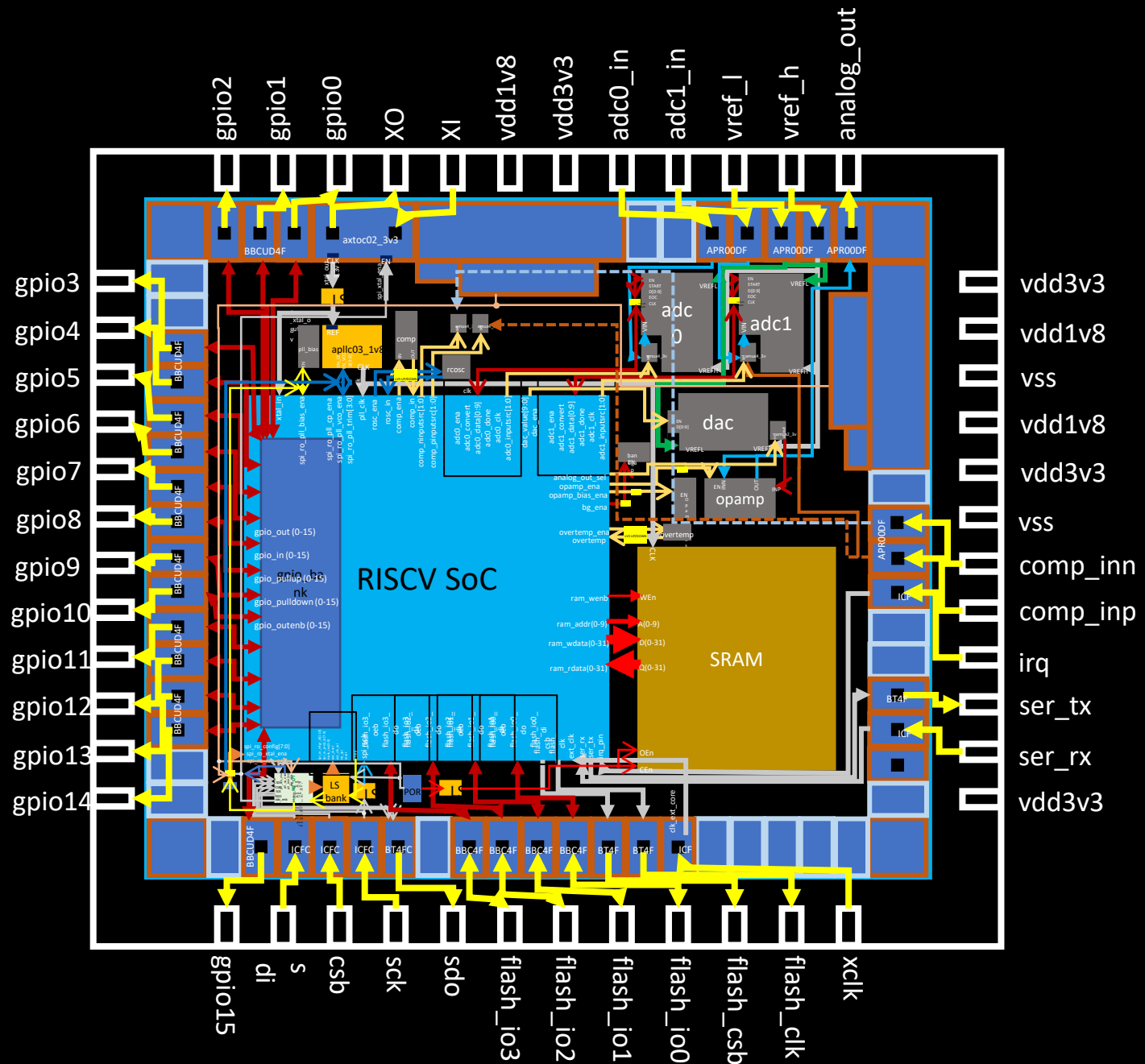
https://www.vlsisystemdesign.com/



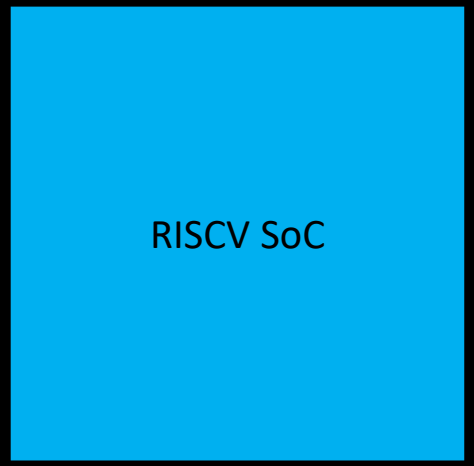
https://www.vlsisystemdesign.com/

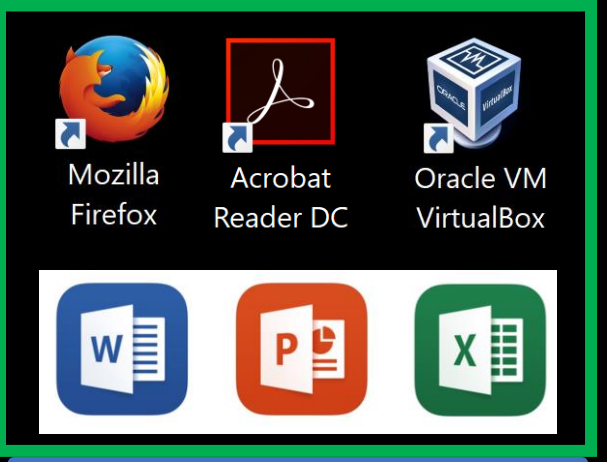
MAGIC LAYOUT VIEW



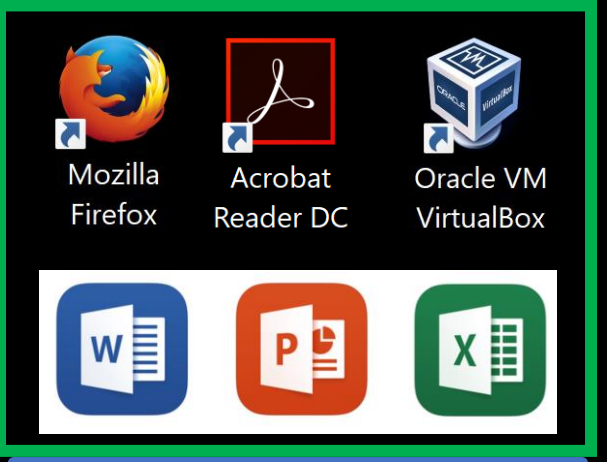


https://www.vlsisystemdesign.com/


RISCV SoC

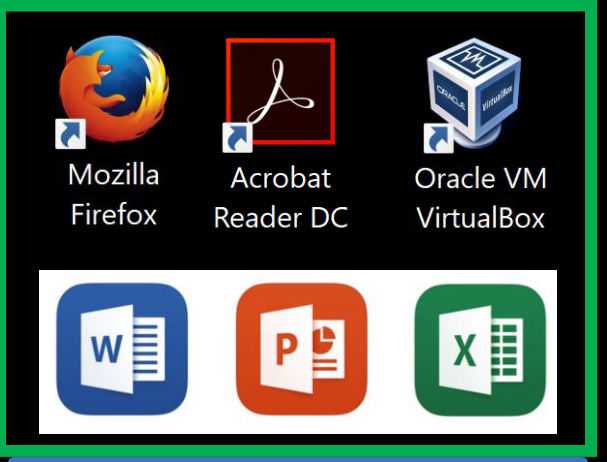


Application Software or Apps



Application Software or Apps

Hardware



Application Software or Apps

?

Hardware

Mozilla Firefox

Acrobat Reader DC

Oracle VM VirtualBox

Application Software or Apps

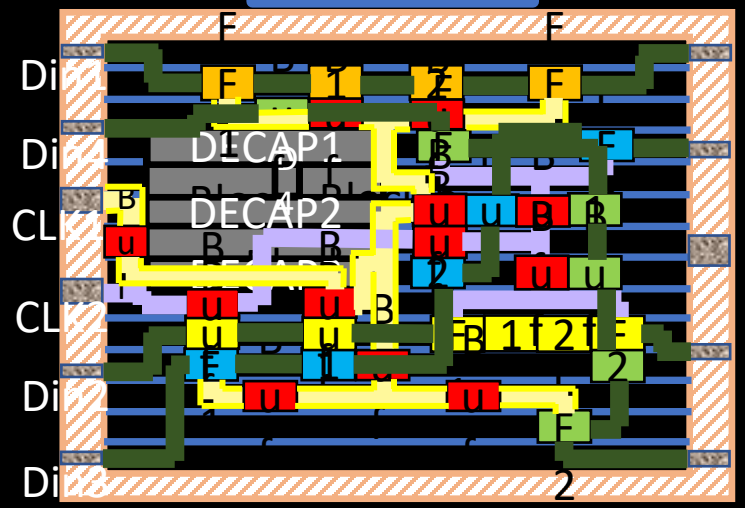
System Software

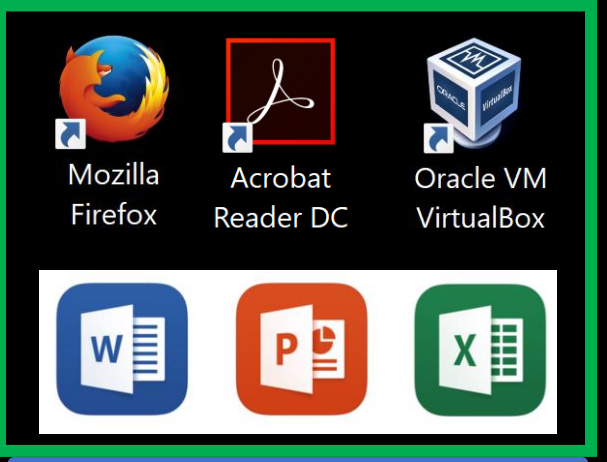
OS

COMPILER

ASSEMBLER

Hardware





Application Software or Apps

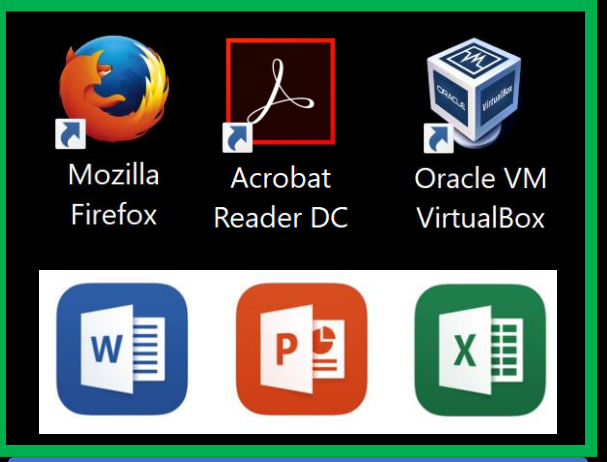
System Software

OS

COMPILER

ASSEMBLER

Hardware



Application Software or Apps

System Software

Windows 7

Linux

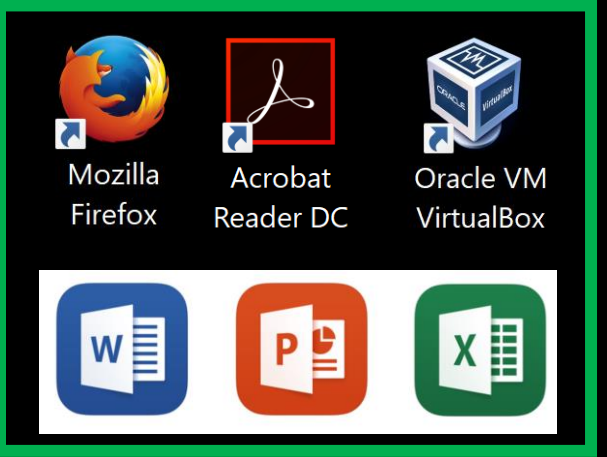
OS

COMPILER

ASSEMBLER

- Handle IO operations
- Allocate memory
- Low level system functions

Hardware



Application Software or Apps

System Software

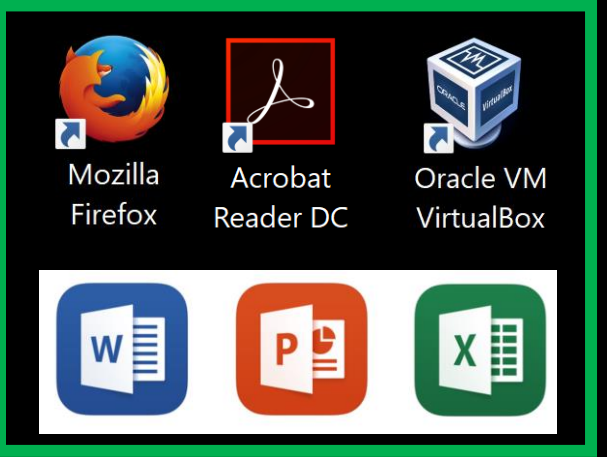
OS

C, C++, VB, Java → COMPILER → Instr1 Instr2 … *.exe file

ASSEMBLER

- Handle IO operations
- Allocate memory
- Low level system functions

Hardware



Application Software or Apps

System Software

OS

C, C++, VB, Java

COMPILER

Instr1
Instr2
...

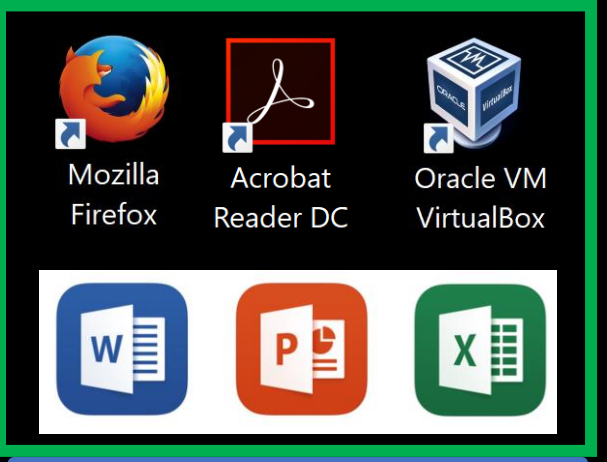
*.exe file

ASSEMBLER

101000110101
101010010010
.....

- Handle IO operations
- Allocate memory
- Low level system functions

Hardware



Application Software or Apps

System Software

OS

C, C++, VB, Java

COMPILER

Instr1
Instr2
...

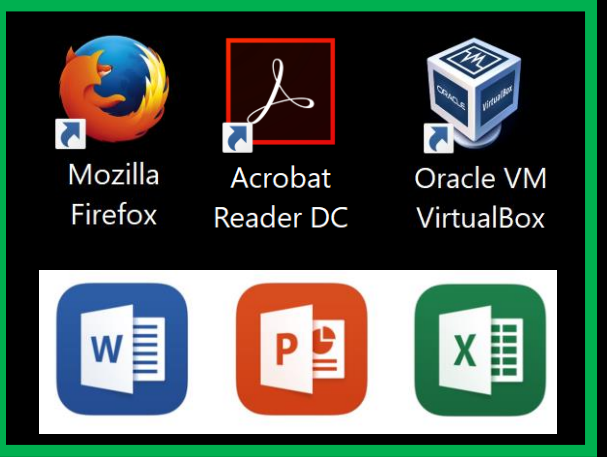
*.exe file

ASSEMBLER

101000110101
101010010010
.....

- Handle IO operations
- Allocate memory
- Low level system functions

Hardware



Application Software or Apps

System Software

OS

C, C++, VB, Java

COMPILER

Instr1
Instr2
...

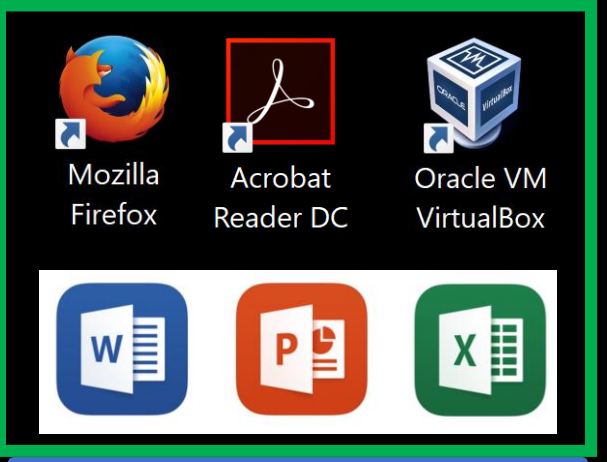
*.exe file

ASSEMBLER

101000 110101
101010010010
.....

- Handle IO operations
- Allocate memory
- Low level system functions

Hardware



Application Software or Apps

System Software

OS

C, C++, VB, Java

COMPILER

Instr1
Instr2
...

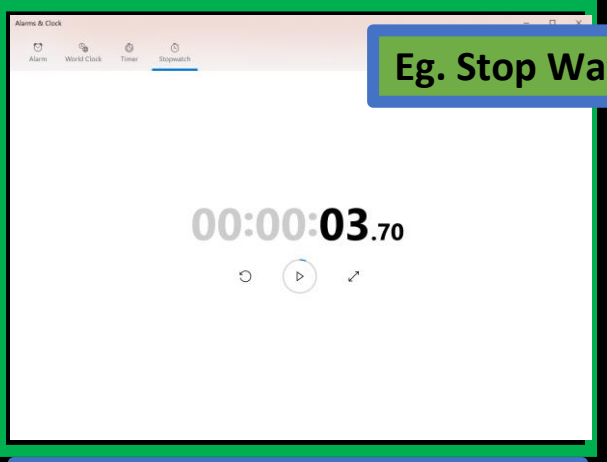
*.exe file

ASSEMBLER

101000110101
101010010010
.....

- Handle IO operations
- Allocate memory
- Low level system functions

Hardware



Eg. Stop Watch app

Application Software or Apps

System Software

OS

C, C++, VB, Java

COMPILER

Instr1
Instr2
...

*.exe file

ASSEMBLER

101000110101
101010010010
.....

- Handle IO operations
- Allocate memory
- Low level system functions

Hardware

Eg. Stop Watch app

Appli

```
#include <stdio.h>
#include <time.h> //for sleep() function

int main()
{
    int hour, minute, second;
    hour=minute=second=0;
    while(1)
    {
        //clear output screen
        system("clear");
        //print time in HH : MM : SS format
        printf("%02d : %02d : %02d ",hour,minute,second);
         //clear output buffer in gcc
        fflush(stdout);
         //increase second
        second++;
        //update hour, minute and second
        if(second==60){
            minute+=1;
            second=0;
        }
        if(minute==60){
            hour+=1;
            minute=0;
        }
        if(hour==24){
            hour=0;
            minute=0;
            second=0;
        }
        sleep(1);    //wait till 1 second
    }
    return 0;
}
```

Compiler input, a ‘c’ function for stopwatch

System Software

Windows 7

Linux

OS

COMPILER

Instr1
Instr2
...

*.exe file

ASSEMBLER

101000110101
101010010010
.....

- Handle IO operations
- Allocate memory
- Low level system functions

Hardware

Eg. Stop Watch app

Appli

System Software

OS

- Handle IO operations
- Allocate memory
- Low level system functions

```c
#include <stdio.h>
#include <time.h> //for sleep() function

int main()
{
    int hour, minute, second;
    hour=minute=second=0;
    while(1)
    {
        //clear output screen
        system("clear");
        //print time in HH : MM : SS format
        printf("%02d : %02d : %02d ",hour,minute,second);
         //clear output buffer in gcc
        fflush(stdout);
         //increase second
        second++;
        //update hour, minute and second
        if(second==60){
            minute+=1;
            second=0;
        }
        if(minute==60){
            hour+=1;
            minute=0;
        }
        if(hour==24){
            hour=0;
            minute=0;
            second=0;
        }
        sleep(1);    //wait till 1 second
    }
    return 0;
}
```

Compiler input, a ‘c’ function for stopwatch

COMPILER

Instr1
Instr2
...

*.exe file

```
0000000000000001 <main>:
   1:   715d                addi    sp,
   3:   e45e                sd      s7,8(sp)
   5:   00000bb7            lui     s7,0x0
   9:   000b8513            mv      a0,s7
   d:   e486                sd      ra,72(sp)
   f:   e0a2                sd      s0,64(sp)
  11:   fc26                sd      s1,56(sp)
  13:   f84a                sd      s2,48(sp)
  15:   f052                sd      s4,32(sp)
  17:   ec56                sd      s5,24(sp)
  19:   e85a                sd      s6,16(sp)
  1b:   e062                sd      s8,0(sp)
  1d:   f44e                sd      s3,40(sp)
  1f:   00000b37            lui     s6,0x0
  23:   00000097            auipc   ra,0x0
  27:   000080e7            jalr    ra
  2b:   4681                li      a3,0
  2d:   4601                li      a2,0
  2f:   4581                li      a1,0
  31:   000b0513            mv      a0,s6
  35:   00000097            auipc   ra,0x0
  39:   000080e7            jalr    ra
  3d:   00000ab7            lui     s5,0x0
```

Compiler and assembler output, A RISC-V assembly language program

Hardware

Eg. Stop Watch app

Appli

```
#include <stdio.h>
#include <time.h> //for sleep() function

int main()
{
    int hour, minute, second;
    hour=minute=second=0;
    while(1)
    {
        //clear output screen
        system("clear");
        //print time in HH : MM : SS format
        printf("%02d : %02d : %02d ",hour,minute,second);
         //clear output buffer in gcc
        fflush(stdout);
         //increase second
        second++;
        //update hour, minute and second
        if(second==60){
            minute+=1;
            second=0;
        }
        if(minute==60){
            hour+=1;
            minute=0;
        }
        if(hour==24){
            hour=0;
            minute=0;
            second=0;
        }
        sleep(1);    //wait till 1 second
    }
    return 0;
}
```

Compiler input, a ‘c’ function for stopwatch

System Software

OS

- Handle IO operations
- Allocate memory
- Low level system functions

COMPILER

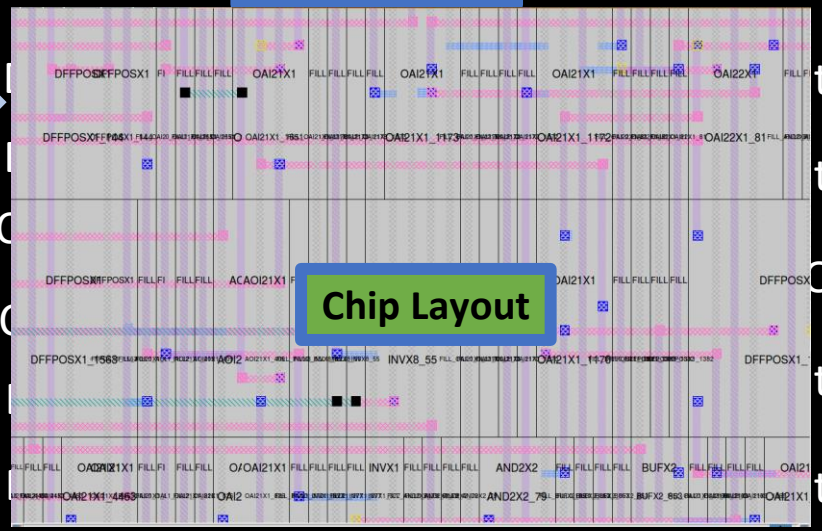
Instr1
Instr2
...

*.exe file

```
0000000000000001 <main>:
   1:   715d                addi    sp,
   3:   e45e                sd      s7,8(sp)
   5:   00000bb7            lui     s7,0x0
   9:   000b8513            mv      a0,s7
   d:   e486                sd      ra,72(sp)
   f:   e0a2                sd      s0,64(sp)
  11:   fc26                sd      s1,56(sp)
  13:   f84a                sd      s2,48(sp)
  15:   f052                sd      s4,32(sp)
  17:   ec56                sd      s5,24(sp)
  19:   e85a                sd      s6,16(sp)
  1b:   e062                sd      s8,0(sp)
  1d:   f44e                sd      s3,40(sp)
  1f:   00000b37            lui     s6,0x0
  23:   00000097            auipc   ra,0x0
  27:   000080e7            jalr    ra
  2b:   4681                li      a3,0
  2d:   4601                li      a2,0
  2f:   4581                li      a1,0
  31:   000b0513            mv      a0,s6
  35:   00000097            auipc   ra,0x0
  39:   000080e7            jalr    ra
  3d:   00000ab7            lui     s5,0x0
```

Compiler and assembler output, A RISC-V assembly language program

Hardware



Eg. Stop Watch app

Appli

```c
#include <stdio.h>
#include <time.h> //for sleep() function

int main()
{
    int hour, minute, second;
    hour=minute=second=0;
    while(1)
    {
        //clear output screen
        system("clear");
        //print time in HH : MM : SS format
        printf("%02d : %02d : %02d ",hour,minute,second);
         //clear output buffer in gcc
        fflush(stdout);
         //increase second
        second++;
        //update hour, minute and second
        if(second==60){
            minute+=1;
            second=0;
        }
        if(minute==60){
            hour+=1;
            minute=0;
        }
        if(hour==24){
            hour=0;
            minute=0;
            second=0;
        }
        sleep(1);    //wait till 1 second
    }
    return 0;
}
```

Compiler input, a ‘c’ function for stopwatch

System Software

OS

COMPILER

Instr1
Instr2
...

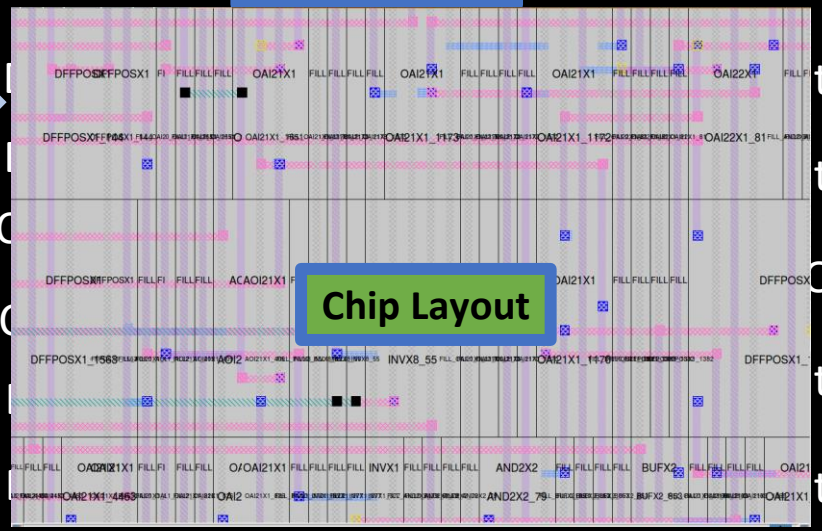
*.exe file

- Handle IO operations
- Allocate memory
- Low level system functions

```
0000000000000001 <main>:
   1:   715d                addi    sp,
   3:   e45e                sd      s7,8(sp
   5:   00000bb7            lui     s7,0x0
   9:   000b8513            mv      a0,s7
   d:   e486                sd      ra,72(sp)
   f:   e0a2                sd      s0,64(sp)
  11:   fc26                sd      s1,56(sp)
  13:   f84a                sd      s2,48(sp)
  15:   f052                sd      s4,32(sp)
  17:   ec56                sd      s5,24(sp)
  19:   e85a                sd      s6,16(sp)
  1b:   e062                sd      s8,0(sp)
  1d:   f44e                sd      s3,40(sp)
  1f:   00000b37            lui     s6,0x0
  23:   00000097            auipc   ra,0x0
  27:   000080e7            jalr    ra
  2b:   4681                li      a3,0
  2d:   4601                li      a2,0
  2f:   4581                li      a1,0
  31:   000b0513            mv      a0,s6
  35:   00000097            auipc   ra,0x0
  39:   000080e7            jalr    ra
  3d:   00000ab7            lui     s5,0x
```

Compiler and assembler output, A RISC-V assembly language program

Hardware



```c
#include <stdio.h>
#include <time.h> //for sleep() function

int main()
{
    int hour, minute, second;
    hour=minute=second=0;
    while(1)
    {
        //clear output screen
        system("clear");
        //print time in HH : MM : SS format
        printf("%02d : %02d : %02d ",hour,minute,second);
         //clear output buffer in gcc
        fflush(stdout);
         //increase second
        second++;
        //update hour, minute and second
        if(second==60){
            minute+=1;
            second=0;
        }
        if(minute==60){
            hour+=1;
            minute=0;
        }
        if(hour==24){
            hour=0;
            minute=0;
            second=0;
        }
        sleep(1);   //wait till 1 second
    }
    return 0;
}
```

Compiler input, a ‘c’ function for stopwatch

Compiler and assembler output, A RISC-V assembly language program

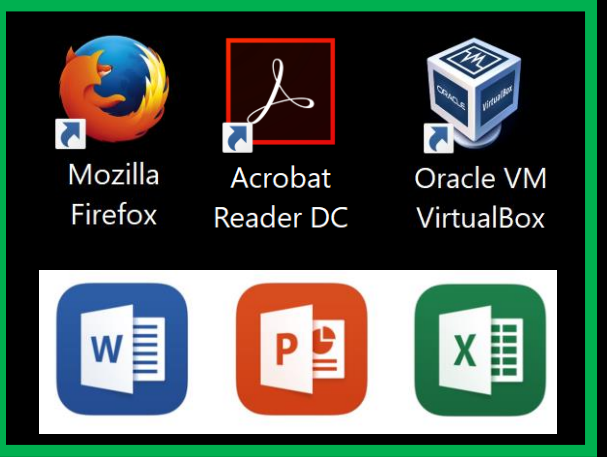
```
0000000000000001 <main>:
   1:   715d                addi    sp,
   3:   e45e                sd      s7,8(sp)
   5:   00000bb7            lui     s7,0x0
   9:   000b8513            mv      a0,s7
   d:   e486                sd      ra,72(sp)
   f:   e0a2                sd      s0,64(sp)
  11:   fc26                sd      s1,56(sp)
  13:   f84a                sd      s2,48(sp)
  15:   f052                sd      s4,32(sp)
  17:   ec56                sd      s5,24(sp)
  19:   e85a                sd      s6,16(sp)
  1b:   e062                sd      s8,0(sp)
  1d:   f44e                sd      s3,40(sp)
  1f:   00000b37            lui     s6,0x0
  23:   00000097            auipc   ra,0x0
  27:   000080e7            jalr    ra
  2b:   4681                li      a3,0
  2d:   4601                li      a2,0
  2f:   4581                li      a1,0
  31:   000b0513            mv      a0,s6
  35:   00000097            auipc   ra,0x0
  39:   000080e7            jalr    ra
  3d:   00000ab7            lui     s5,0x0
```

```c
#include <stdio.h>
#include <time.h> //for sleep() function

int main()
{
    int hour, minute, second;
    hour=minute=second=0;
    while(1)
    {
        //clear output screen
        system("clear");
        //print time in HH : MM : SS format
        printf("%02d : %02d : %02d ",hour,minute,second);
         //clear output buffer in gcc
        fflush(stdout);
         //increase second
        second++;
        //update hour, minute and second
        if(second==60){
            minute+=1;
            second=0;
        }
        if(minute==60){
            hour+=1;
            minute=0;
        }
        if(hour==24){
            hour=0;
            minute=0;
            second=0;
        }
        sleep(1);    //wait till 1 second
    }
    return 0;
}
```

Compiler input, a ‘c’ function for stopwatch

```
0000000000000001 <main>:
   1:   715d                addi    sp,sp,-80
   3:   e45e                sd      s7,8(sp)
   5:   00000bb7            lui     s7,0x0
   9:   000b8513            mv      a0,s7
   d:   e486                sd      ra,72(sp)
   f:   e0a2                sd      s0,64(sp)
  11:   fc26                sd      s1,56(sp)
  13:   f84a                sd      s2,48(sp)
  15:   f052                sd      s4,32(sp)
  17:   ec56                sd      s5,24(sp)
  19:   e85a                sd      s6,16(sp)
  1b:   e062                sd      s8,0(sp)
  1d:   f44e                sd      s3,40(sp)
  1f:   00000b37            lui     s6,0x0
  23:   00000097            auipc   ra,0x0
  27:   000080e7            jalr    ra
  2b:   4681                li      a3,0
  2d:   4601                li      a2,0
  2f:   4581                li      a1,0
  31:   000b0513            mv      a0,s6
  35:   00000097            auipc   ra,0x0
  39:   000080e7            jalr    ra
  3d:   00000ab7            lui     s5,0x0
  41:   000ab783            ld      a5,0(s5) # 0 <_impure_ptr>
  45:   4405                li      s0,1
  47:   4901                li      s2,0
  49:   6b88                ld      a0,16(a5)
  4b:   4481                li      s1,0
  4d:   4c61                li      s8,24
  4f:   00000097            auipc   ra,0x0
  53:   000080e7            jalr    ra
  57:   03c00a13            li      s4,60

000000000000005b <.L2>:
  5b:   4505                li      a0,1
  5d:   0014099b            addiw   s3,s0,1
  61:   05848963            beq     s1,s8,166 <.L5+0x75>

0000000000000065 <.L9>:
  65:   00000097            auipc   ra,0x0
  69:   000080e7            jalr    ra
  6d:   000b8513            mv      a0,s7
  71:   00000097            auipc   ra,0x0
  75:   000080e7            jalr    ra
  79:   86a2                mv      a3,s0
  7b:   864a                mv      a2,s2
  7d:   85a6                mv      a1,s1
  7f:   000b0513            mv      a0,s6
  83:   00000097            auipc   ra,0x0
  87:   000080e7            jalr    ra
  8b:   000ab783            ld      a5,0(s5)
  8f:   4401                li      s0,0
  91:   6b88                ld      a0,16(a5)
  93:   00000097            auipc   ra,0x0
  97:   000080e7            jalr    ra
  9b:   05498b63            beq     s3,s4,1e2 <.L5+0xf1>
  9f:   844e                mv      s0,s3

00000000000000a1 <.L6>:
  a1:   fb491de3            bne     s2,s4,b6 <.L4+0x3>
  a5:   2485                addiw   s1,s1,1
  a7:   4901                li      s2,0
  a9:   4505                li      a0,1
  ab:   0014099b            addiw   s3,s0,1
  af:   fb849be3            bne     s1,s8,ca <.L4+0x17>

00000000000000b3 <.L4>:
  b3:   4505                li      a0,1
  b5:   00000097            auipc   ra,0x0
  b9:   000080e7            jalr    ra
  bd:   000b8513            mv      a0,s7
  c1:   00000097            auipc   ra,0x0
  c5:   000080e7            jalr    ra
  c9:   4681                li      a3,0
  cb:   4601                li      a2,0
  cd:   4581                li      a1,0
  cf:   000b0513            mv      a0,s6
  d3:   00000097            auipc   ra,0x0
  d7:   000080e7            jalr    ra
  db:   000ab783            ld      a5,0(s5)
  df:   4405                li      s0,1
  e1:   4901                li      s2,0
  e3:   6b88                ld      a0,16(a5)
  e5:   4481                li      s1,0
  e7:   00000097            auipc   ra,0x0
  eb:   000080e7            jalr    ra
  ef:   b7b5                j       b6 <.L4+0x3>
00000000000000f1 <.L5>:
  f1:   2905                addiw   s2,s2,1
  f3:   b77d                j       142 <.L5+0x51>
        ...
```

Compiler and assembler output, A RISC-V assembly language program



Application Software or Apps

System Software

OS

Windows 7

Linux

C, C++, VB, Java

COMPILER

Instr1
Instr2
...

*.exe file

ASSEMBLER

Abstract interface

101000110101
101010010010
.....

Hardware

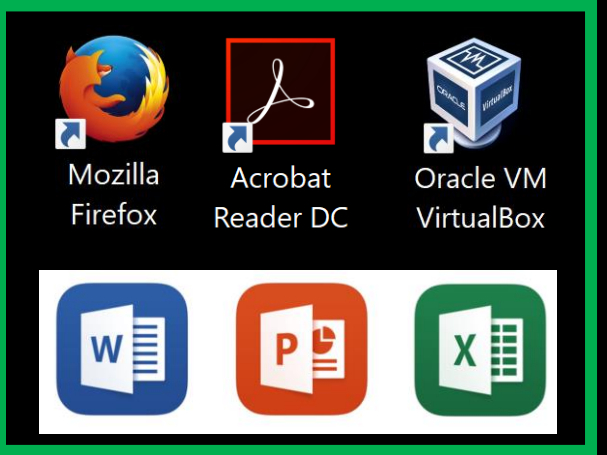
Dout1

Dout3

Clk Out

Dout2

Dout4



Application Software or Apps

System Software

OS

Windows 7

Linux

C, C++, VB, Java

COMPILER

Instr1
Instr2
...

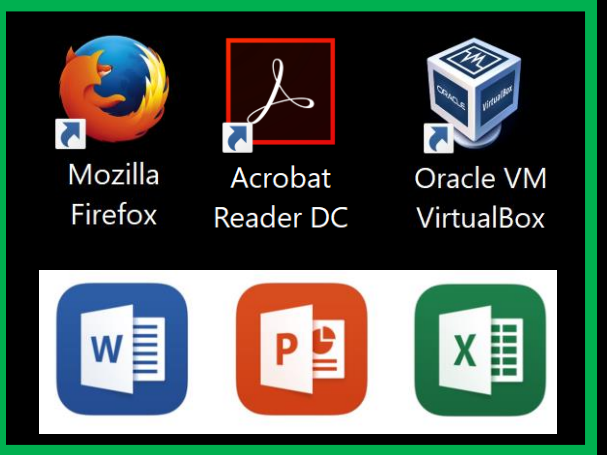
*.exe file

ASSEMBLER

Abstract interface
Instruction Set Architecture

101000110101
101010010010
.....

Hardware



Application Software or Apps

System Software

OS

Windows 7

C, C++, VB, Java

COMPILER

Instr1
Instr2
...

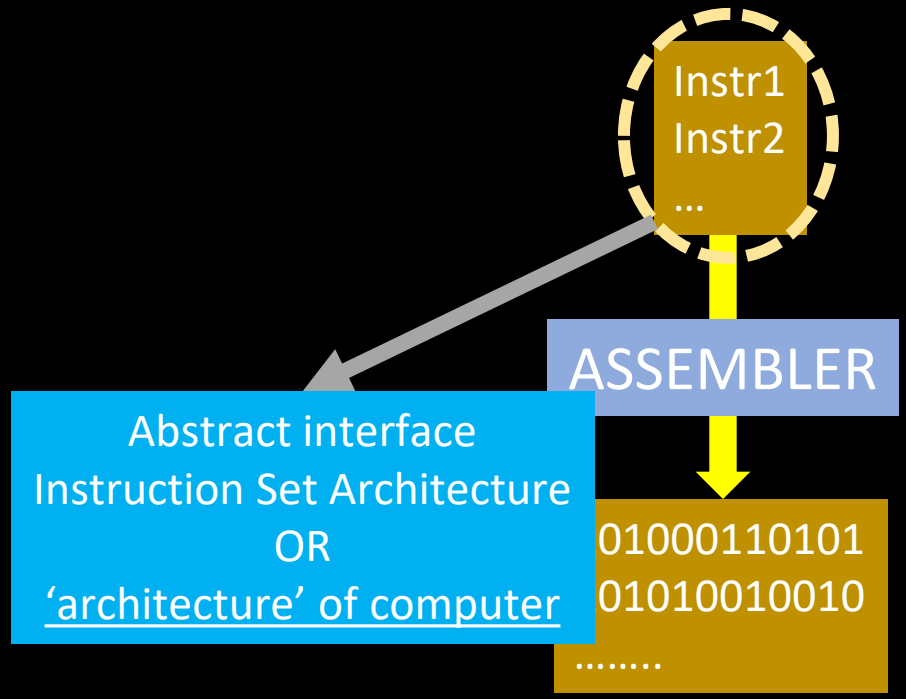
*.exe file

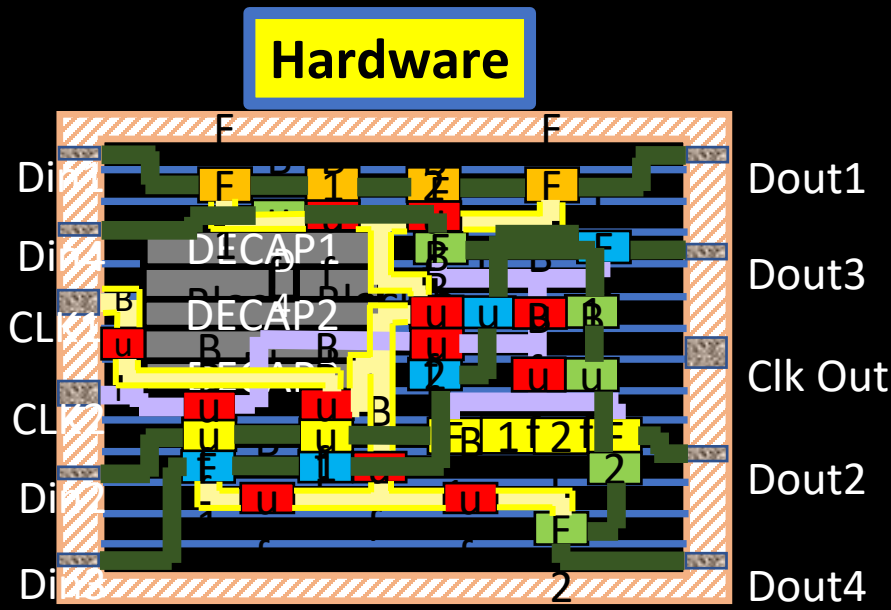
ASSEMBLER

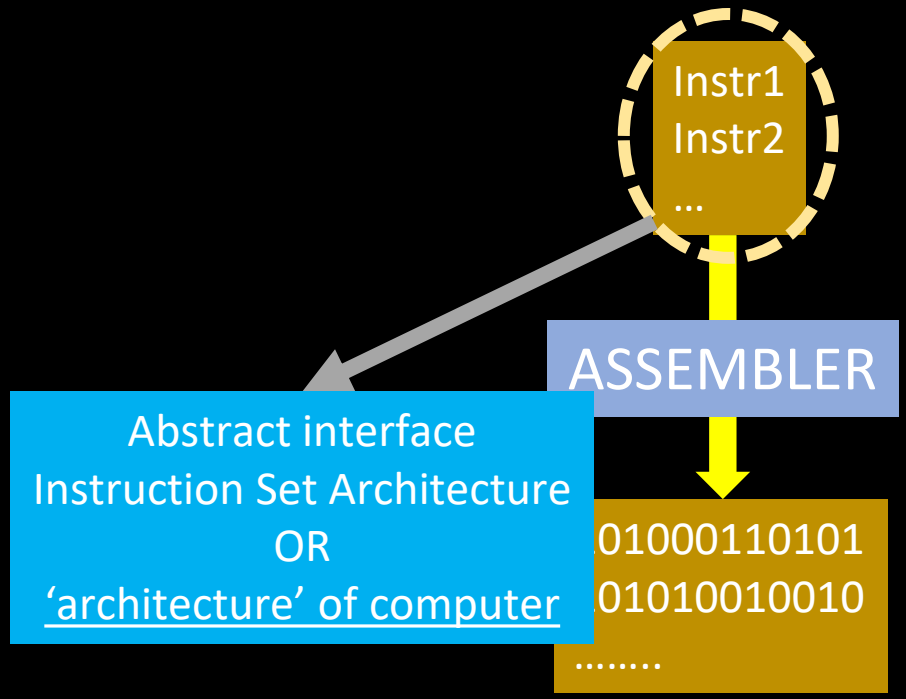
Abstract interface
Instruction Set Architecture
OR
'architecture' of computer

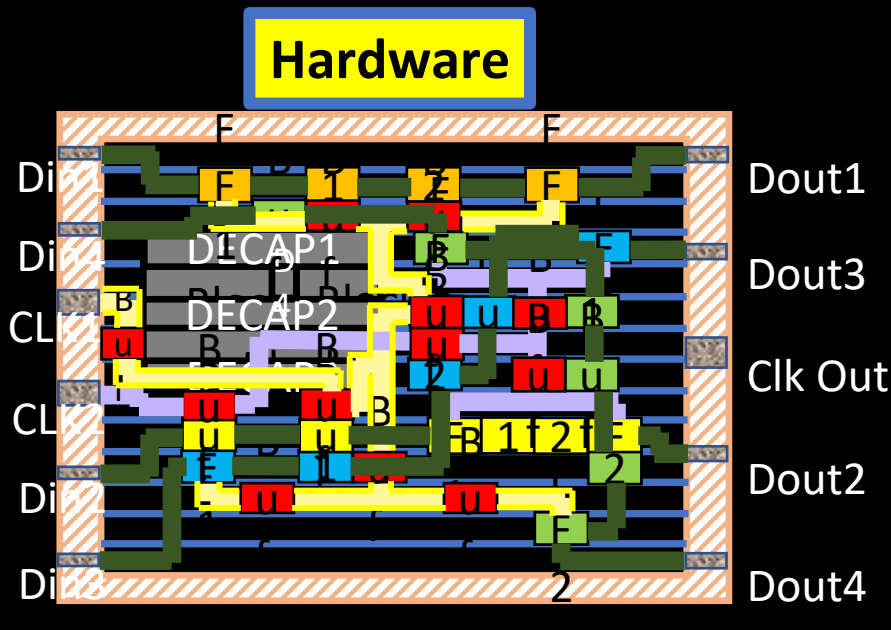
01000110101
01010010010
.....

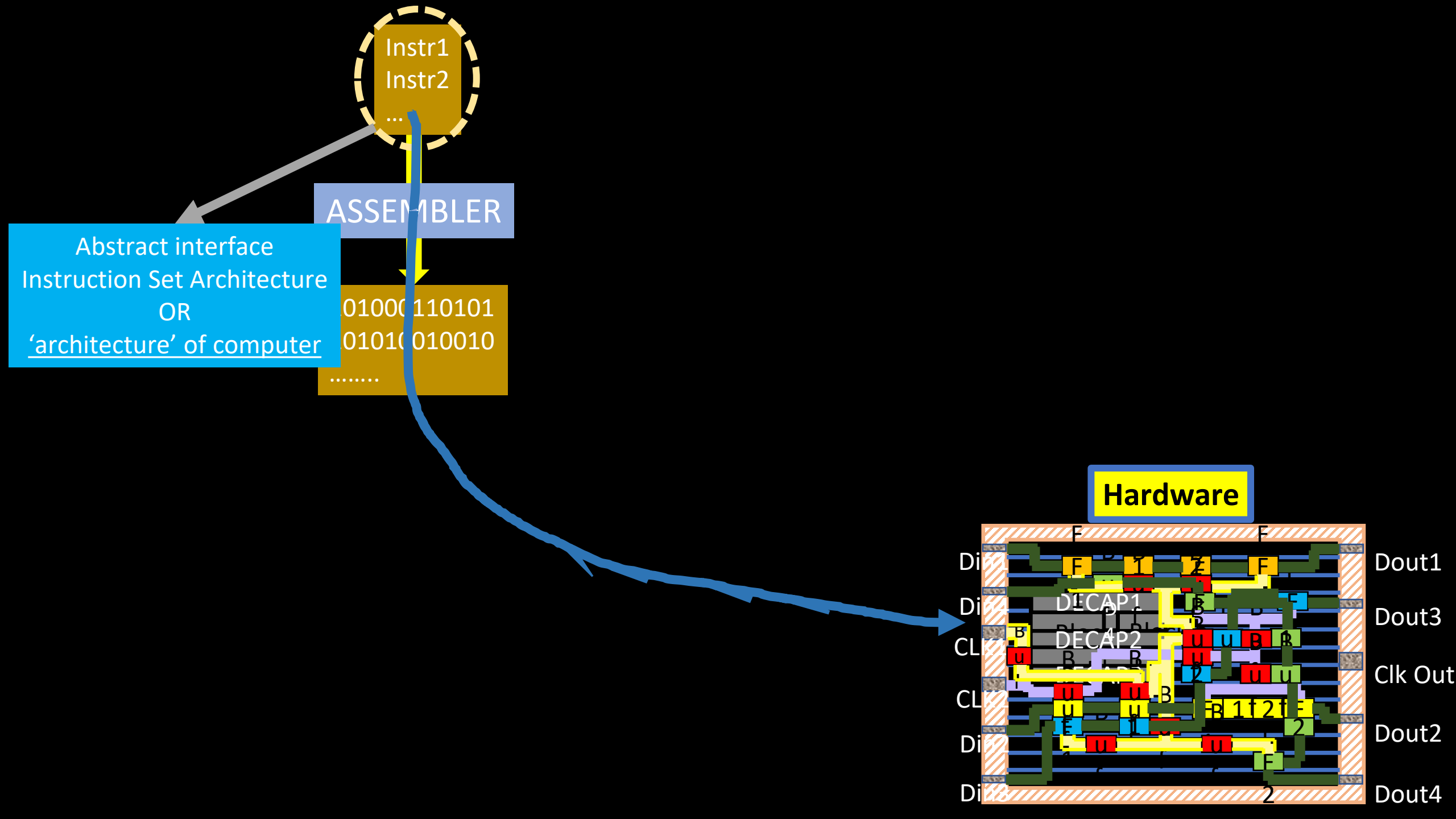
Hardware

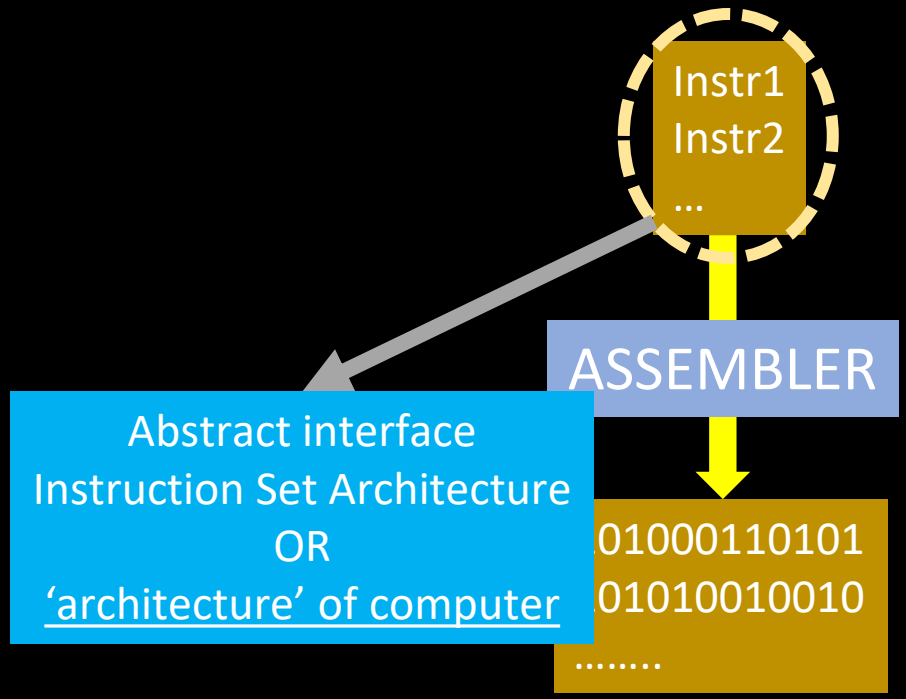

Instr1
Instr2
...
ASSEMBLER
Abstract interface
Instruction Set Architecture
OR
'architecture' of computer
01000110101
01010010010
.....

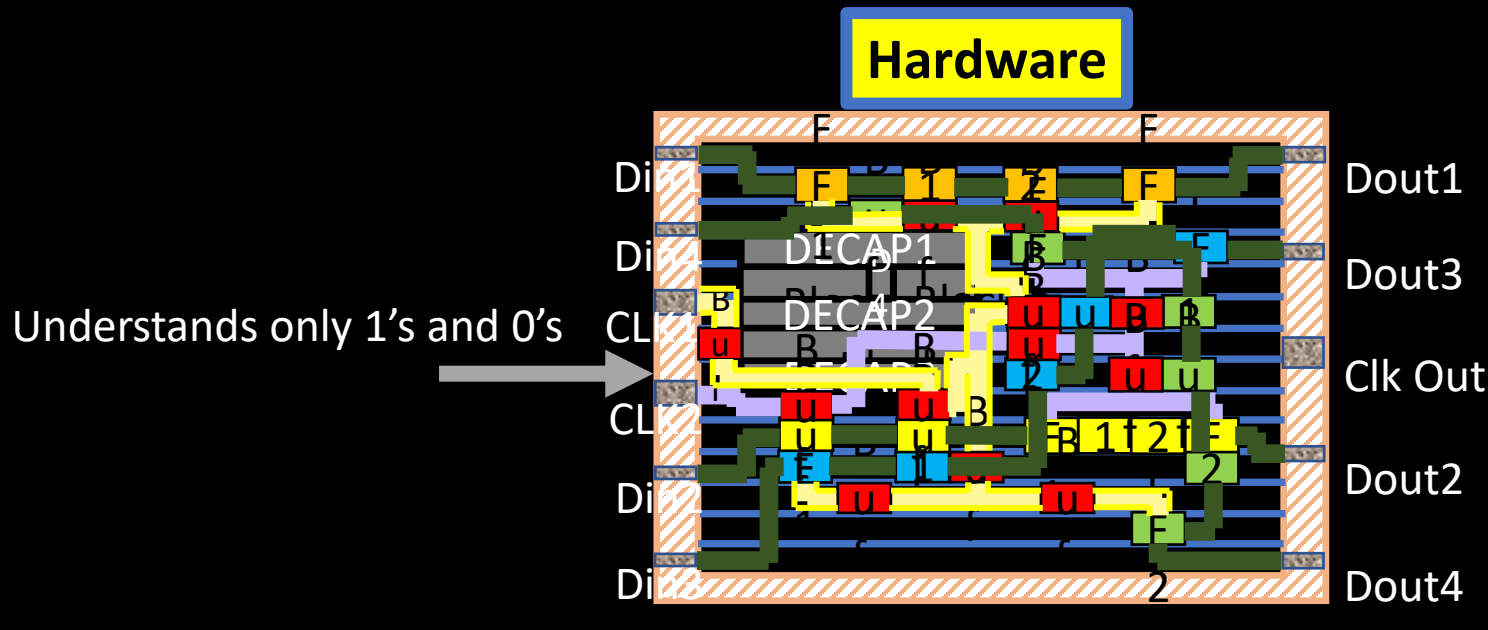

Hardware
DECAP1
DECAP2
Dout1
Dout3
Clk Out
Dout2
Dout4

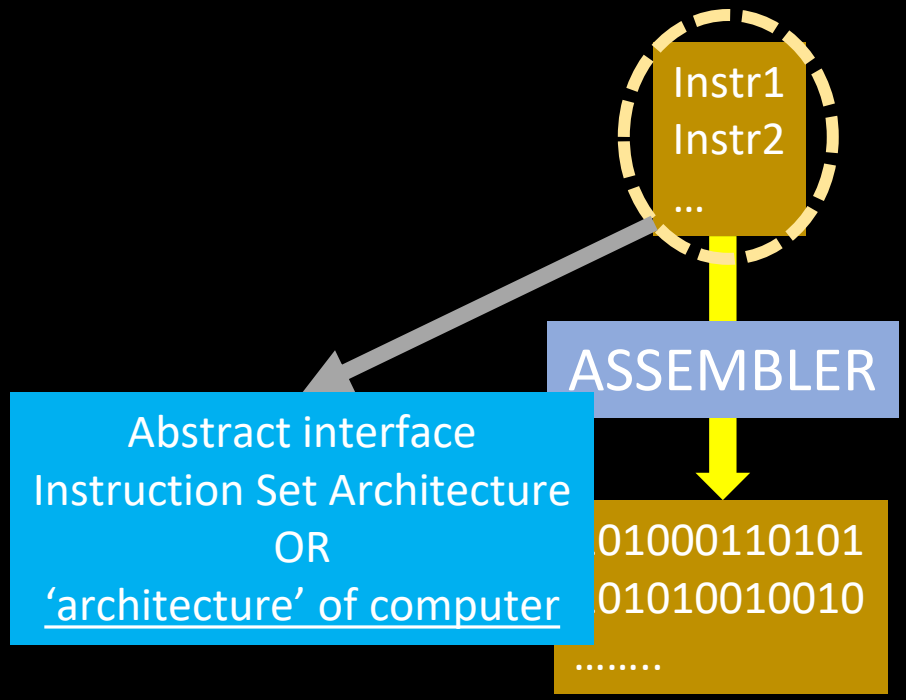

Instr1
Instr2
…
ASSEMBLER
Abstract interface
Instruction Set Architecture
OR
‘architecture’ of computer
01000110101
01010010010
……..


Hardware
DECAP1
DECAP2
Dout1
Dout3
Clk Out
Dout2
Dout4


Instr1
Instr2
…
ASSEMBLER
Abstract interface
Instruction Set Architecture
OR
‘architecture’ of computer
01000 110101
01010 010010
……..
Hardware
DECAP1
DECAP2
Dout1
Dout3
Clk Out
Dout2
Dout4

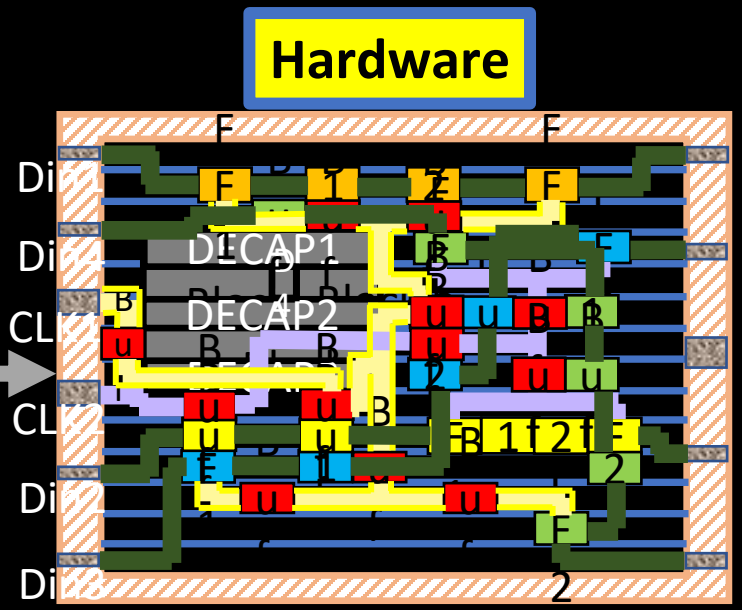

Instr1
Instr2
…
ASSEMBLER
Abstract interface
Instruction Set Architecture
OR
‘architecture’ of computer
01000110101
01010010010
……..


Hardware
Understands only 1’s and 0’s
DECAP1
DECAP2
Dout1
Dout3
Clk Out
Dout2
Dout4



For eg.
00000000011001010000001100110011

Understands only 1's and 0's



Instr1
Instr2
…

ASSEMBLER

Abstract interface
Instruction Set Architecture
OR
‘architecture’ of computer

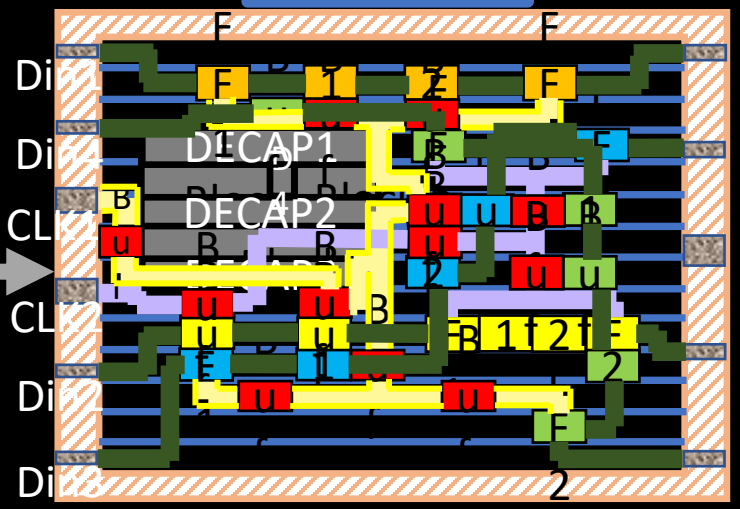
01000110101
01010010010
……..

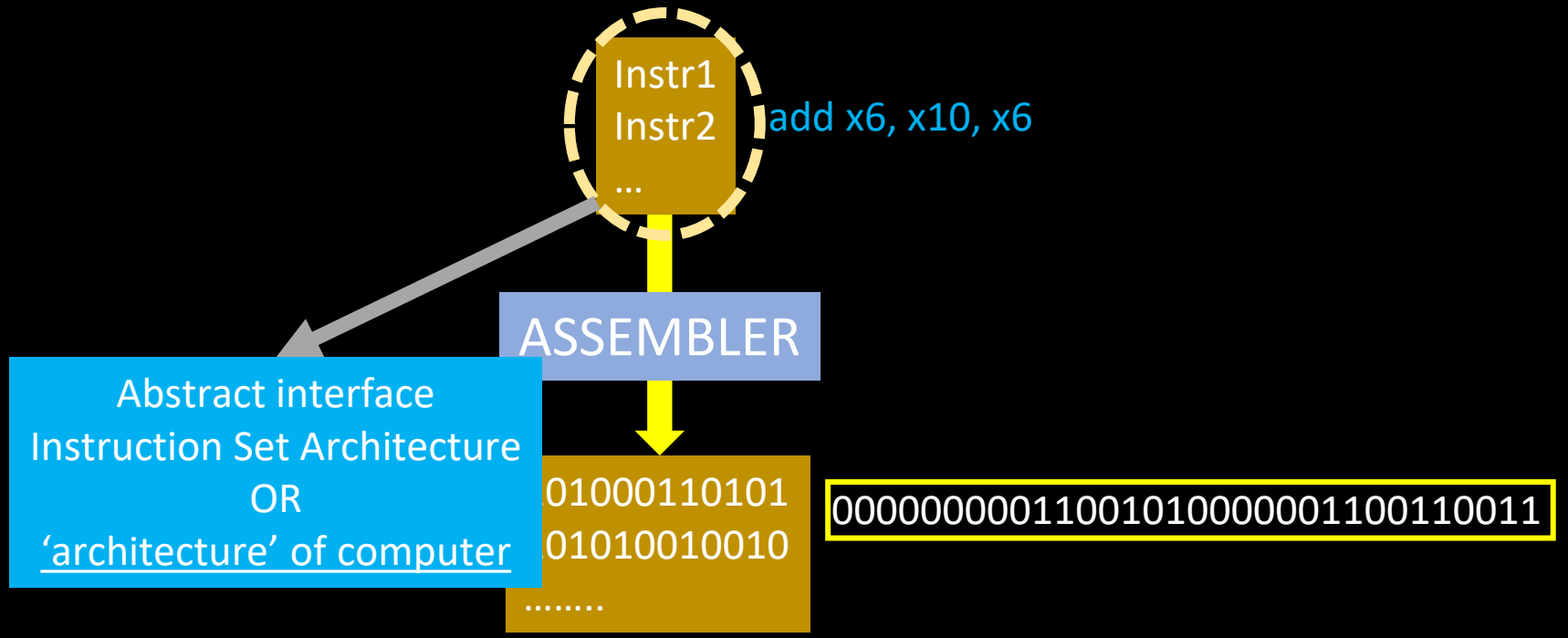
For eg. add x6, x10, x6

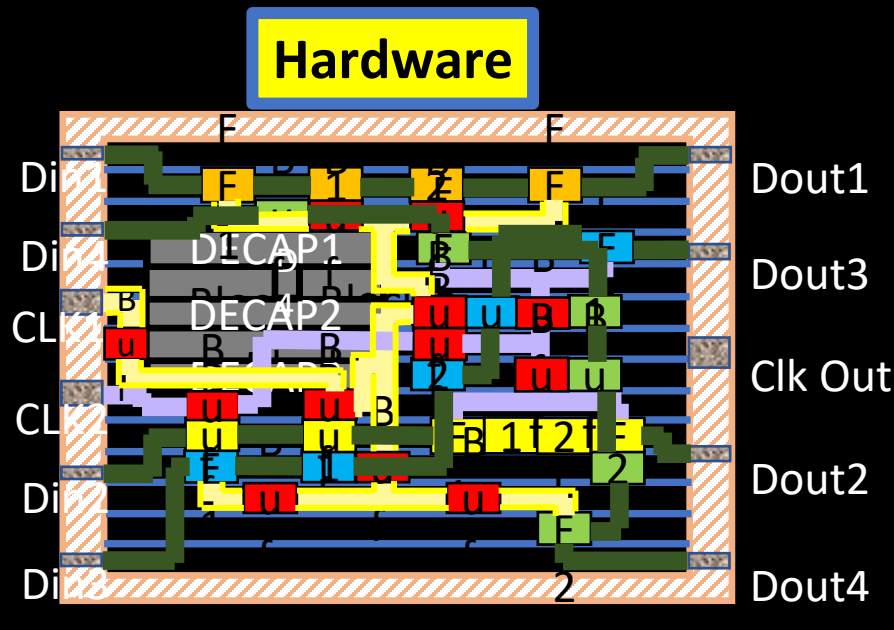
00000000011001010000001100110011

Understands only 1’s and 0’s

Hardware




Instr1
Instr2
...
add x6, x10, x6
ASSEMBLER
Abstract interface
Instruction Set Architecture
OR
'architecture' of computer
01000110101
01010010010
.....
00000000011001010000001100110011


Hardware
DECAP1
DECAP2
Dout1
Dout3
Clk Out
Dout2
Dout4

Instr1
Instr2
...

add x6, x10, x6

ASSEMBLER

Abstract interface
Instruction Set Architecture
OR
‘architecture’ of computer

01000110101
01010010010
.....

00000000011001010000001100110011

RTL snippet of hardware which understands instructions like ‘add x6, x10, x6

```
module picorv32 #(
        parameter [ 0:0] ENABLE_COUNTERS = 1,
        parameter [ 0:0] ENABLE_COUNTERS64 = 1,
        parameter [ 0:0] ENABLE_REGS_
        parameter [ 0:0] ENABLE_REGS_D
        parameter [ 0:0] LATCHED_MEM_R
        parameter [ 0:0] TWO_STAGE_SHI
        parameter [ 0:0] BARREL_SHIFTE
        parameter [ 0:0] TWO_CYCLE_COM
        parameter [ 0:0] TWO_CYCLE_ALU
        parameter [ 0:0] COMPRESSED_IS
        parameter [ 0:0] CATCH_MISALIG
        parameter [ 0:0] CATCH_ILLINSN
```

```
        always @(posedge clk) begin
                is_lui_auipc_jal <= |{instr_lui, instr_auipc, instr_jal};
                is_lui_auipc_jal_jalr_addi_add_sub <= |{instr_lui, instr_auipc, instr_jal, instr_jalr, instr_addi, instr_add, instr_sub};
                is_slti_blt_slt <= |{instr_slti, instr_blt, instr_slt};
                is_sltiu_bltu_sltu <= |{instr_sltiu, instr_bltu, instr_sltu};
                is_lbu_lhu_lw <= |{instr_lbu, instr_lhu, instr_lw};
                is_compare <= |{is_beq_bne_blt_bge_bltu_bgeu, instr_slti, instr_slt, instr_sltiu, instr_sltu};

                if (mem_do_rinst && mem_done) begin
                        instr_lui     <= mem_rdata_latched[6:0] == 7'b0110111;
                        instr_auipc   <= mem_rdata_latched[6:0] == 7'b0010111;
                        instr_jal     <= mem_rdata_latched[6:0] == 7'b1101111;
                        instr_jalr    <= mem_rdata_latched[6:0] == 7'b1100111 && mem_rdata_latched[14:12] == 3'b000;
                        instr_retirq  <= mem_rdata_latched[6:0] == 7'b0001011 && mem_rdata_latched[31:25] == 7'b0000010 && ENABLE_IRQ;
                        instr_waitirq <= mem_rdata_latched[6:0] == 7'b0001011 && mem_rdata_latched[31:25] == 7'b0000100 && ENABLE_IRQ;
```

Hardware

Instr1
Instr2
...

add x6, x10, x6

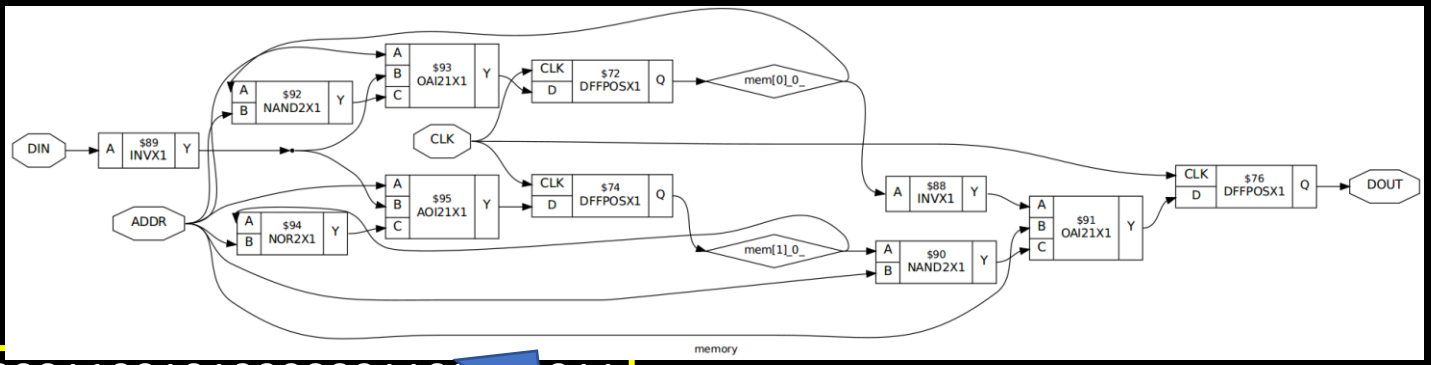
ASSEMBLER

Abstract interface
Instruction Set Architecture
OR
'architecture' of computer

01000110101
01010010010
.....

0000000001100101000000110011

Synthesized netlist of below RTL



RTL snippet of hardware which understands instructions like 'a x6, x10, x6

```
module picorv32 #(
        parameter [ 0:0] ENABLE_COUNTERS = 1,
        parameter [ 0:0] ENABLE_COUNTERS64 = 1,
        parameter [ 0:0] ENABLE_REGS_
        parameter [ 0:0] ENABLE_REGS_
        parameter [ 0:0] LATCHED_MEM_
        parameter [ 0:0] TWO_STAGE_SHI
        parameter [ 0:0] BARREL_SHIFTE
        parameter [ 0:0] TWO_CYCLE_COM
        parameter [ 0:0] TWO_CYCLE_ALU
        parameter [ 0:0] COMPRESSED_IS
        parameter [ 0:0] CATCH_MISALIG
        parameter [ 0:0] CATCH_ILLINSN
```

```
always @(posedge clk) begin
        is_lui_auipc_jal <= |{instr_lui, instr_auipc, instr_jal};
        is_lui_auipc_jal_jalr_addi_add_sub <= |{instr_lui, instr_auipc, instr_jal, instr_jalr, instr_addi, instr_add, instr_sub};
        is_slti_blt_slt <= |{instr_slti, instr_blt, instr_slt};
        is_sltiu_bltu_sltu <= |{instr_sltiu, instr_bltu, instr_sltu};
        is_lbu_lhu_lw <= |{instr_lbu, instr_lhu, instr_lw};
        is_compare <= |{is_beq_bne_blt_bge_bltu_bgeu, instr_slti, instr_slt, instr_sltiu, instr_sltu};

        if (mem_do_rinst && mem_done) begin
                instr_lui     <= mem_rdata_latched[6:0] == 7'b0110111;
                instr_auipc   <= mem_rdata_latched[6:0] == 7'b0010111;
                instr_jal     <= mem_rdata_latched[6:0] == 7'b1101111;
                instr_jalr    <= mem_rdata_latched[6:0] == 7'b1100111 && mem_rdata_latched[14:12] == 3'b000;
                instr_retirq  <= mem_rdata_latched[6:0] == 7'b0001011 && mem_rdata_latched[31:25] == 7'b0000010 && ENABLE_IRQ;
                instr_waitirq <= mem_rdata_latched[6:0] == 7'b0001011 && mem_rdata_latched[31:25] == 7'b0000100 && ENABLE_IRQ;
```

Hardware

Instr1
Instr2
...

add x6, x10, x6

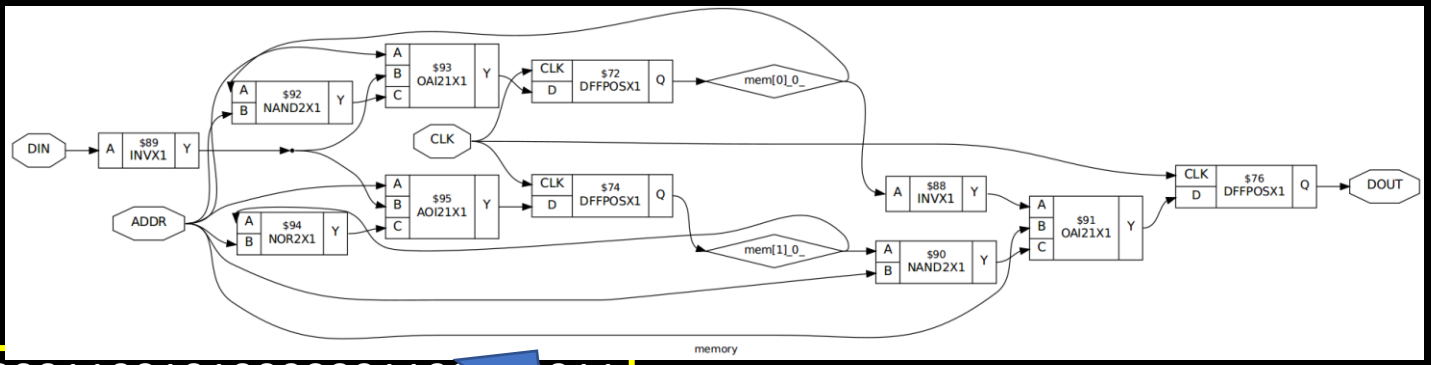
ASSEMBLER

Abstract interface
Instruction Set Architecture
OR
'architecture' of computer

01000110101
01010010010
.....

0000000001100101000000110011

Synthesized netlist of below RTL



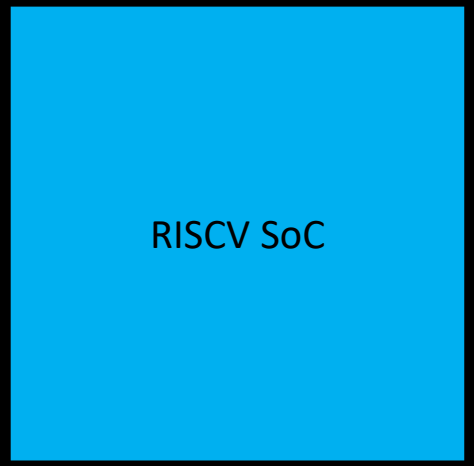
Physical Design Implementation of above netlist

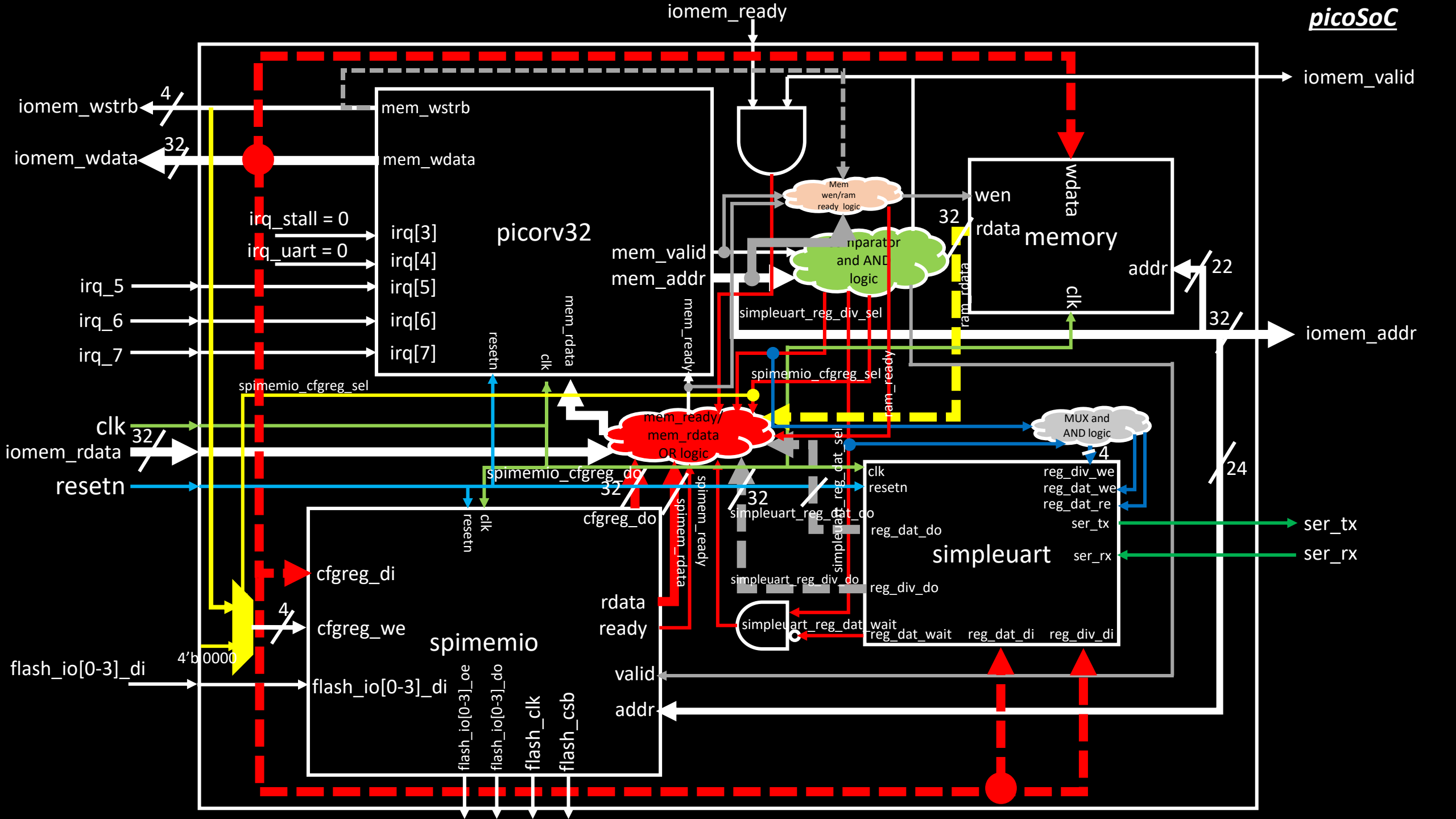
RTL snippet of hardware which understands instructions like 'a x6, x10, x6

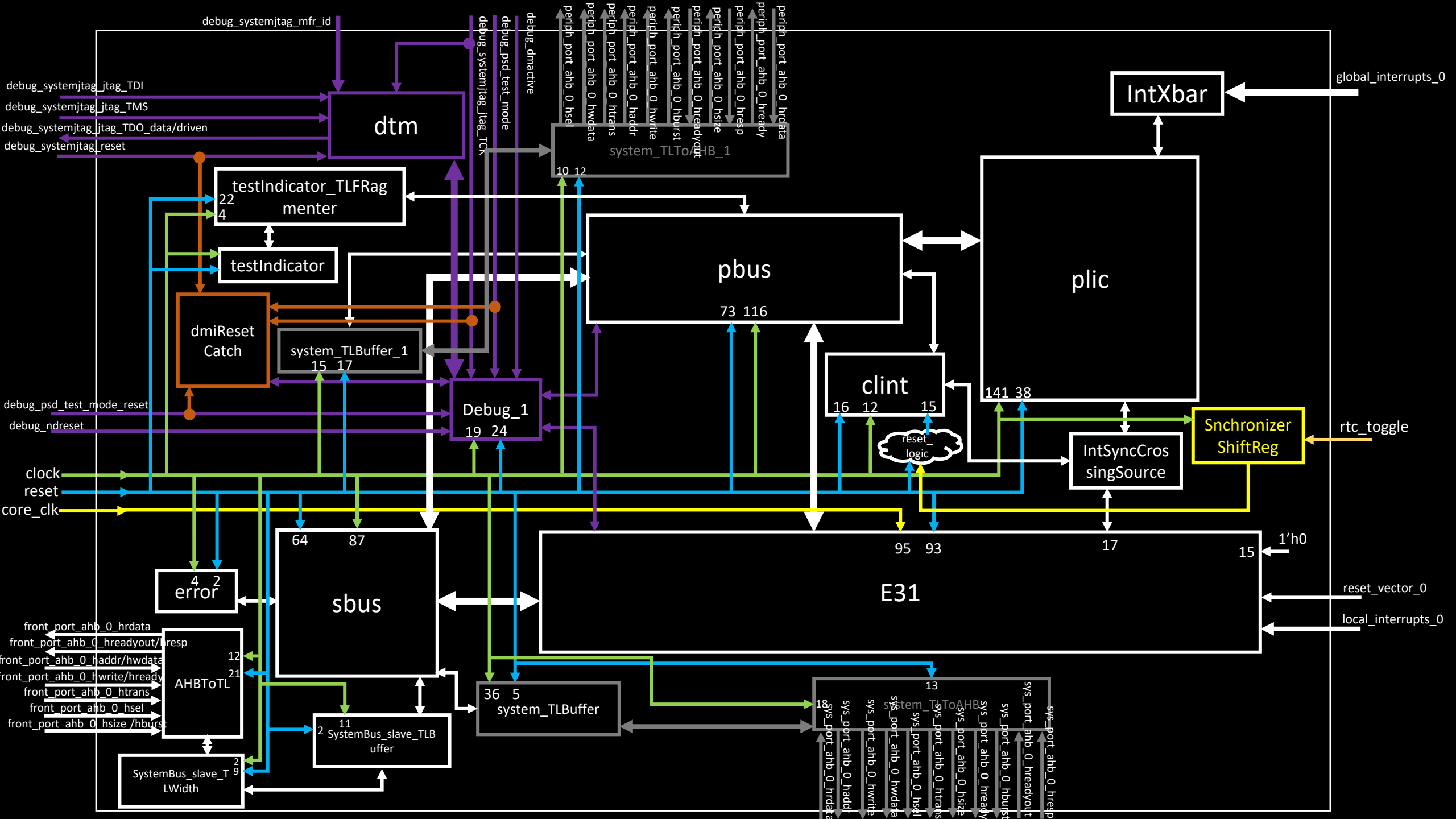
```
module picorv32 #(
        parameter [ 0:0] ENABLE_COUNTERS = 1,
        parameter [ 0:0] ENABLE_COUNTERS64 = 1,
        parameter [ 0:0] ENABLE_REGS_
        parameter [ 0:0] ENABLE_REGS_D
        parameter [ 0:0] LATCHED_MEM_R
        parameter [ 0:0] TWO_STAGE_SHI
        parameter [ 0:0] BARREL_SHIFTE
        parameter [ 0:0] TWO_CYCLE_COM
        parameter [ 0:0] TWO_CYCLE_ALU
        parameter [ 0:0] COMPRESSED_IS
        parameter [ 0:0] CATCH_MISALIG
        parameter [ 0:0] CATCH_ILLINSN
```

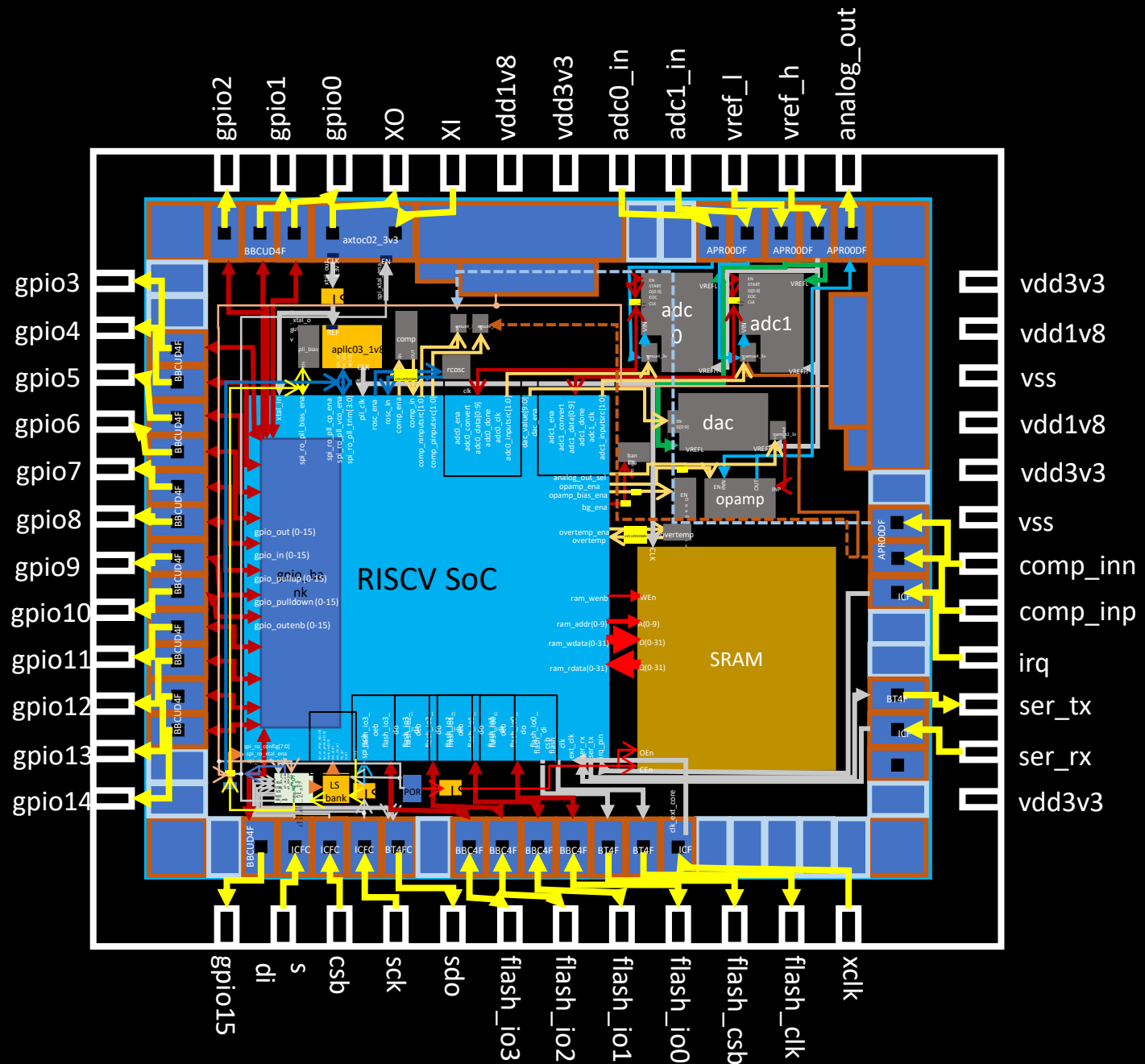
```
        always @(posedge clk) begin
                is_lui_auipc_jal <= |{instr_lui, instr_auipc, instr_jal};
                is_lui_auipc_jal_jalr_addi_add_sub <= |{instr_lui, instr_auipc, instr_jal, instr_jalr, instr_addi, instr_add, instr_sub};
                is_slti_blt_slt <= |{instr_slti, instr_blt, instr_slt};
                is_sltiu_bltu_sltu <= |{instr_sltiu, instr_bltu, instr_sltu};
                is_lbu_lhu_lw <= |{instr_lbu, instr_lhu, instr_lw};
                is_compare <= |{is_beq_bne_blt_bge_bltu_bgeu, instr_slti, instr_slt, instr_sltiu, instr_sltu};

                if (mem_do_rinst && mem_done) begin
                        instr_lui     <= mem_rdata_latched[6:0] == 7'b0110111;
                        instr_auipc   <= mem_rdata_latched[6:0] == 7'b0010111;
                        instr_jal     <= mem_rdata_latched[6:0] == 7'b1101111;
                        instr_jalr    <= mem_rdata_latched[6:0] == 7'b1100111 && mem_rdata_latched[14:12] == 3'b000;
                        instr_retirq  <= mem_rdata_latched[6:0] == 7'b0001011 && mem_rdata_latched[31:25] == 7'b0000010 && ENABLE_IRQ;
                        instr_waitirq <= mem_rdata_latched[6:0] == 7'b0001011 && mem_rdata_latched[31:25] == 7'b0000100 && ENABLE_IRQ;
```

Hardware

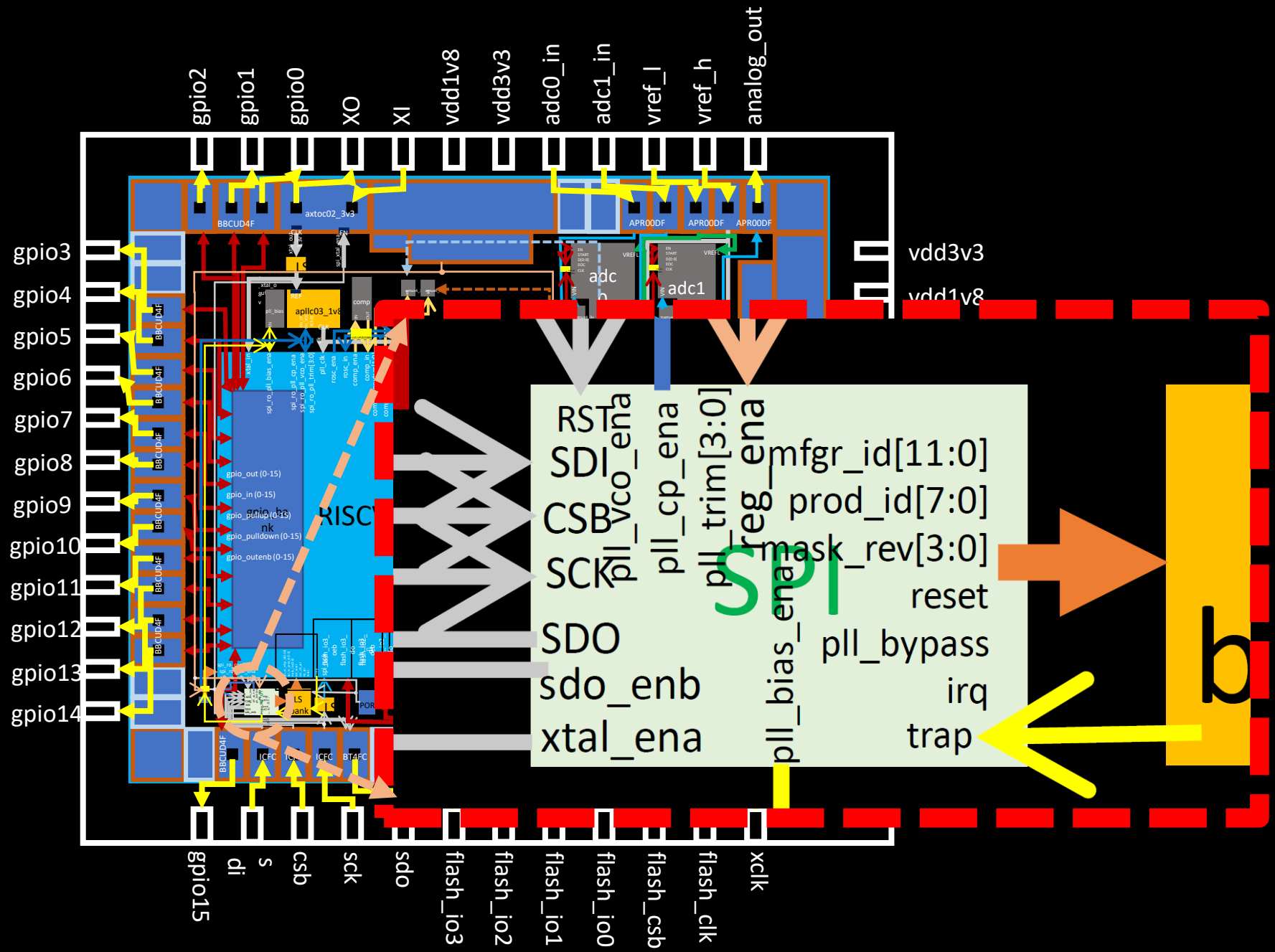

RISCV SoC

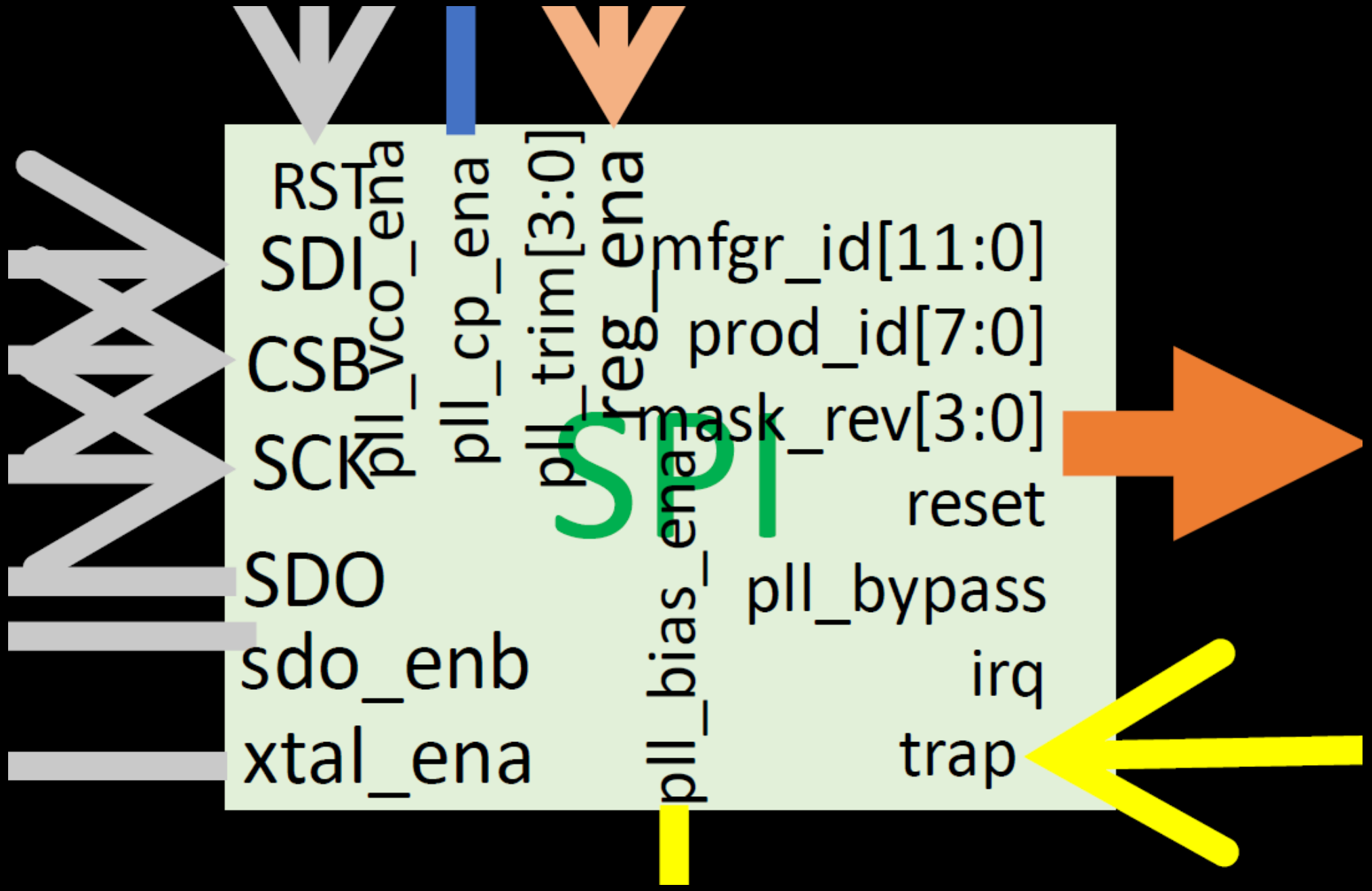

picoSoC
iomem_ready
iomem_valid
iomem_wstrb
4
iomem_wdata
32
mem_wstrb
mem_wdata
irq_stall = 0
irq_uart = 0
irq_5
irq_6
irq_7
irq[3]
irq[4]
irq[5]
irq[6]
irq[7]
picorv32
mem_valid
mem_addr
mem_ready
mem_rdata
resetn
clk
spimemio_cfgreg_sel
Mem wen/ram ready logic
wen
rdata
wdata
memory
addr
22
32
iomem_addr
ram_rdata
ram_ready
simpleuart_reg_div_sel
spimemio_cfgreg_sel
clk
iomem_rdata
resetn
mem_ready/ mem_rdata OR logic
MUX and AND logic
4
24
spimemio_cfgreg_do
32
cfgreg_do
clk
resetn
spimem_rdata
spimem_ready
simpleuart_reg_dat_sel
simpleuart_reg_dat_do
simpleuart_reg_div_do
simpleuart_reg_dat_wait
clk
resetn
reg_dat_do
reg_div_do
reg_dat_wait
reg_dat_di
reg_div_di
reg_div_we
reg_dat_we
reg_dat_re
ser_tx
ser_rx
simpleuart
ser_tx
ser_rx
cfgreg_di
cfgreg_we
4
4'b0000
spimemio
rdata
ready
valid
addr
flash_io[0-3]_di
flash_io[0-3]_di
flash_io[0-3]_oe
flash_io[0-3]_do
flash_clk
flash_csb


debug_systemjtag_mfr_id
debug_systemjtag_jtag_TDI
debug_systemjtag_jtag_TMS
debug_systemjtag_jtag_TDO_data/driven
debug_systemjtag_reset
dtm
debug_systemjtag_jtag_TCK
debug_psd_test_mode
debug_dmactive
periph_port_ahb_0_hsel
periph_port_ahb_0_hwdata
periph_port_ahb_0_htrans
periph_port_ahb_0_haddr
periph_port_ahb_0_hwrite
periph_port_ahb_0_hburst
periph_port_ahb_0_hreadyout
periph_port_ahb_0_hsize
periph_port_ahb_0_hresp
periph_port_ahb_0_hready
periph_port_ahb_0_hrdata
system_TLToAHB_1
10 12
IntXbar
global_interrupts_0
testIndicator_TLFRagmenter
22
4
testIndicator
dmiReset Catch
system_TLBuffer_1
15 17
pbus
73 116
plic
141 38
clint
16 12 15
reset_logic
Debug_1
19 24
debug_psd_test_mode_reset
debug_ndreset
IntSyncCrossingSource
Snchronizer ShiftReg
rtc_toggle
clock
reset
core_clk
64 87
95 93
17
15
1'h0
error
4 2
sbus
E31
reset_vector_0
local_interrupts_0
front_port_ahb_0_hrdata
front_port_ahb_0_hreadyout/hresp
front_port_ahb_0_haddr/hwdata
front_port_ahb_0_hwrite/hready
front_port_ahb_0_htrans
front_port_ahb_0_hsel
front_port_ahb_0_hsize /hburst
AHBToTL
12
21
SystemBus_slave_TLWidth
2
9
11
2
SystemBus_slave_TLBuffer
36 5
system_TLBuffer
13
18
system_TLToAHB
sys_port_ahb_0_hrdata
sys_port_ahb_0_haddr
sys_port_ahb_0_hwrite
sys_port_ahb_0_hwdata
sys_port_ahb_0_hsel
sys_port_ahb_0_htrans
sys_port_ahb_0_hsize
sys_port_ahb_0_hready
sys_port_ahb_0_hburst
sys_port_ahb_0_hreadyout
sys_port_ahb_0_hresp

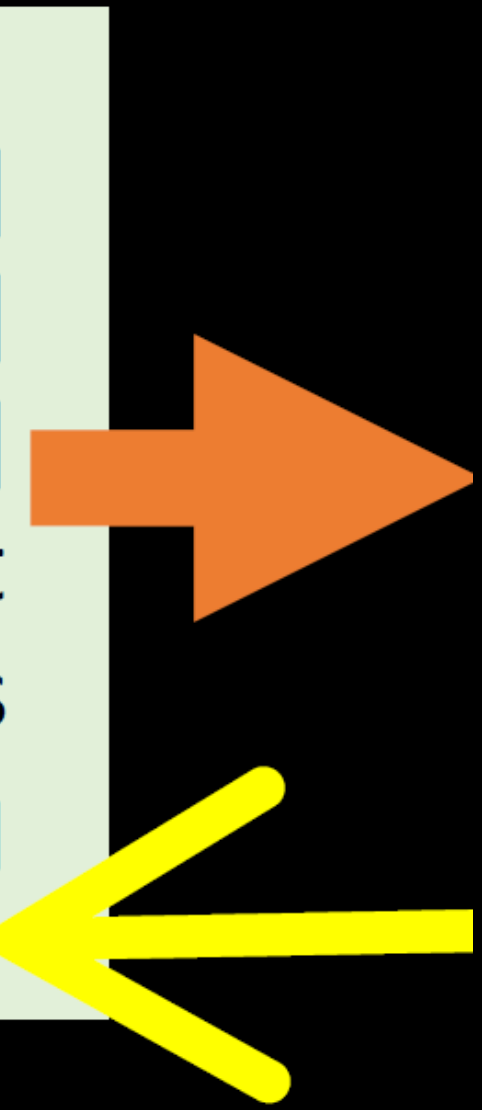


https://www.vlsisystemdesign.com/


gpio2
gpio1
gpio0
XO
XI
vdd1v8
vdd3v3
adc0_in
adc1_in
vref_l
vref_h
analog_out
gpio3
gpio4
gpio5
gpio6
gpio7
gpio8
gpio9
gpio10
gpio11
gpio12
gpio13
gpio14
vdd3v3
vdd1v8
BBCUD4F
axtoc02_3v3
APR00DF
adc
adc1
apllc03_1v8
RISC
gpio_out (0-15)
gpio_in (0-15)
gpio_pullup (0-15)
gpio_pulldown (0-15)
gpio_outenb (0-15)
RST
SDI
CSB
SCK
SDO
sdo_enb
xtal_ena
pll_vco_ena
pll_cp_ena
pll_trim[3:0]
pll_reg_ena
pll_bias_ena
SPI
mfgr_id[11:0]
prod_id[7:0]
mask_rev[3:0]
reset
pll_bypass
irq
trap
b
gpio15
di
s
csb
sck
sdo
flash_io3
flash_io2
flash_io1
flash_io0
flash_csb
flash_clk
xclk



```
module spi(RST, SCK, SDI, CSB, SDO, sdo_enb,
        xtal_ena, reg_ena, pll_vco_ena, pll_cp_ena, pll_bias_ena,
        pll_trim, pll_bypass, irq, reset, trap,
        mfgr_id, prod_id, mask_rev_in, mask_rev);

    input RST;
    input SCK;
    input SDI;
    input CSB;
    output SDO;
    output sdo_enb;
    output xtal_ena;
    output reg_ena;
    output pll_vco_ena;
    output pll_cp_ena;
    output pll_bias_ena;
    output [3:0] pll_trim;
    output pll_bypass;
    output irq;
    output reset;
    input  trap;
    input [3:0] mask_rev_in;    // metal programmed
    output [11:0] mfgr_id;
    output [7:0] prod_id;
    output [3:0] mask_rev;

    reg xtal_ena;
    reg reg_ena;
    reg [3:0] pll_trim;
    reg pll_vco_ena;
    reg pll_cp_ena;
    reg pll_bias_ena;
```

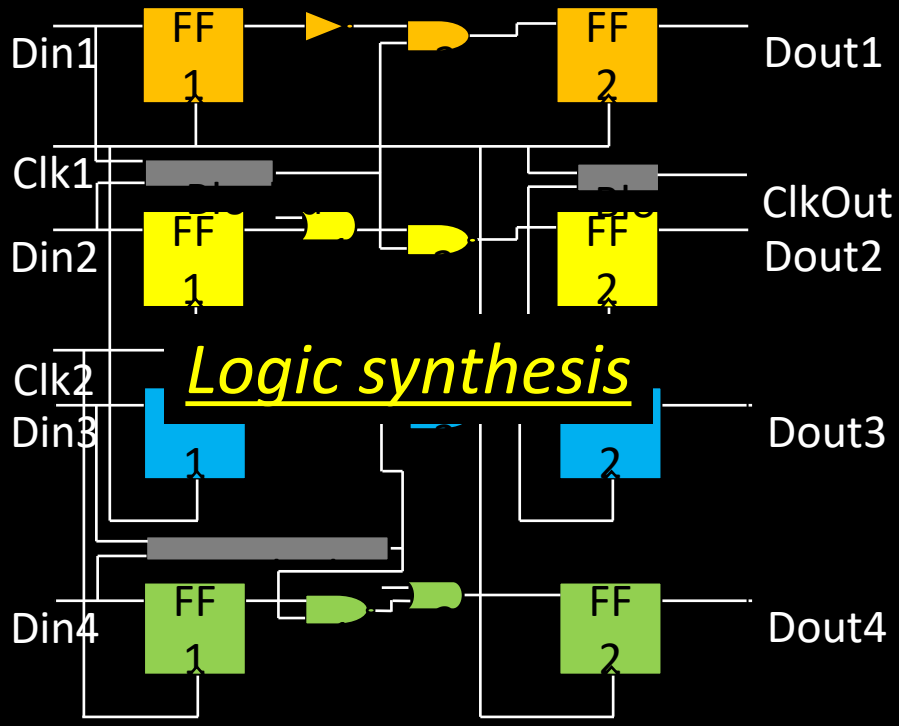


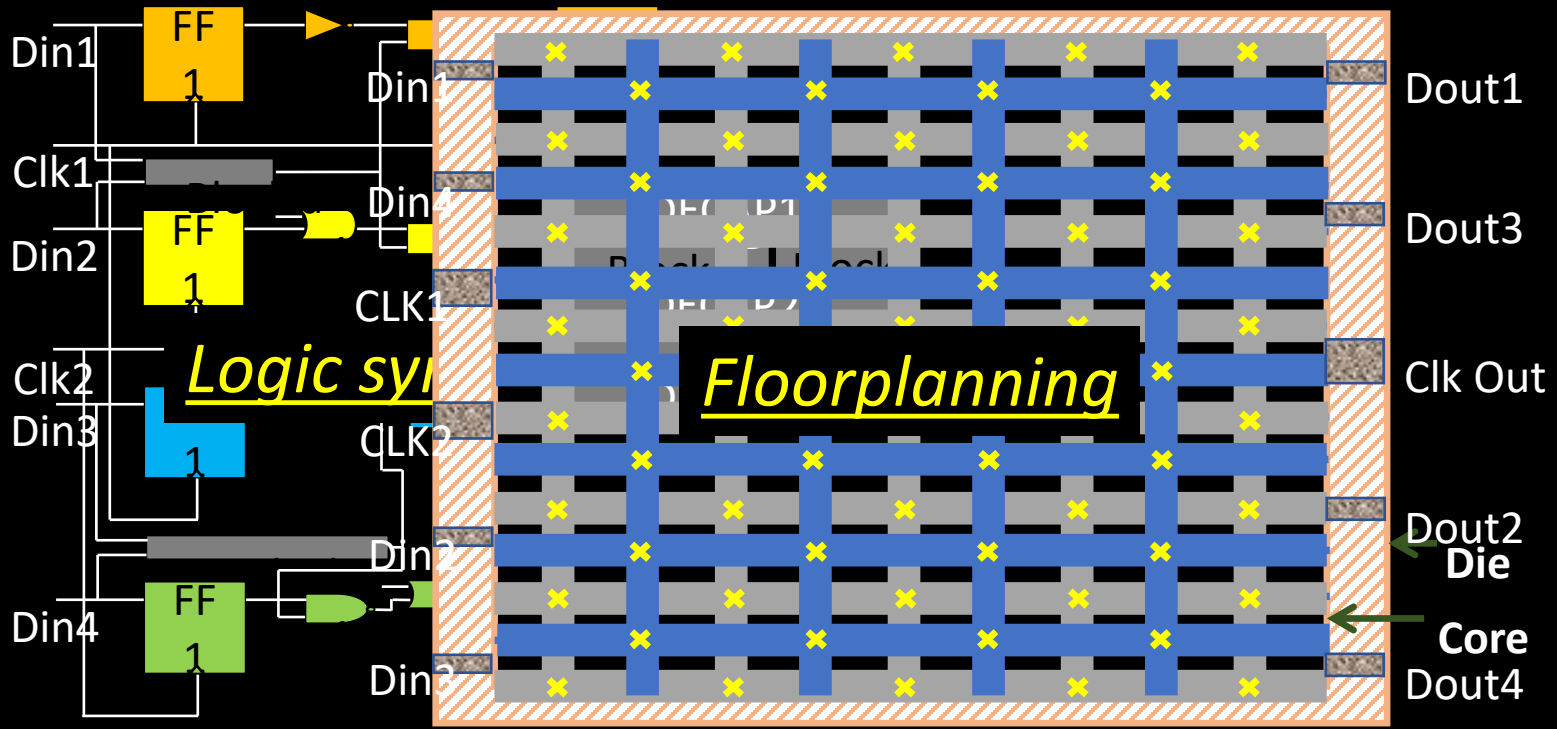
```
module spi(RST, SCK, SDI, CSB, SDO, sdo_enb,
        xtal_ena, reg_ena, pll_vco_ena, pll_cp_ena, pll_bias_ena,
        pll_trim, pll_byp
        mfgr_id, prod_id,

    input RST;
    input SCK;
    input SDI;
    input CSB;
    output SDO;
    output sdo_enb;
    output xtal_ena;
    output reg_ena;
    output pll_vco_ena;
    output pll_cp_ena;
    output pll_bias_ena;
    output [3:0] pll_trim;
    output pll_bypass;
    output irq;
    output reset;
    input  trap;
    input [3:0] mask_rev_i
    output [11:0] mfgr_id
    output [7:0] prod_id;
    output [3:0] mask_rev

    reg xtal_ena;
    reg reg_ena;
    reg [3:0] pll_trim;
    reg pll_vco_ena;
    reg pll_cp_ena;
    reg pll_bias_ena;
```

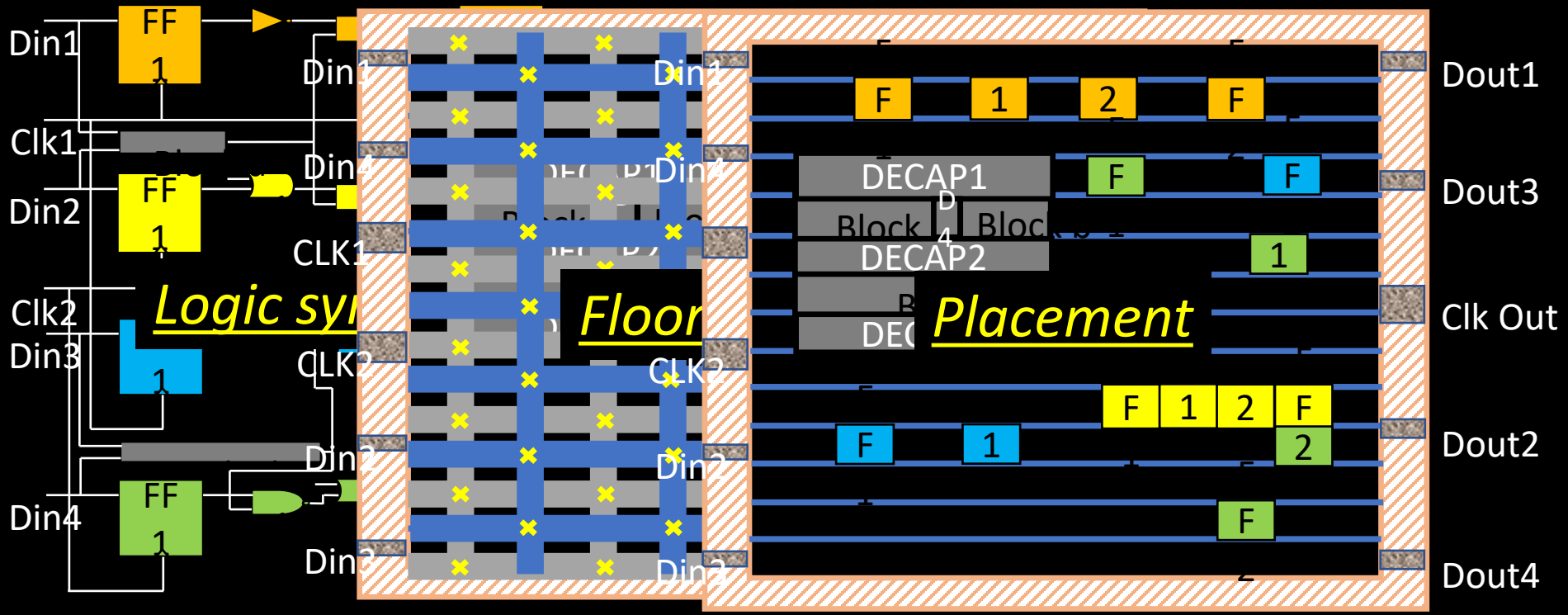
```
assign odata =
    (iaddr == 8'h00) ? 8'h00 :      // SPI status (fixed)
    (iaddr == 8'h01) ? {mask_rev, mfgr_id[11:8]} :  // Mask rev (metal programmed)
    (iaddr == 8'h02) ? mfgr_id[7:0] :       // Manufacturer ID (fixed)
    (iaddr == 8'h03) ? prod_id :    // Product ID (fixed)
    (iaddr == 8'h04) ? {xtal_ena, reg_ena, pll_vco_ena, pll_cp_ena, pll_trim} :
    (iaddr == 8'h05) ? {7'b0000000, pll_bypass} :
    (iaddr == 8'h06) ? {7'b0000000, irq} :
    (iaddr == 8'h07) ? {7'b0000000, reset} :
    (iaddr == 8'h08) ? {7'b0000000, trap} :
                       8'h00;       // Default

// Register mapping and I/O to slave module

always @(posedge SCK or posedge RST) begin
    if (RST == 1'b1) begin
        pll_trim <= 4'b0000;
        xtal_ena <= 1'b1;
        reg_ena <= 1'b1;
        pll_vco_ena <= 1'b1;
        pll_cp_ena <= 1'b1;
        pll_bias_ena <= 1'b1;
        pll_bypass <= 1'b0;
        irq <= 1'b0;
        reset <= 1'b0;
    end else if (wrstb == 1'b1) begin
        case (iaddr)
            8'h04: begin
                pll_trim    <= idata[7:4];
                pll_cp_ena  <= idata[3];
                pll_vco_ena <= idata[2];
                reg_ena     <= idata[1];
                xtal_ena    <= idata[0];
            end
```

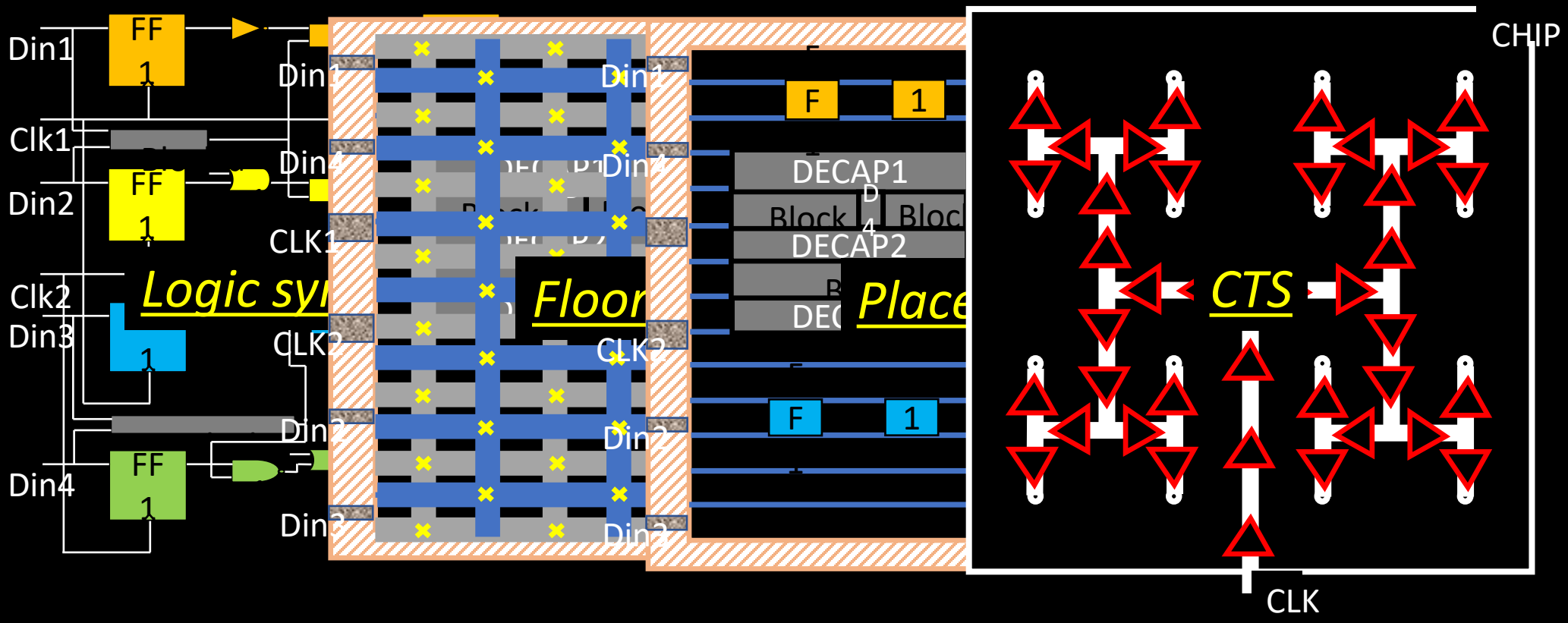
```
module spi(RST, SCK, SDI, CSB, SDO, sdo_enb,
        xtal_ena, reg_ena, pll_vco_ena, pll_cp_ena, pll_bias_ena
        pll_trim, pll_byp
        mfgr_id, prod_id,

    assign odata =
        (iaddr == 8'h00) ? 8'h00 :      // SPI status (fixed)
        (iaddr == 8'h01) ? {mask_rev, mfgr_id[11:8]} :  // Mask rev (metal programmed)
```

Yosys Open Synthesis Suite

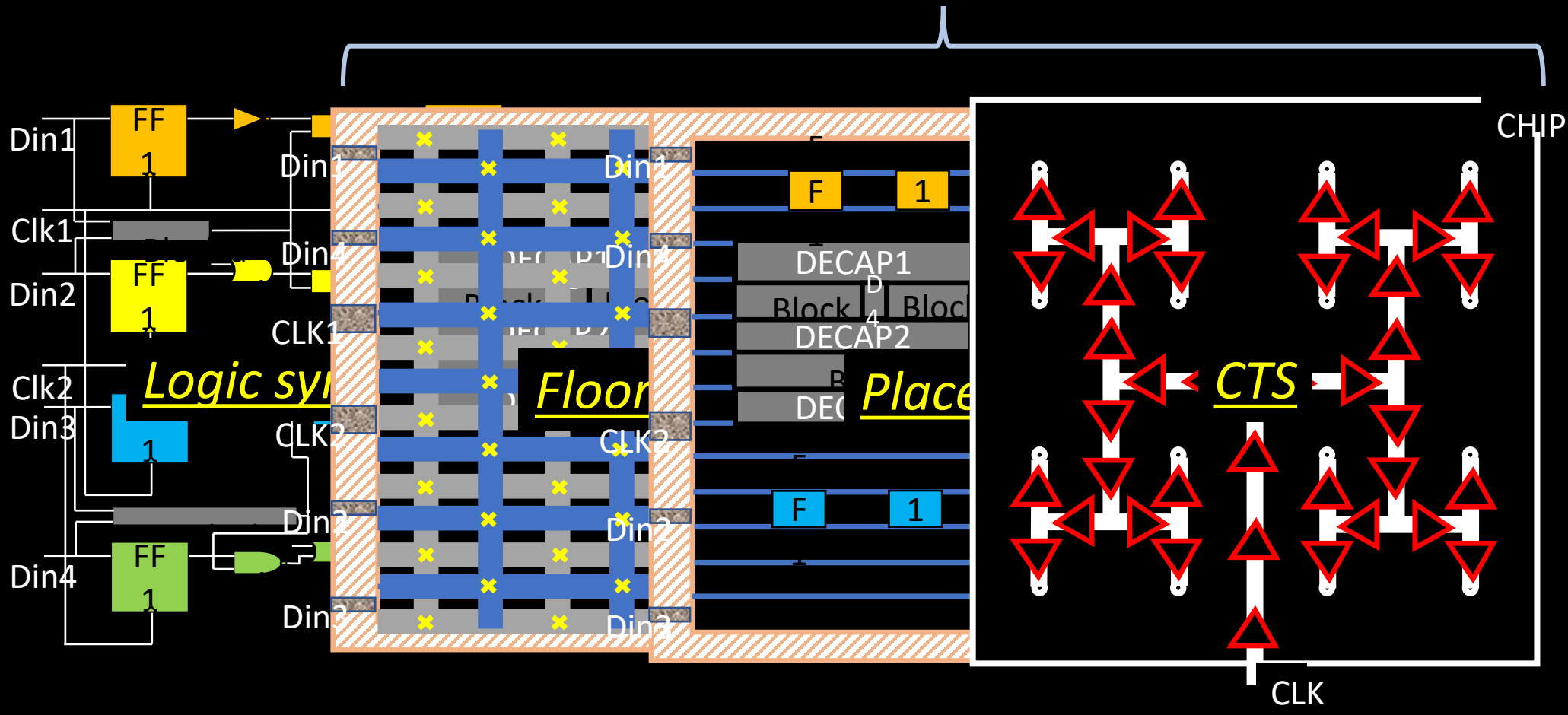


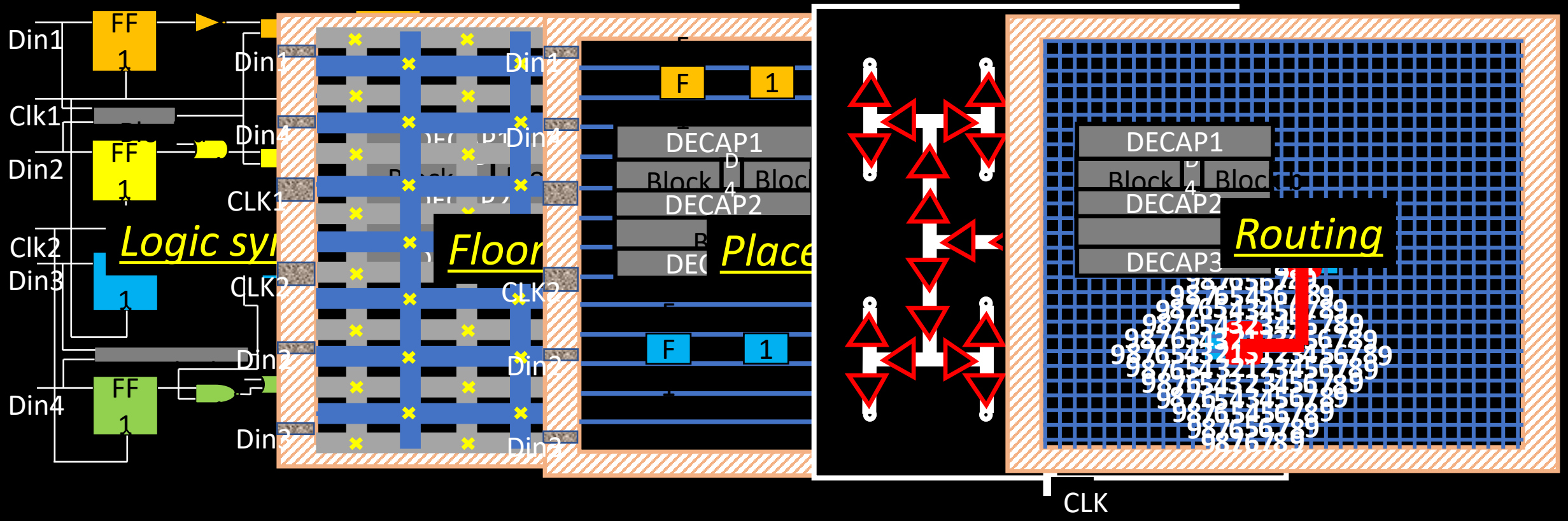

Din1
FF
1
Clk1
Din2
FF
1
Clk2
Din3
1
Din4
FF
1
Logic syn
Floorplanning
Dout1
Dout3
Clk Out
Dout2
Die
Core
Dout4


Din1
FF
1
Clk1
Din2
FF
1
Clk2
Din3
1
Din4
FF
1
Logic syn
Floor
Placement
DECAP1
DECAP2
F
1
2
Dout1
Dout3
Clk Out
Dout2
Dout4

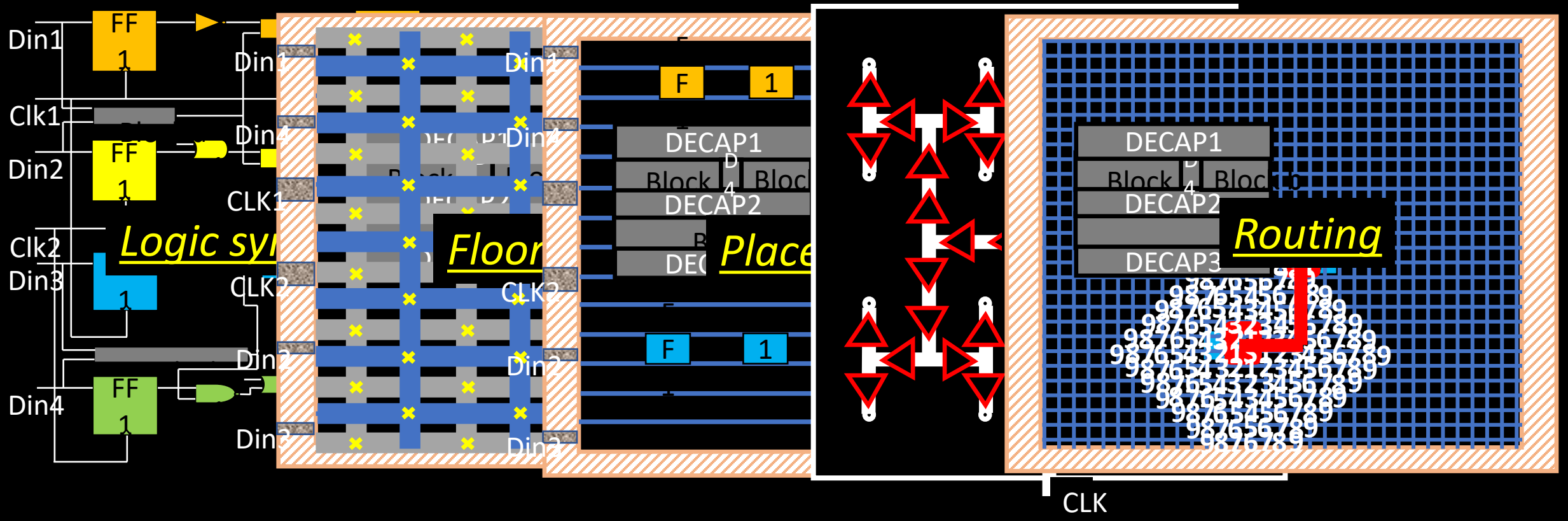

Din1
FF
1
Clk1
Din2
FF
1
Clk2
Din3
1
Din4
FF
1
Logic syn
Din1
Din4
CLK1
CLK2
Din2
Din3
Floor
Din1
Din4
CLK2
Din2
Din3
Place
F
1
DECAP1
DECAP2
F
1
CHIP
CTS
CLK

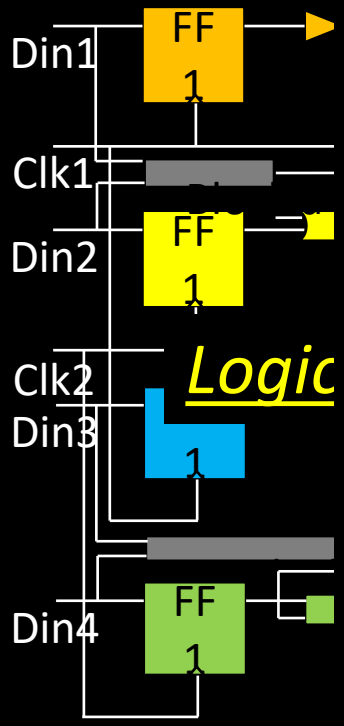
Graywolf


Din1
FF
1
Clk1
Din2
FF
1
Clk2
Din3
1
Din4
FF
1
Logic syn
Floor
Place
CTS
CHIP
CLK
CLK1
CLK2
DECAP1
DECAP2
F
1


Din1
FF
1
Clk1
Din2
FF
1
Clk2
Din3
1
Din4
FF
1
Logic syn
Din1
Din4
CLK1
CLK2
Din2
Din3
Floor
Din1
Din4
CLK2
Din2
Din3
F
1
DECAP1
Block
D
4
Bloc
DECAP2
DEC
Place
F
1
DECAP1
Block
D
4
Bloc
DECAP2
DECAP3
Routing
CLK

Qrouter

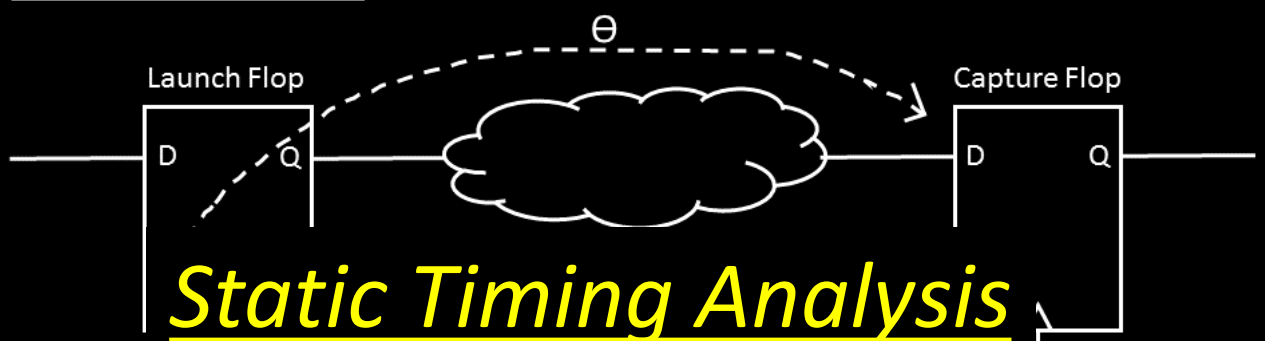

Din1
FF
1
Clk1
Din2
FF
1
Clk2
Din3
1
Din4
FF
1
Logic syn
Floor
Place
Routing
DECAP1
DECAP2
DECAP3
CLK1
CLK2
CLK



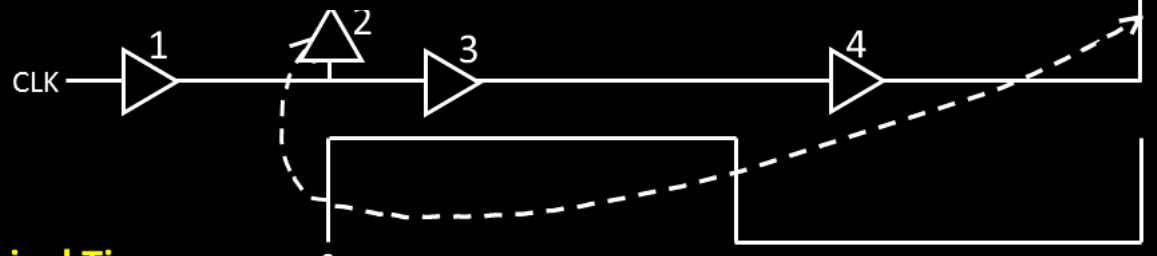
Logic

Timing Analysis (with real Clocks)

Hold Analysis - Single Clock

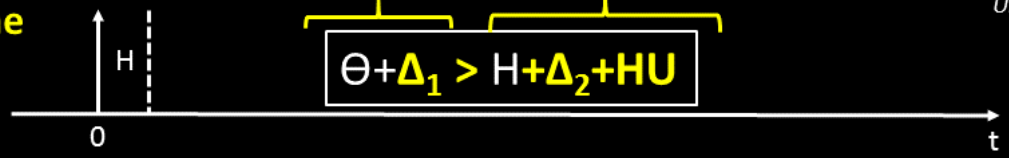


Static Timing Analysis



Data Arrival Time
- Data Required Time

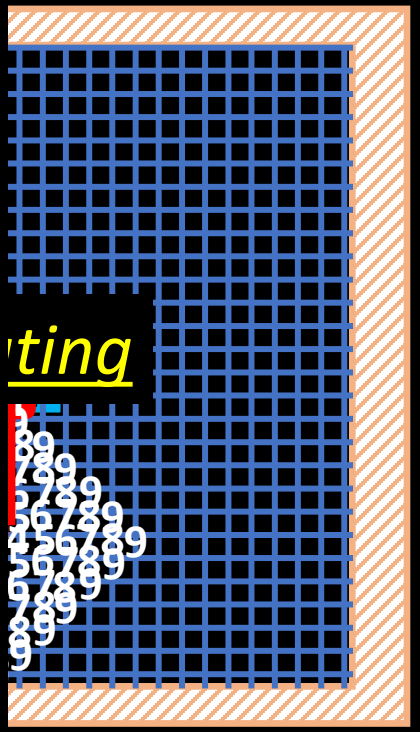
SLACK(should be +ve or '0')



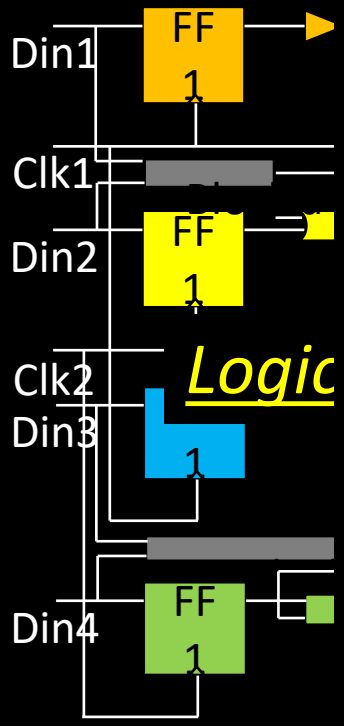
Specifications:
Clock Frequency (F) = *1GHz*
Clock Period (T)= 1/F
= 1/1GHz
= *1ns*

With, T = *1ns*,
Assume H = 10ps
= 0.01ns
Uncertainty = 50ps
= 0.05ns

ting



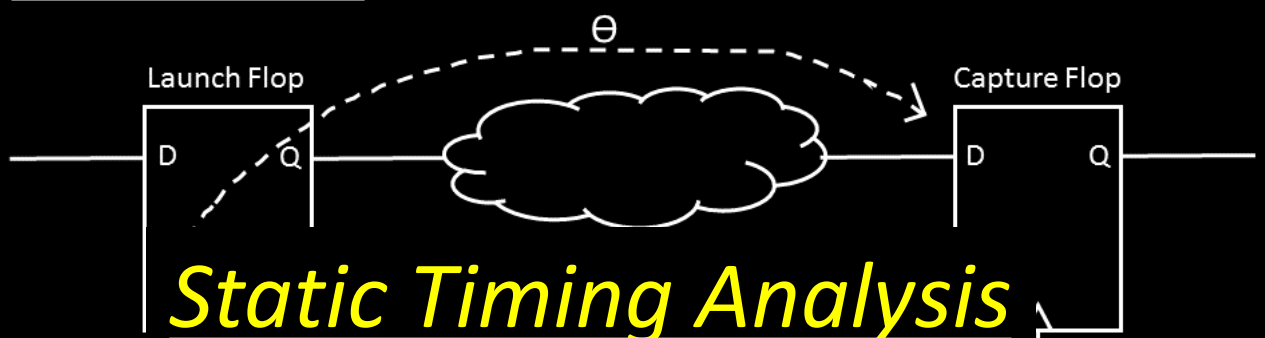
OpenSTA/Opentimer



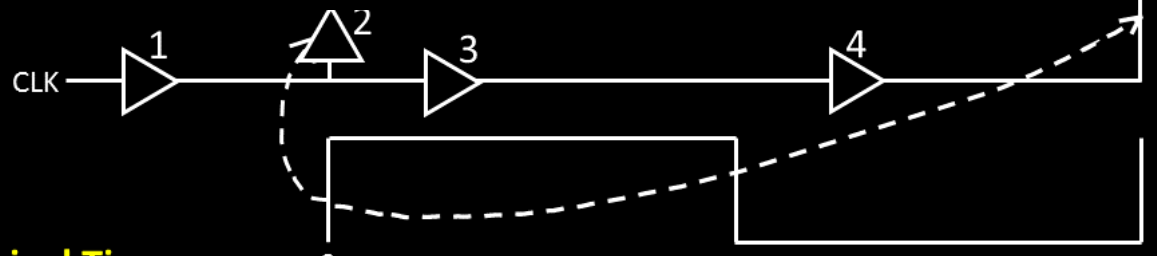
Logic

Timing Analysis (with real Clocks)

Hold Analysis - Single Clock

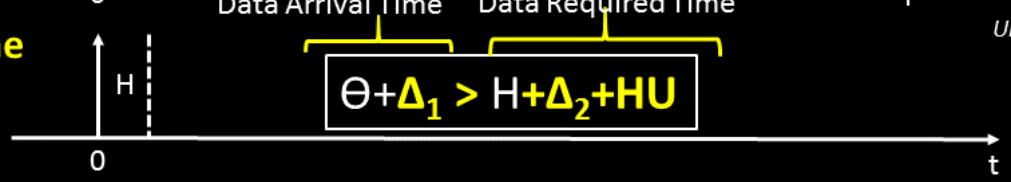


Static Timing Analysis



Data Arrival Time
- Data Required Time

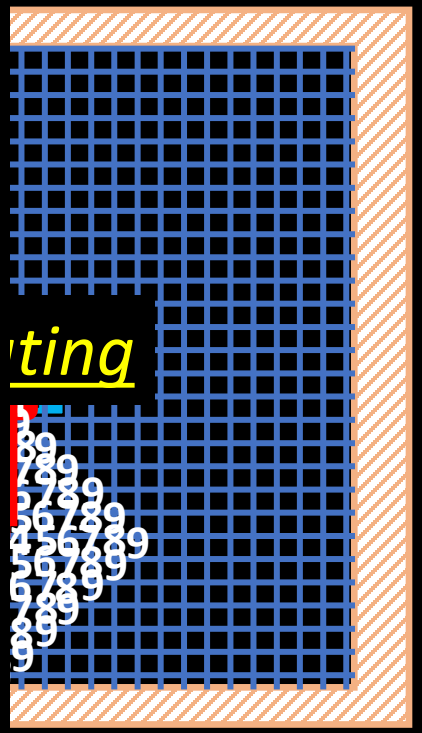
SLACK(should be +ve or '0')



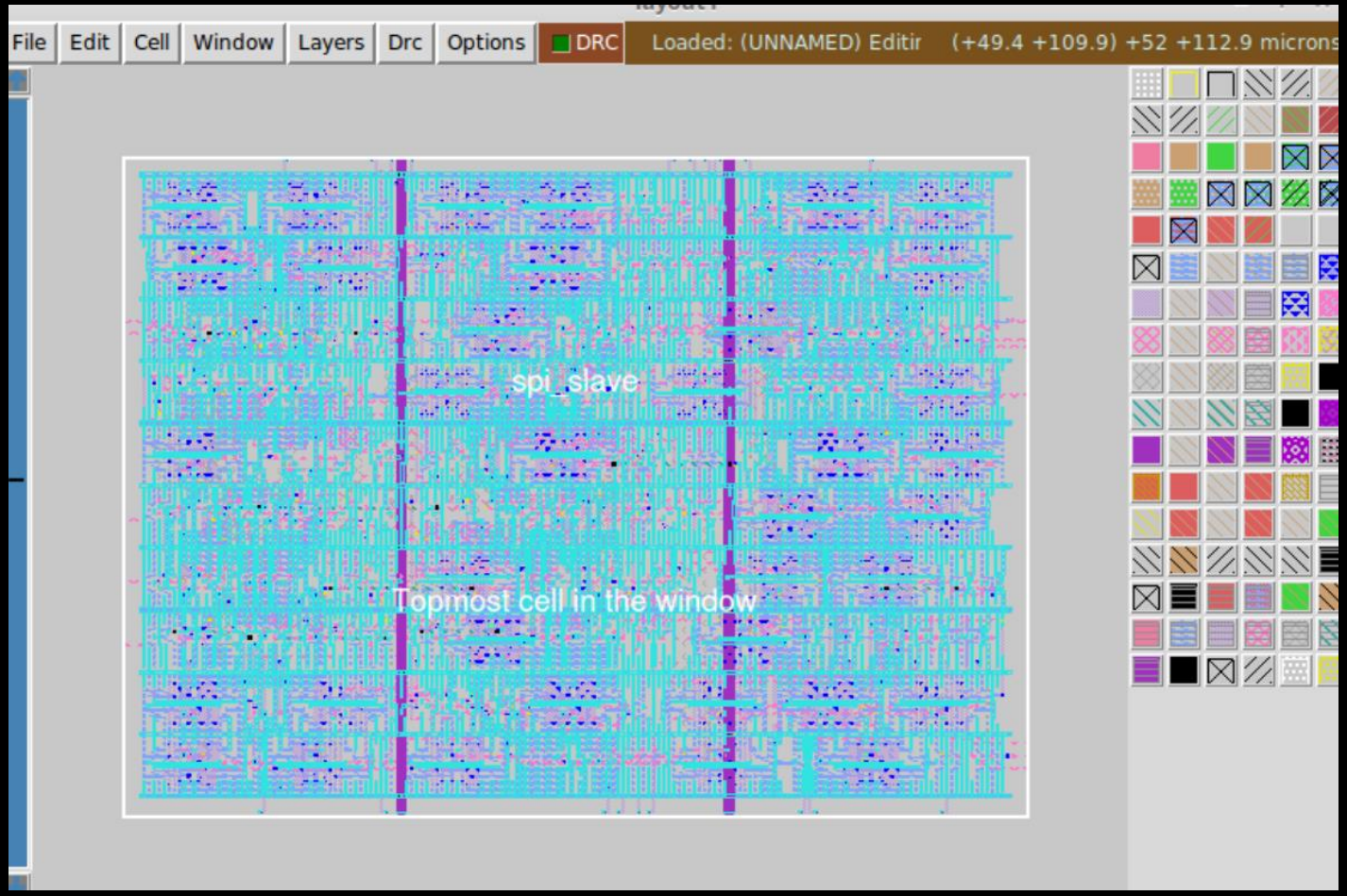
Specifications:
Clock Frequency (F) = *1GHz*
Clock Period (T)= 1/F
= 1/1GHz
= *1ns*

With, T = *1ns*,
Assume H = 10ps
= 0.01ns
Uncertainty = 50ps
= 0.05ns

ting

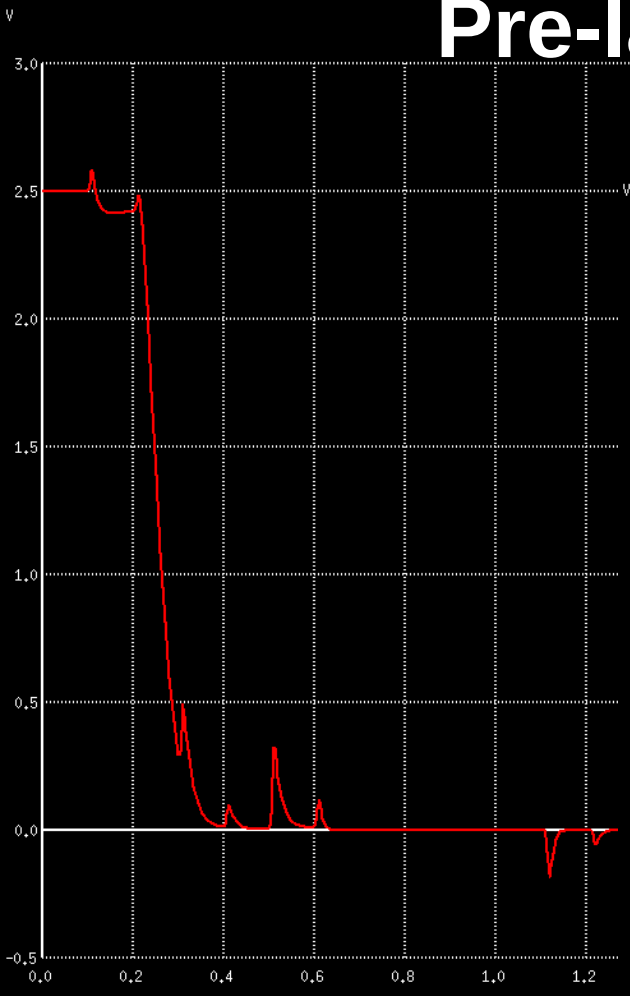


MAGIC - Layout Viewer

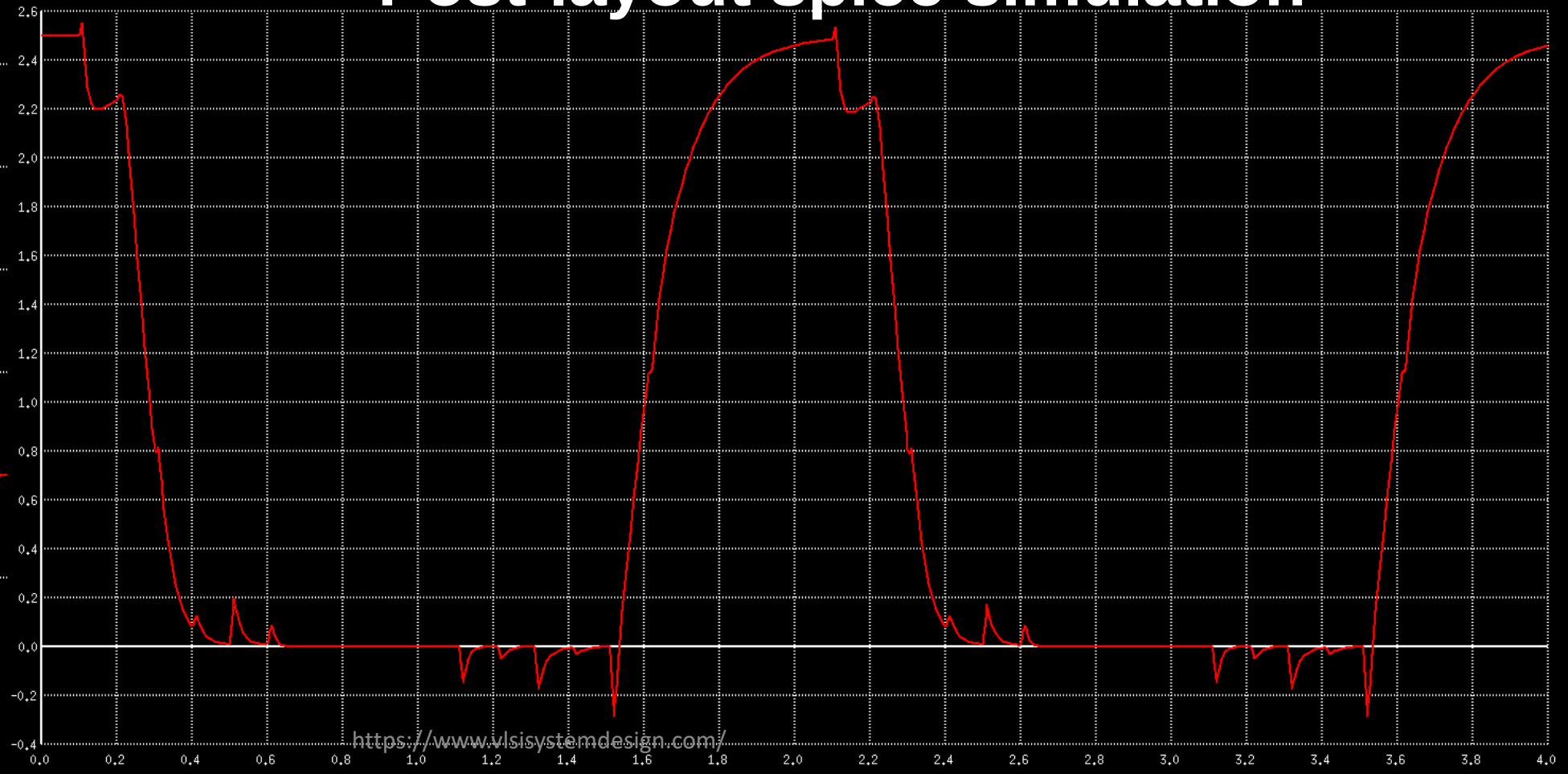


ngSPICE

Pre-layout spice simulation



Post-layout spice simulation



Opensource EDA tool list to be used for this workshop

List of Tools:

- Yosys - RTL Synthesis
- blifFanout - High fanout net (HFN) synthesis
- graywolf – Placement
- qrouter - Detailed routing
- Magic - VLSI Layout tool
- Netgen – LVS
- OpenTimer and OpenSTA - Static timing analysis tool

Opensource EDA tool list to be used for this workshop

Exercise 1: Type below 2 commands in Linux Shell

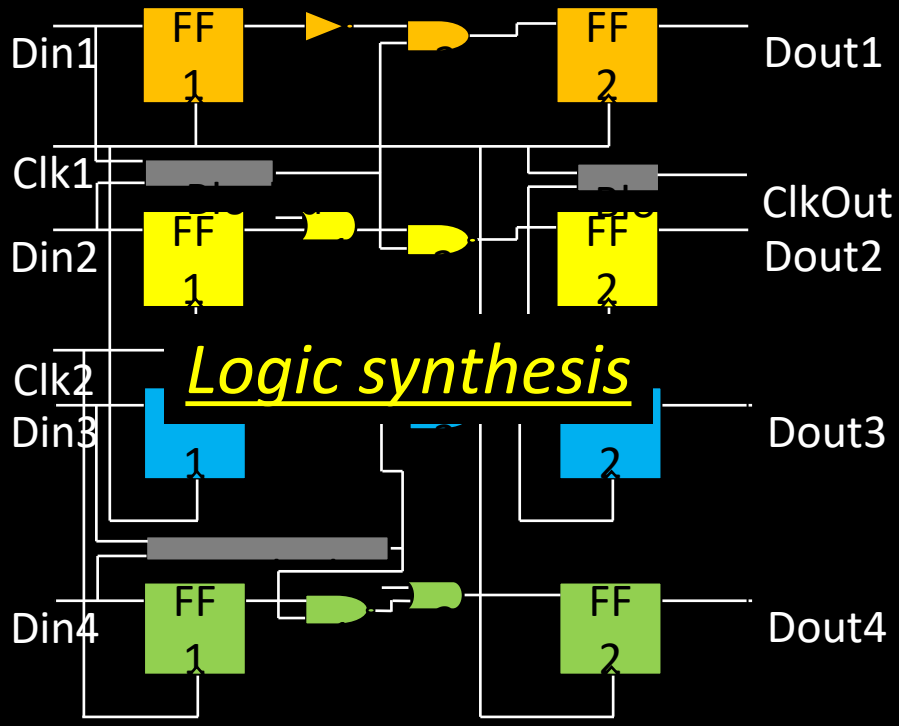
```
sudo apt-get install python3-tk
```

```
sudo apt install ngspice
```

Physical Design Overview

```
module spi(RST, SCK, SDI, CSB, SDO, sdo_enb,
        xtal_ena, reg_ena, pll_vco_ena, pll_cp_ena, pll_bias_ena
        pll_trim, pll_byp
        mfgr_id, prod_id,

    assign odata =
        (iaddr == 8'h00) ? 8'h00 :      // SPI status (fixed)
        (iaddr == 8'h01) ? {mask_rev, mfgr_id[11:8]} :  // Mask rev (metal programmed)
```

Yosys Open Synthesis Suite



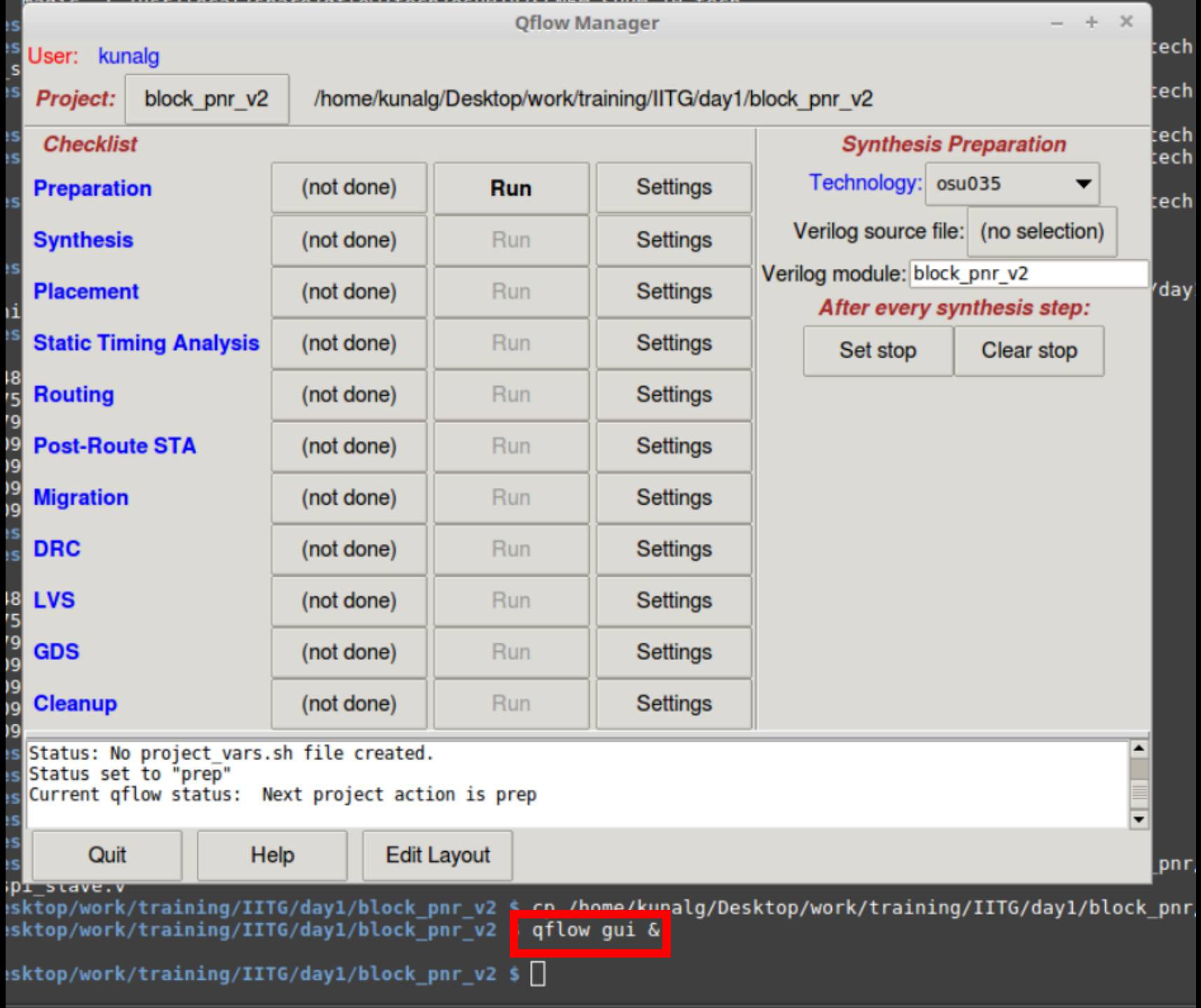
Exercise 2: Download, copy and run

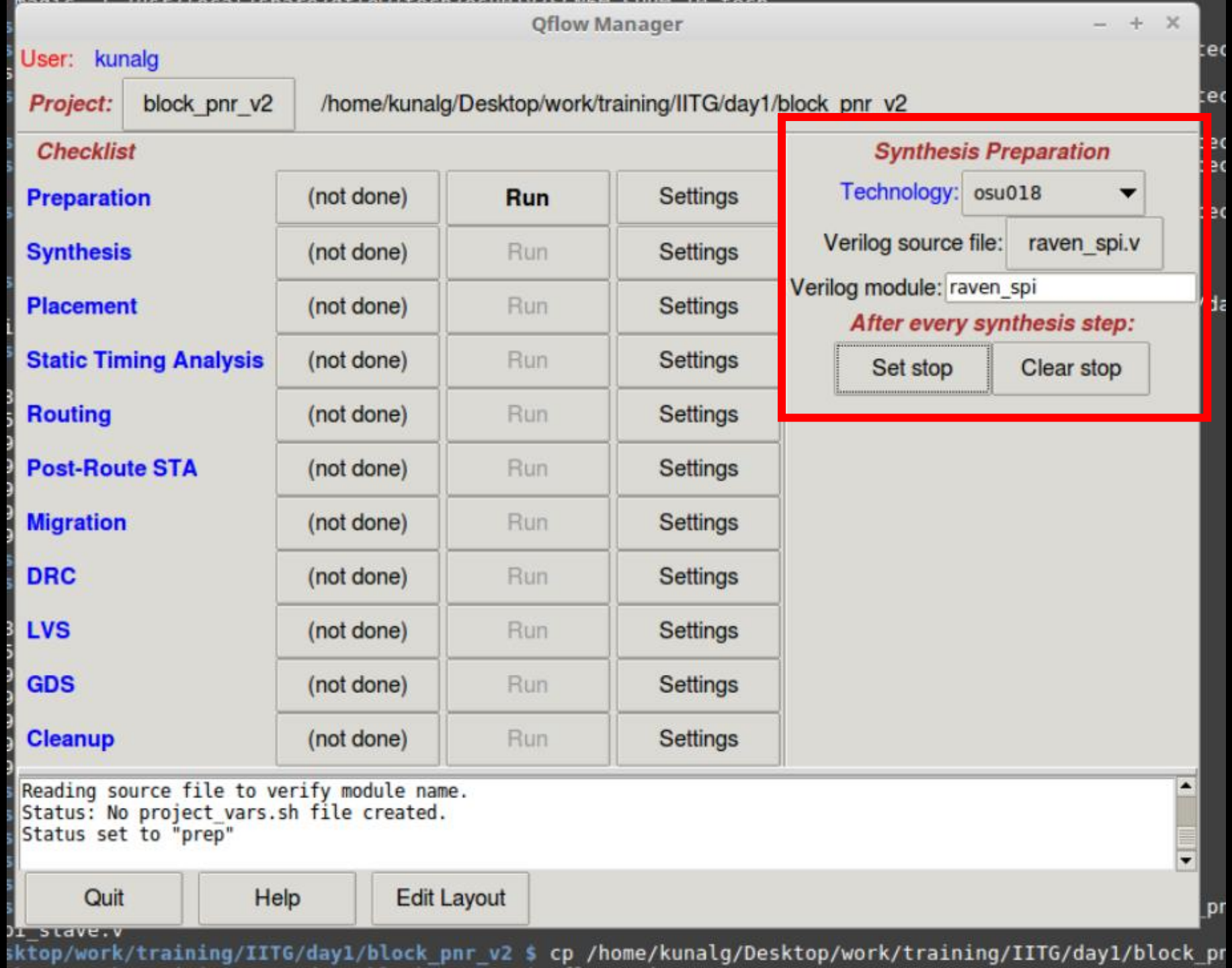
git clone https://github.com/kunalg123/raven_spi_hierarchical_physical_design

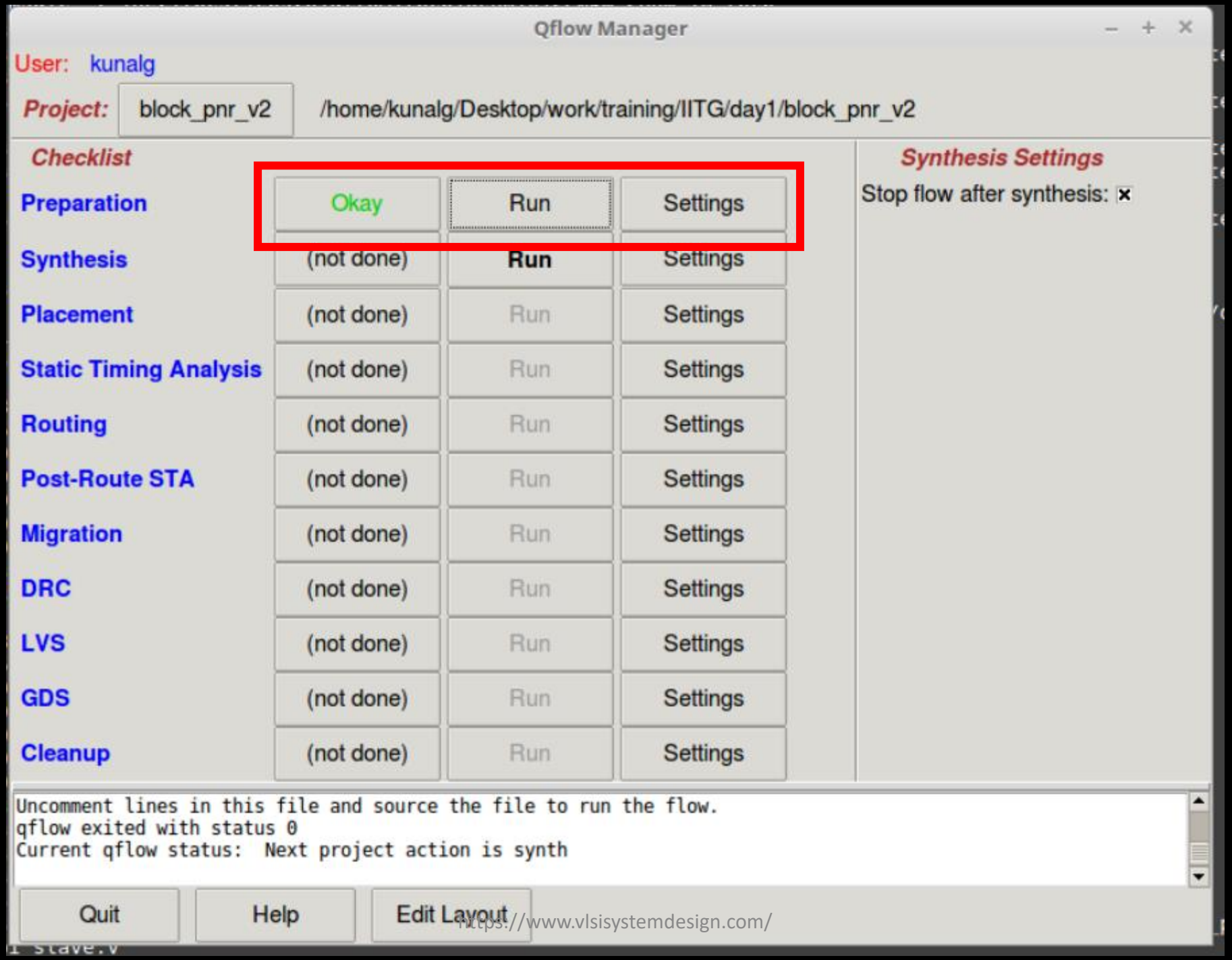
Demo

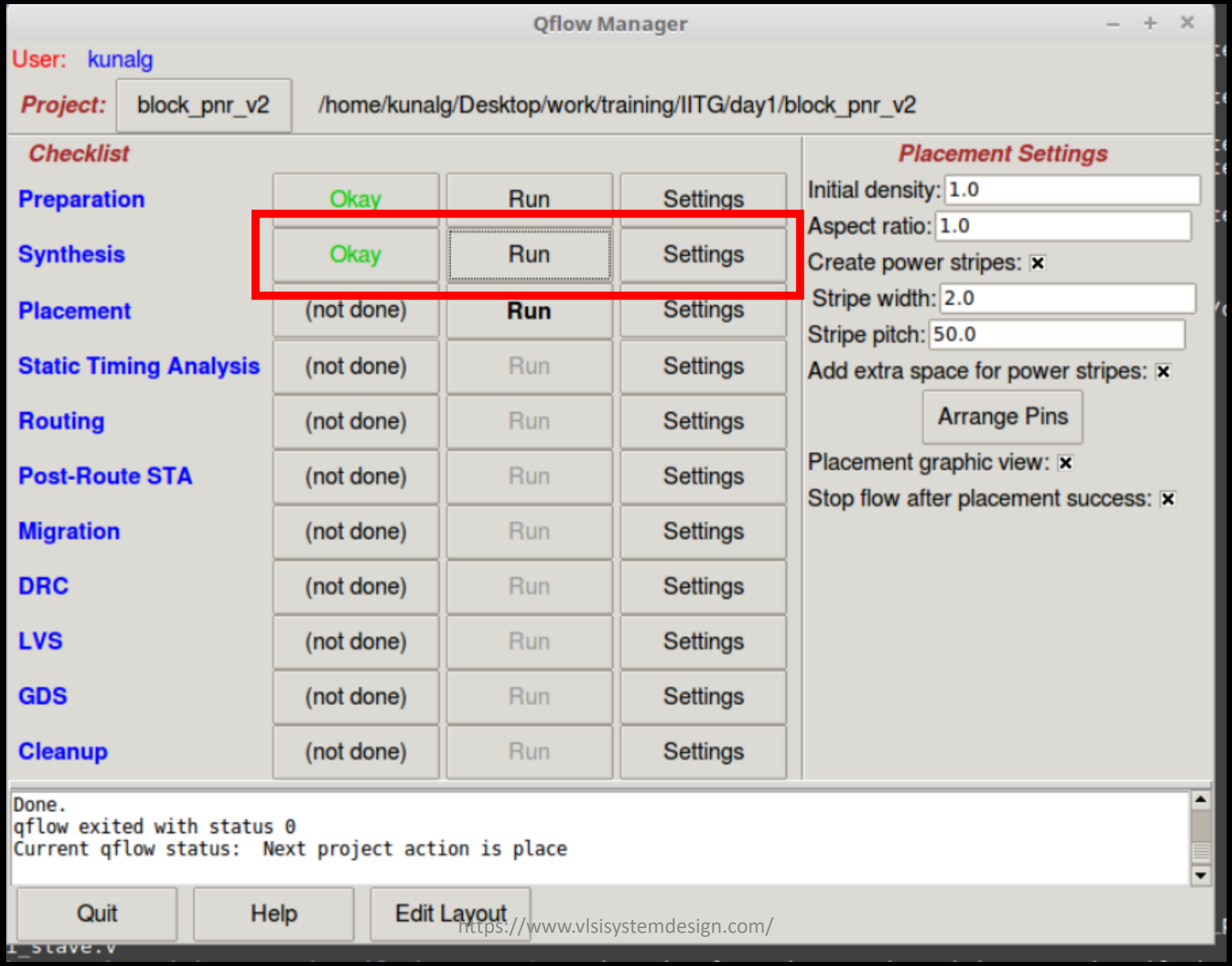
```
mkdir source synthesis layout
```

```
cp /home/kunalg/Desktop/work/training/IITG/day1/block_pnr/source/*.v source/.
qflow gui &
```


Qflow Manager
User: kunalg
Project: block_pnr_v2
/home/kunalg/Desktop/work/training/IITG/day1/block_pnr_v2
Checklist
Preparation (not done) Run Settings
Synthesis (not done) Run Settings
Placement (not done) Run Settings
Static Timing Analysis (not done) Run Settings
Routing (not done) Run Settings
Post-Route STA (not done) Run Settings
Migration (not done) Run Settings
DRC (not done) Run Settings
LVS (not done) Run Settings
GDS (not done) Run Settings
Cleanup (not done) Run Settings
Synthesis Preparation
Technology: osu035
Verilog source file: (no selection)
Verilog module: block_pnr_v2
After every synthesis step:
Set stop
Clear stop
Status: No project_vars.sh file created.
Status set to "prep"
Current qflow status: Next project action is prep
Quit
Help
Edit Layout
qflow gui &


Qflow Manager
User: kunalg
Project: block_pnr_v2
/home/kunalg/Desktop/work/training/IITG/day1/block_pnr_v2
Checklist
Preparation (not done) Run Settings
Synthesis (not done) Run Settings
Placement (not done) Run Settings
Static Timing Analysis (not done) Run Settings
Routing (not done) Run Settings
Post-Route STA (not done) Run Settings
Migration (not done) Run Settings
DRC (not done) Run Settings
LVS (not done) Run Settings
GDS (not done) Run Settings
Cleanup (not done) Run Settings
Synthesis Preparation
Technology: osu018
Verilog source file: raven_spi.v
Verilog module: raven_spi
After every synthesis step:
Set stop
Clear stop
Reading source file to verify module name.
Status: No project_vars.sh file created.
Status set to "prep"
Quit
Help
Edit Layout
ktop/work/training/IITG/day1/block_pnr_v2 $ cp /home/kunalg/Desktop/work/training/IITG/day1/block_pr


Qflow Manager
User: kunalg
Project: block_pnr_v2
/home/kunalg/Desktop/work/training/IITG/day1/block_pnr_v2
Checklist
Synthesis Settings
Stop flow after synthesis:
Preparation
Okay
Run
Settings
Synthesis
(not done)
Run
Settings
Placement
(not done)
Run
Settings
Static Timing Analysis
(not done)
Run
Settings
Routing
(not done)
Run
Settings
Post-Route STA
(not done)
Run
Settings
Migration
(not done)
Run
Settings
DRC
(not done)
Run
Settings
LVS
(not done)
Run
Settings
GDS
(not done)
Run
Settings
Cleanup
(not done)
Run
Settings
Uncomment lines in this file and source the file to run the flow.
qflow exited with status 0
Current qflow status: Next project action is synth
Quit
Help
Edit Layout
https://www.vlsisystemdesign.com/



```
module raven_spi (RST, SCK, SDI, CSB, trap, mask_rev_in, SDO, sdo_enb, xtal_ena, reg_ena, pll_vco_ena, pll_cp_ena, pll_bias_ena, pll_trim, pll_bypass, irq, re
set, mfgr_id, prod_id, mask_rev);

input RST;
input SCK;
input SDI;
input CSB;
input trap;
output SDO;
output sdo_enb;
output xtal_ena;
output reg_ena;
output pll_vco_ena;
output pll_cp_ena;
output pll_bias_ena;
output pll_bypass;
output irq;
output reset;
input [3:0] mask_rev_in;
output [3:0] pll_trim;
output [11:0] mfgr_id;
output [7:0] prod_id;
output [3:0] mask_rev;

wire vdd = 1'b1;
wire gnd = 1'b0;

CLKBUF1 CLKBUF1_1 ( .A(SCK), .Y(SCK_bF_buf5) );
CLKBUF1 CLKBUF1_2 ( .A(SCK), .Y(SCK_bF_buf4) );
CLKBUF1 CLKBUF1_3 ( .A(SCK), .Y(SCK_bF_buf3) );
CLKBUF1 CLKBUF1_4 ( .A(SCK), .Y(SCK_bF_buf2) );
CLKBUF1 CLKBUF1_5 ( .A(SCK), .Y(SCK_bF_buf1) );
CLKBUF1 CLKBUF1_6 ( .A(SCK), .Y(SCK_bF_buf0) );
BUFX4 BUFX4_1 ( .A(U1_state_0_), .Y(U1_state_0_bF_buf3_) );
BUFX4 BUFX4_2 ( .A(U1_state_0_), .Y(U1_state_0_bF_buf2_) );
BUFX4 BUFX4_3 ( .A(U1_state_0_), .Y(U1_state_0_bF_buf1_) );
BUFX4 BUFX4_4 ( .A(U1_state_0_), .Y(U1_state_0_bF_buf0_) );
BUFX4 BUFX4_5 ( .A(_213_), .Y(_213__bF_buf5) );
BUFX4 BUFX4_6 ( .A(_213_), .Y(_213__bF_buf4) );
BUFX4 BUFX4_7 ( .A(_213_), .Y(_213__bF_buf3) );
BUFX4 BUFX4_8 ( .A(_213_), .Y(_213__bF_buf2) );
BUFX4 BUFX4_9 ( .A(_213_), .Y(_213__bF_buf1) );
BUFX4 BUFX4_10 ( .A(_213_), .Y(_213__bF_buf0) );
BUFX4 BUFX4_11 ( .A(U1_state_2_), .Y(U1_state_2_bF_buf3_) );
"synthesis/raven_spi.rtlnopwr.v" 414L, 25029C                                 1,1           Top
```