With ULX3S

Presented by Miodrag Milanovic and Jan Dolinaj

Who we are ?

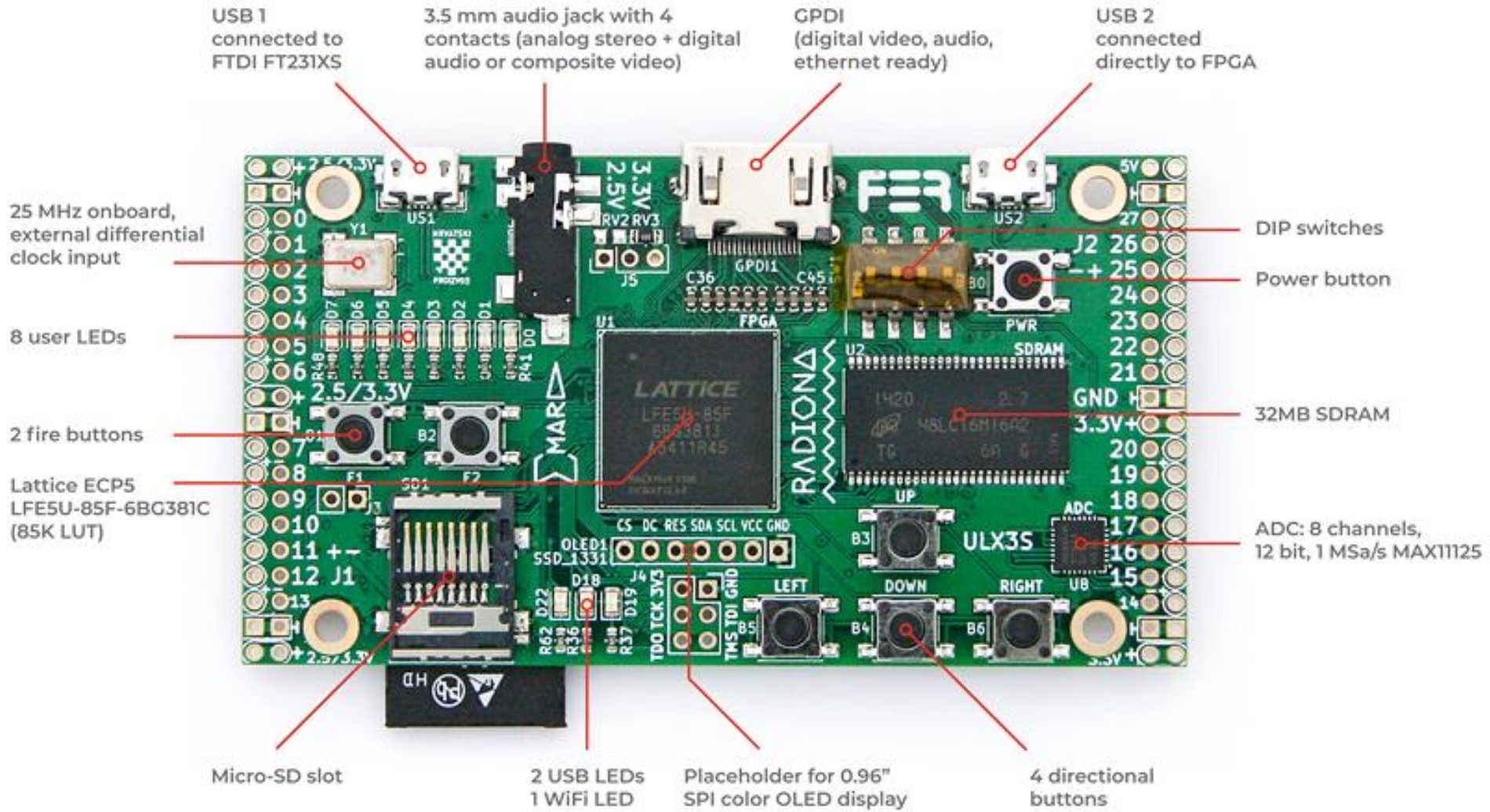
- Engineers
- Hardware enthusiasts
- Software developers
- Very interested in FPGA

- Working with Symbiotic EDA on NextPnR, Yosys, …

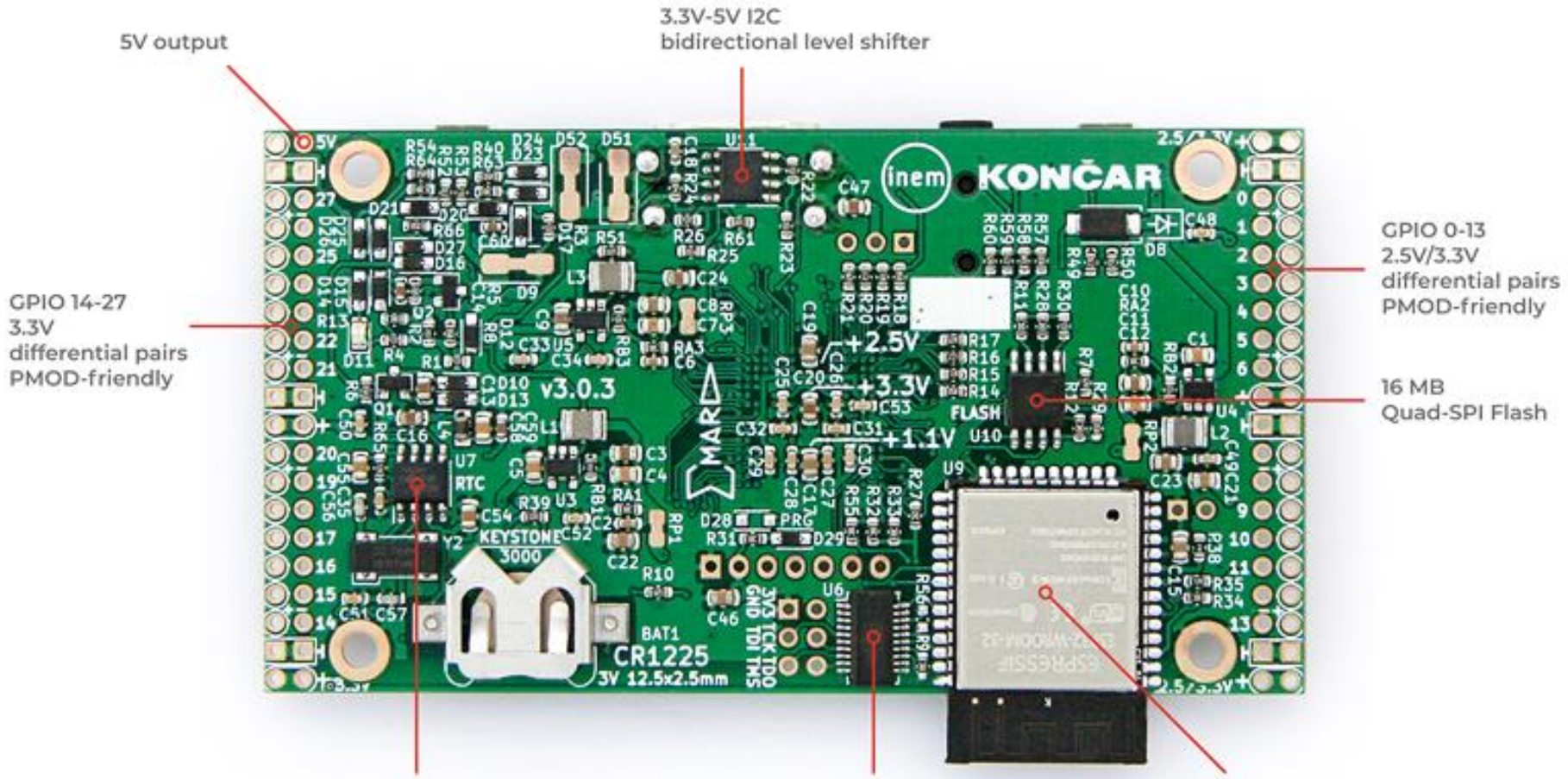
ULX3S

ULX3S - Top

RADIONA ZAGREB MAKERSPACE



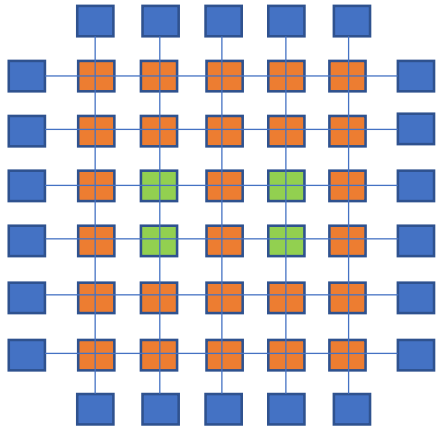
ULX3S-Bottom



FPGA

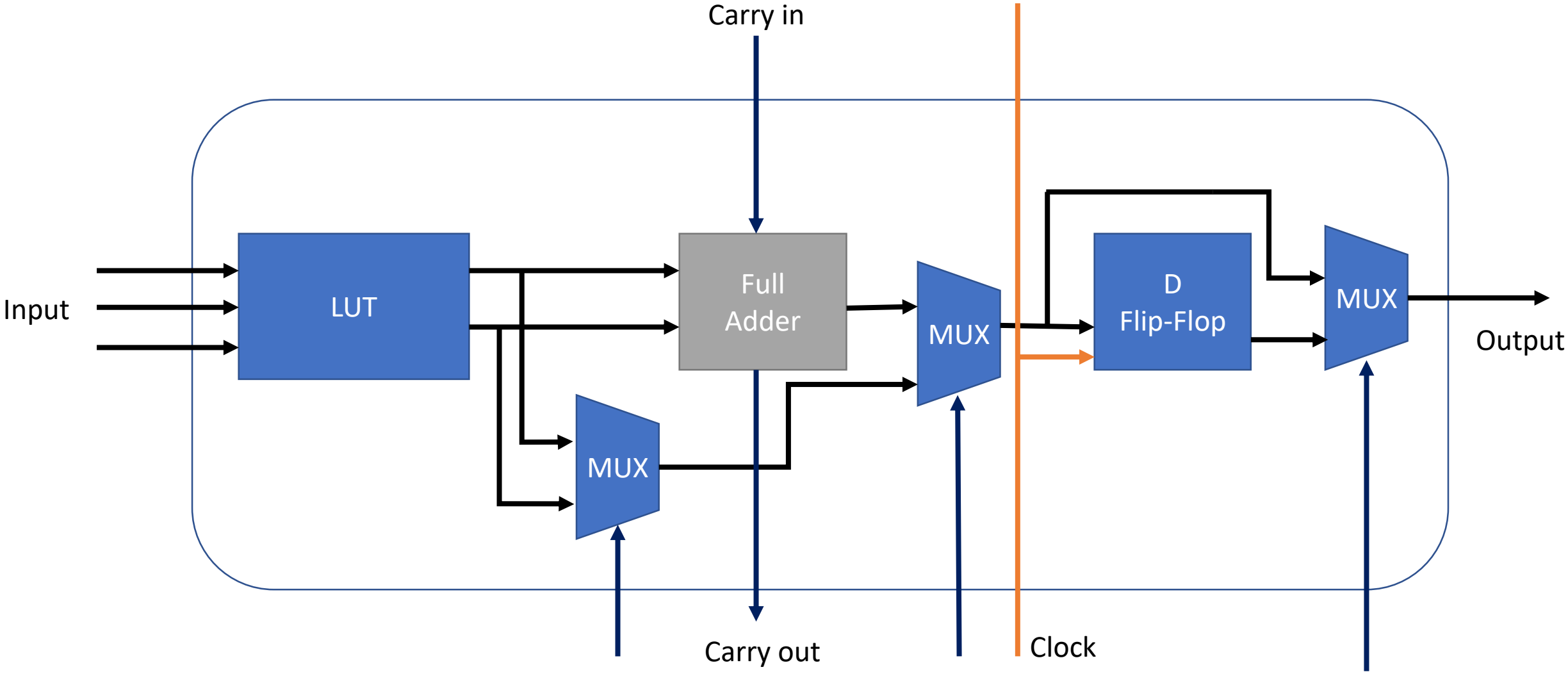
What is FPGA ?

- Field-programmable gate array
- Logic block – LUT + D Flip-Flop + full adder
- I/O block – one per each pin
- Interconnections, clock tree
- Hard block – Block RAM, multipliers, DSP, CPU,…



- Vendors: Xilinx, Intel/Altera, Lattice, Microsemi, Anlogic, AGM, GoWin …

Logic block


Carry in
Input
LUT
Full
Adder
MUX
D
Flip-Flop
MUX
Output
MUX
Carry out
Clock

Lattice ECP5

- Affordable price
- From 12K to 85K
- 5G SERDES
- Open source toolchain
- Lot of nice boards available

TOOLS

How do we program FPGA ?

- HDL - Hardware description language
- VHDL and Verilog

- Analysis: parsing and validation of HDL
- Synthesis: HDL -> netlist
- Place-and-route: netlist -> specific FPGA technology
- Assembler: specific FPGA technology -> bitstream
- Programming: deploying bitstream on device (serial flash memory or SRAM)
- Timing analysis
- Simulation

Open source tools

- Yosys - Verilog synthesis tool by Clifford Wolf
- NextPnR - Place and route tool by Symbiotic EDA team
- Project Trellis – Documenting the Lattice ECP5 bit-stream format by David Shah
- APIO – micro-ecosystem for open FPGAs by Jesús Arroyo Torrens and Juan González (Obijuan)

And more …

- Arachne PnR - Place and route tool by Cotton Seed (ICE40)
- Project IceStorm – Assembler, time analysis and FPGA programming tool by Clifford Wolf and Mathias Lasser (ICE40)
- Icestudio - graphic editor for open FPGAs by Jesús Arroyo Torrens
- Icarus Verilog by Stephen Williams
- Verilator by Wilson Snyder with Duane Galbi and Paul Wasson
- FuseSoC - package manager and a set of build tools for FPGA/ASIC development by Olof Kindgren

YOSYS

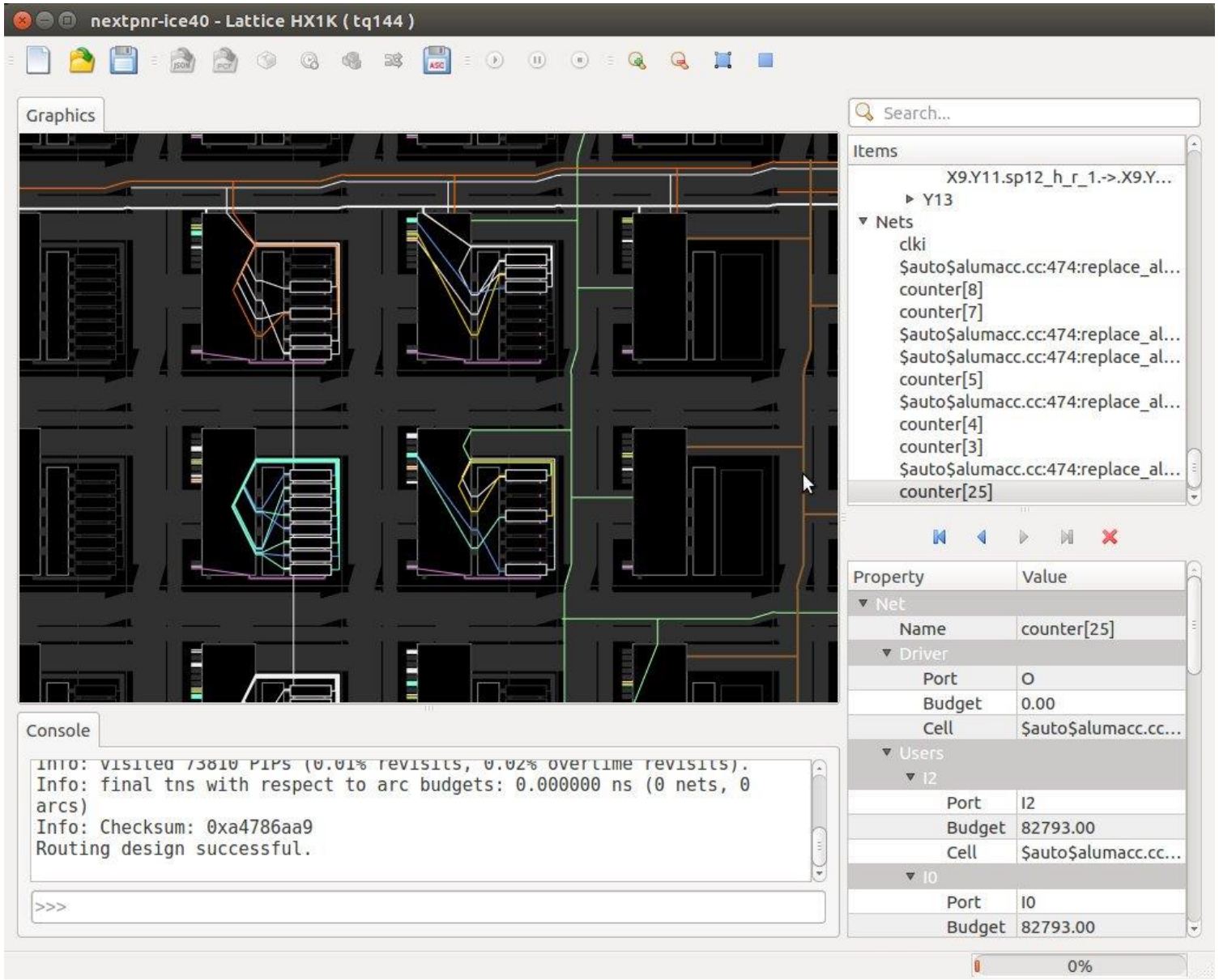
- This is a framework for RTL synthesis tools. It currently has extensive Verilog-2005 support and provides a basic set of synthesis algorithms for various application domains.
- Yosys can be adapted to perform any synthesis job by combining the existing passes (algorithms) using synthesis scripts and adding additional passes as needed by extending the yosys C++ code base.
- Yosys is free software licensed under the ISC license (a GPL compatible license that is similar in terms to the MIT license or the 2-clause BSD license).

NextPnR

NextPnR aims to be a vendor neutral, timing driven, FOSS FPGA place and route tool.

Currently nextpnr supports:

- Lattice iCE40 devices supported by Project IceStorm
- Lattice ECP5 devices supported by Project Trellis
- A "generic" back-end for user-defined architectures



Symbiotic EDA

Project Trellis

LFE5U-25F Tilegrid

APIO

apio build

```
[FPGA] C:\msys64\src\fpga-odysseus\tutorials\01-Basics\03-Counter>apio build
[03/07/19 14:05:14] Processing ulx3s-45f
--------------------------------------------------------------------------------------------------------------------
yosys -p "synth_ecp5 -json hardware.json" -q top.v
Warning: Wire top.cnt has an unprocessed 'init' attribute.
nextpnr-ecp5 --45k --package CABGA381 --json hardware.json --textcfg hardware.config --lpf ulx3s.lpf -q --timing-allow-fail
ecppack --db C:\msys64\opt\homedir\.apio\packages\toolchain-ecp5\share\trellis\database hardware.config hardware.bit
============================================================================ [SUCCESS] Took 4.36 seconds ==============================
```

apio verify

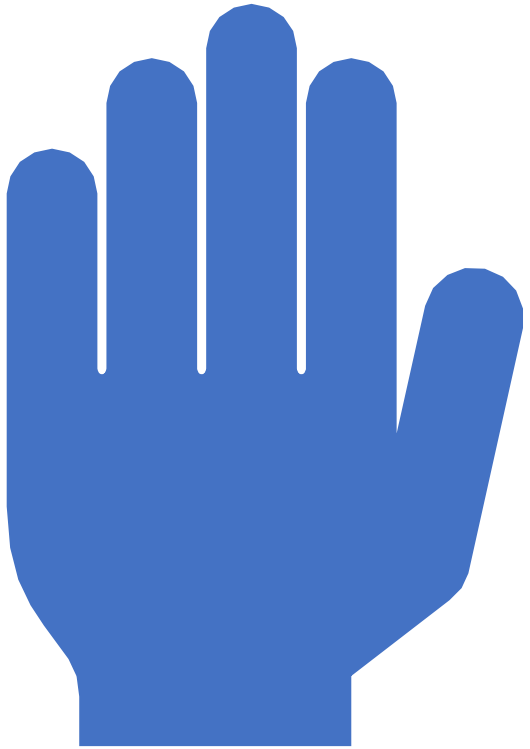
```
[FPGA] C:\msys64\src\fpga-odysseus\tutorials\01-Basics\03-Counter>apio verify
iverilog -o hardware.out -D VCD_OUTPUT= C:\msys64\opt\homedir\.apio\packages\toolchain-yosys\share\yosys/ecp5/cells_sim.v top.v
============================================================================ [SUCCESS] Took 0.83 seconds ==============================
```

apio lint

```
[FPGA] C:\msys64\src\fpga-odysseus\tutorials\01-Basics\03-Counter>apio lint
verilator --lint-only -v C:\msys64\opt\homedir\.apio\packages\toolchain-yosys\share\yosys/ecp5/cells_sim.v top.v
============================================================================ [SUCCESS] Took 0.69 seconds ==============================
```

apio sim

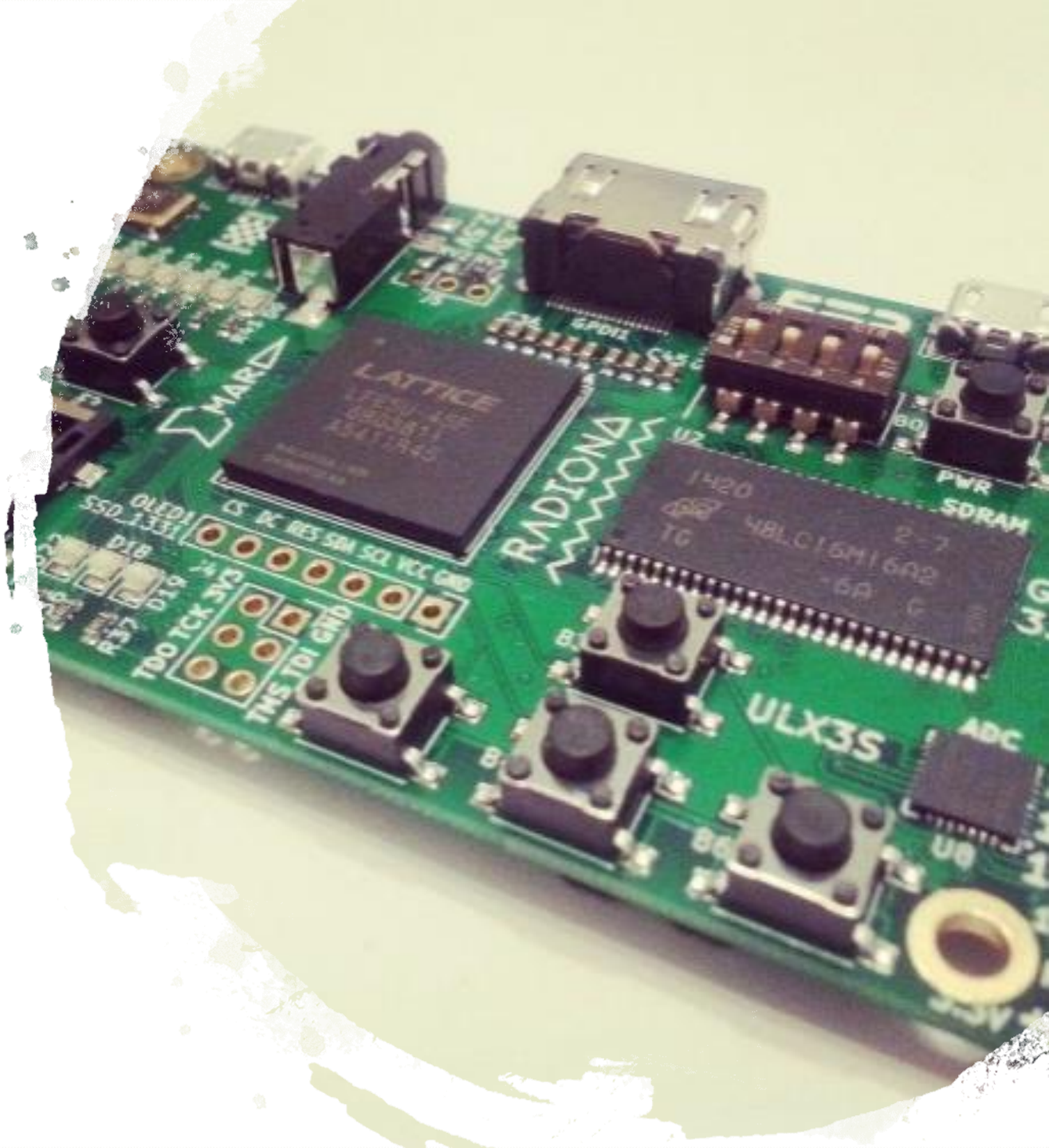
HANDS ON WORKSHOP



Set up for your board

- Enter tutorials folder of fpga-odysseus
- **apio init –board ulx3s-12f**
- If board is 1.7 revision :

  cp ulx3s_v17patch.lpf ulx3s.lpf
- All projects use symbolic links to these files



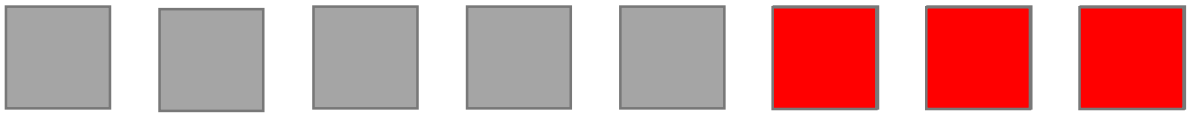
BASICS

Init

```
module top (    output wifi_gpio0 );
    assign wifi_gpio0 = 1'b1;
endmodule
```

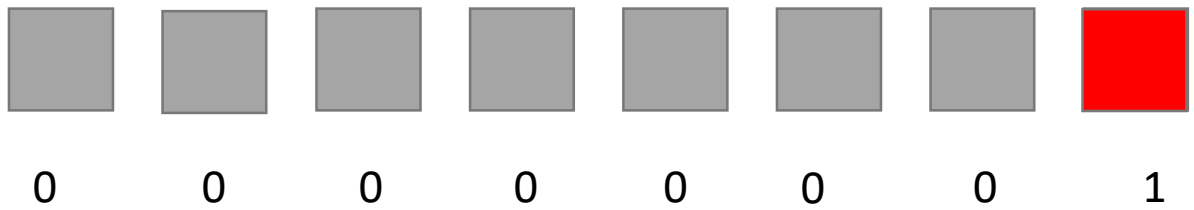
- apio build
- apio upload
- apio clean

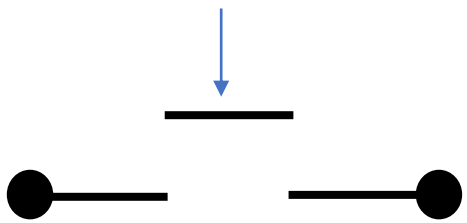
LED

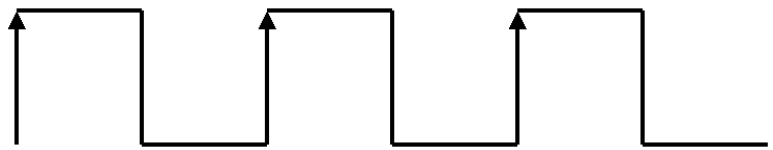


0 0 0 0 0 1 1 1

Button

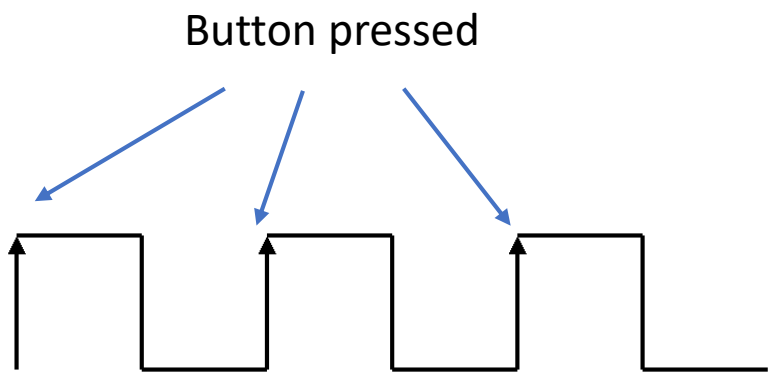






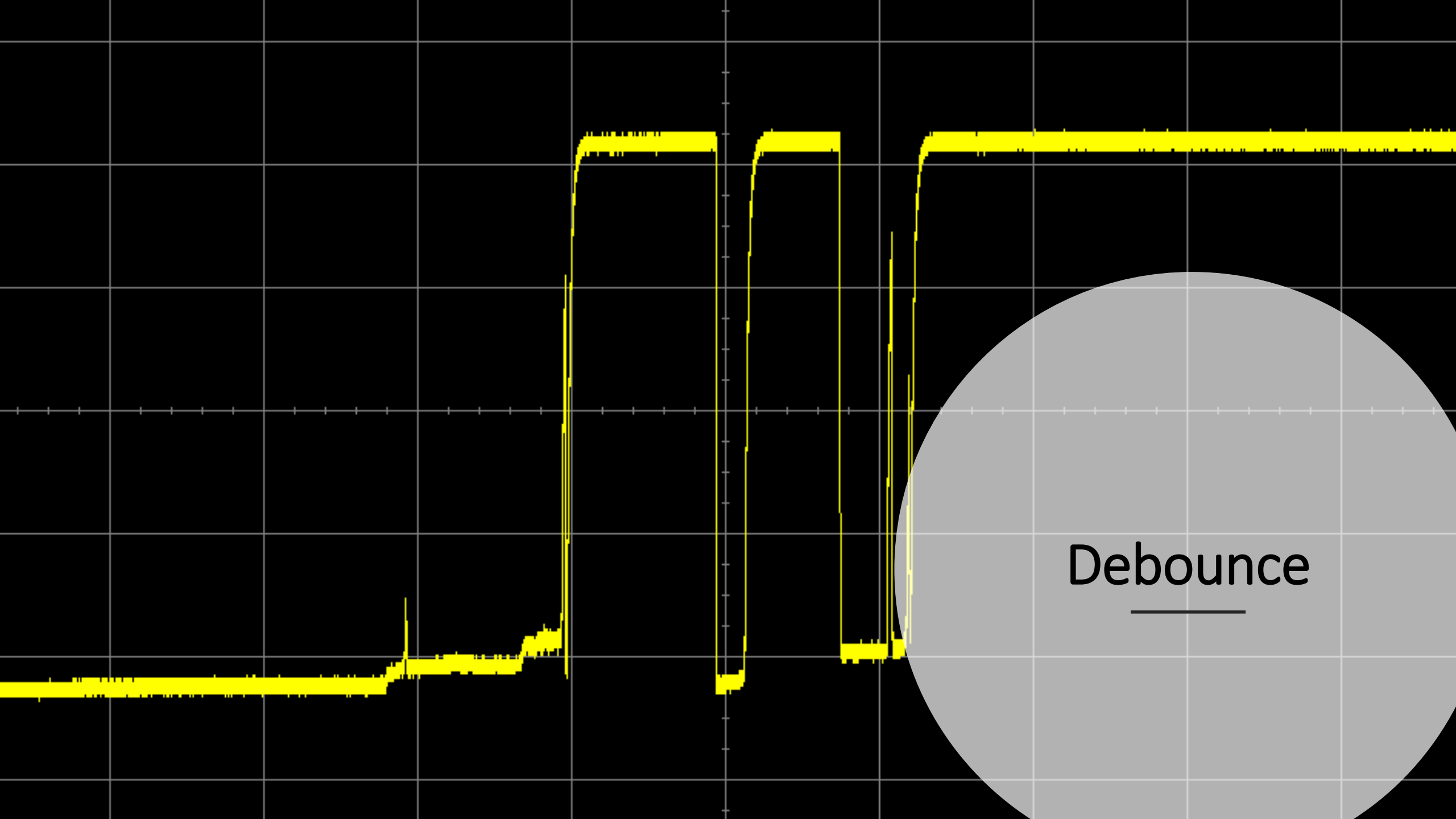
Counter

```
reg [31:0] cnt = 0;

always @(posedge clk_25mhz)
begin
cnt <= cnt + 1;
end

assign led = cnt[31:24];
```



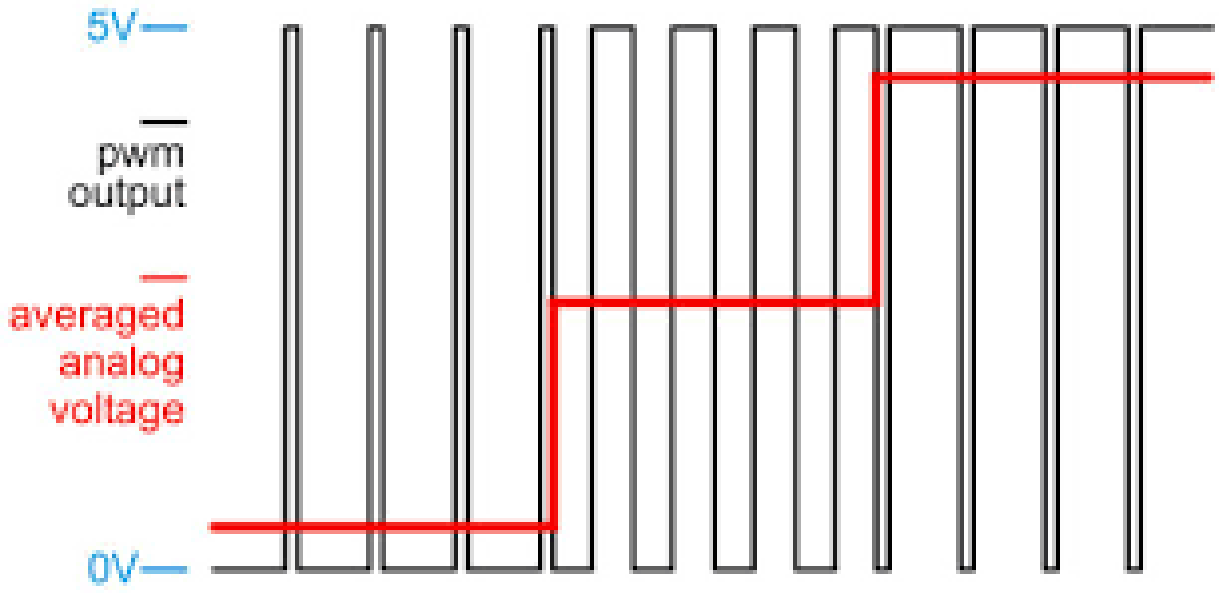
Trigger

```
reg [7:0] cnt = 0;

always @(posedge btn[1])
begin
cnt <= cnt + 1;
end

assign led = cnt;
```


Debounce

AUDIO

Theory



- Digital – Analog Converter
- PWM
- We will change frequency of change, but keep 50% duty cycle

Tone frequency

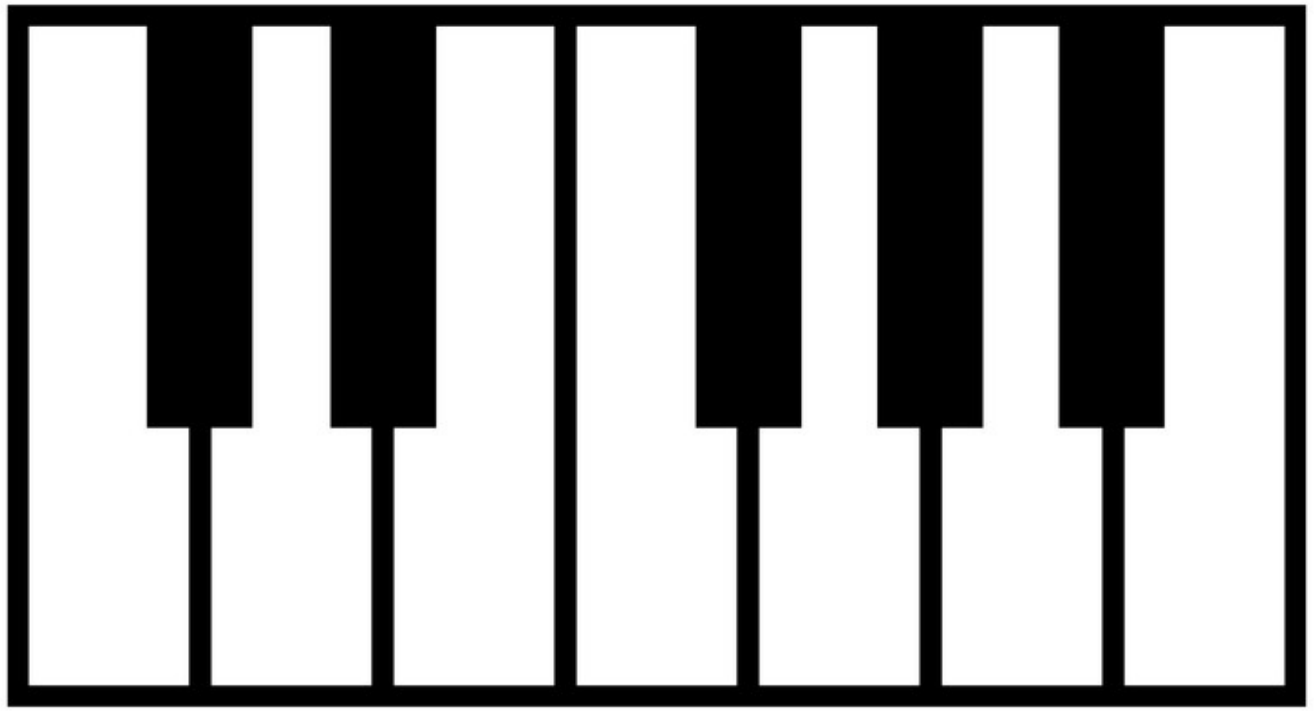
	Octaves								
	1	2	3	4	5	6	7	8	9
C	32.70	65.41	130.81	261.63	523.25	1,046.50	2,093.00	4,186.01	8,372.02
C#	34.65	69.30	138.59	277.18	554.37	1,108.73	2,217.46	4,434.92	8,869.84
D	36.71	73.42	146.83	293.66	587.33	1,174.66	2,349.32	4,698.64	9,397.27
D#	38.89	77.78	155.56	311.13	622.25	1,244.51	2,489.02	4,978.03	9,956.06
E	41.20	82.41	164.81	329.63	659.26	1,318.51	2,637.02	5,274.04	10,548.08
F	43.65	87.31	174.61	349.23	698.46	1,396.91	2,793.83	5,587.65	11,175.30
F#	46.25	92.50	185.00	369.99	739.99	1,479.98	2,959.96	5,919.91	11,839.82
G	49.00	98.00	196.00	392.00	783.99	1,567.98	3,135.96	6,271.93	12,543.85
G#	51.91	103.83	207.65	415.30	830.61	1,661.22	3,322.44	6,644.88	13,289.75
A	55.00	110.00	220.00	440.00	880.00	1,760.00	3,520.00	7,040.00	14,080.00
A#	58.27	116.54	233.08	466.16	932.33	1,864.66	3,729.31	7,458.62	14,917.24
B	61.74	123.47	246.94	493.88	987.77	1,975.53	3,951.07	7,902.13	15,804.27

Siren

```
parameter TONE_A4 = 25000000/440/2;
parameter TONE_A5 = 25000000/880/2;

always @(posedge clk_25mhz)
if(counter==26'b0)
begin
        counter <= (tone[23] ? TONE_A4-1 : TONE_A5-1 );
        audio_l <= ~audio_l;
        audio_r <= ~audio_r;
end
else
        counter <= counter-1;
```

A4 A4 A4 A5 A5 A5 A4 A4 A4 A5 A5 A5

Piano

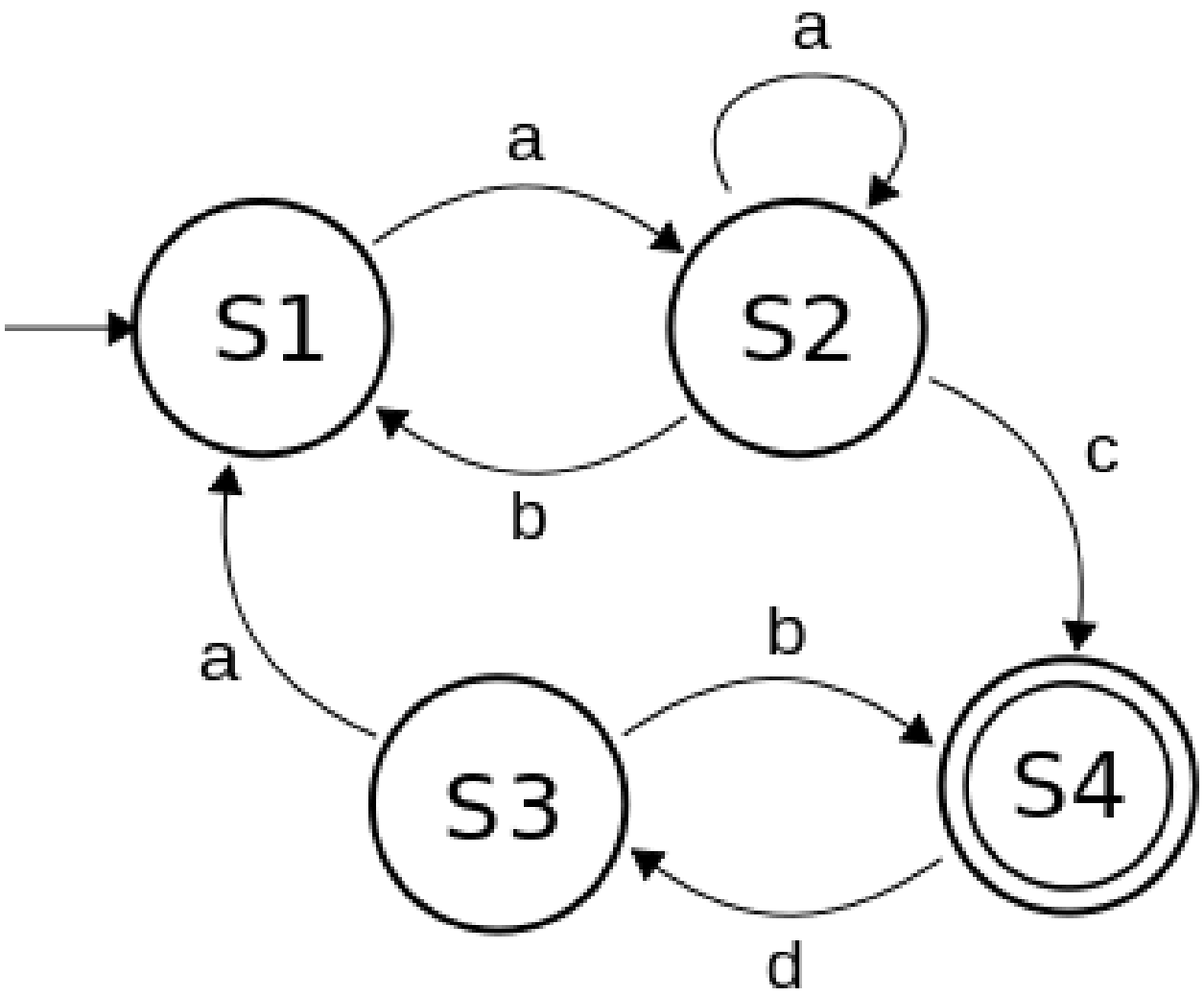


Button	Frequency
6	440 A4
5	494 B4
4	523 C5
3	587 D5
2	659 E5
1	698 F5

PROTOCOLS

FSM

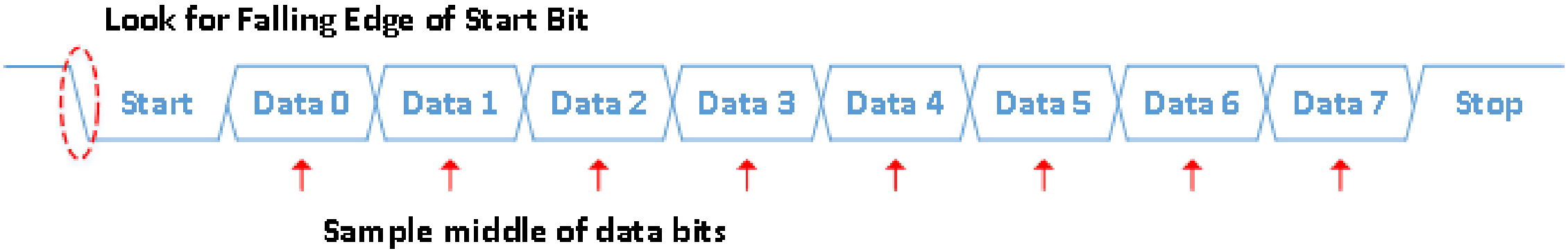
Finite State Machines



Finite State Machines

Mealy Machine	Moore Machine
Output depends both upon the present state and the present input	Output depends only upon the present state.
Generally, it has fewer states than Moore Machine.	Generally, it has more states than Mealy Machine.
The value of the output function is a function of the transitions and the changes, when the input logic on the present state is done.	The value of the output function is a function of the current state and the changes at the clock edges, whenever state changes occur.
Mealy machines react faster to inputs. They generally react in the same clock cycle.	In Moore machines, more logic is required to decode the outputs resulting in more circuit delays. They generally react one clock cycle later.

Taken from www.tutorialspoint.com



Serial protocol

Image taken from nandland.com

Serial Transmit

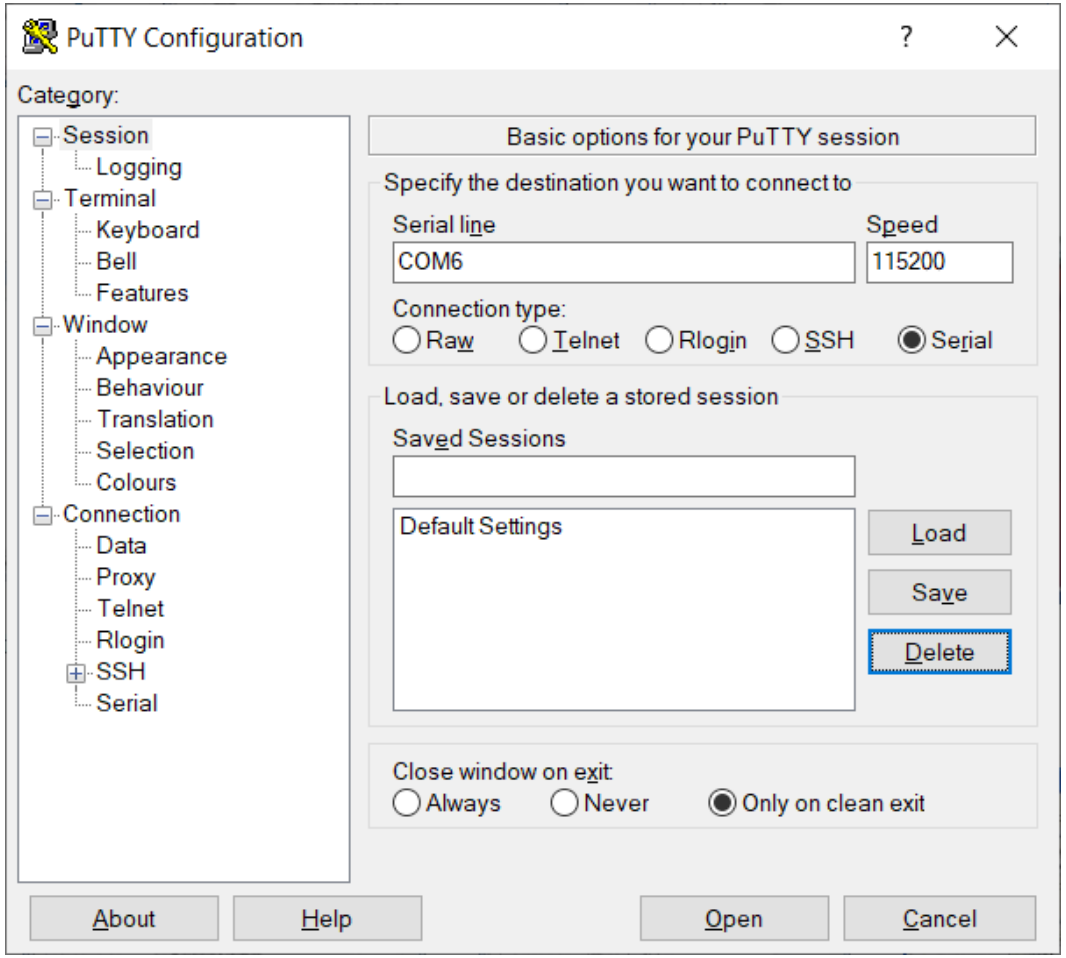
data = 8'd65

cfg_divider = 25000000/115200

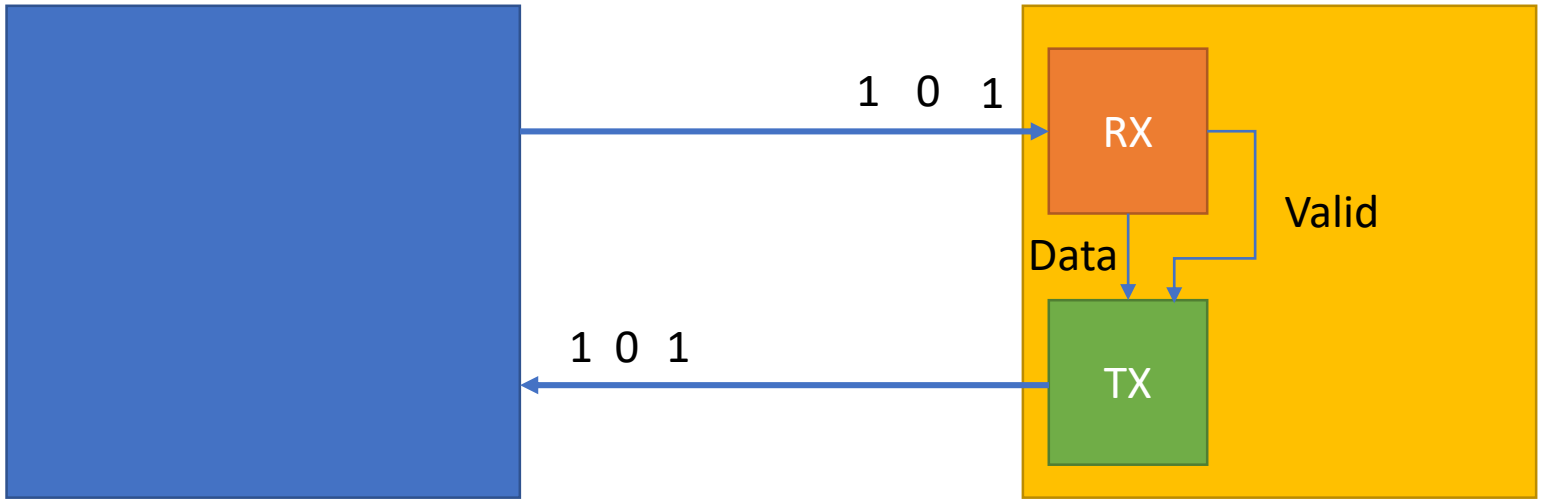
```
send_pattern <= {1'b1, data, 1'b0};
send_bitcnt <= 10;
send_divcnt <= 0;

send_pattern <= {1'b1, send_pattern[9:1]};
send_bitcnt <= send_bitcnt - 1;
send_divcnt <= 0;
```

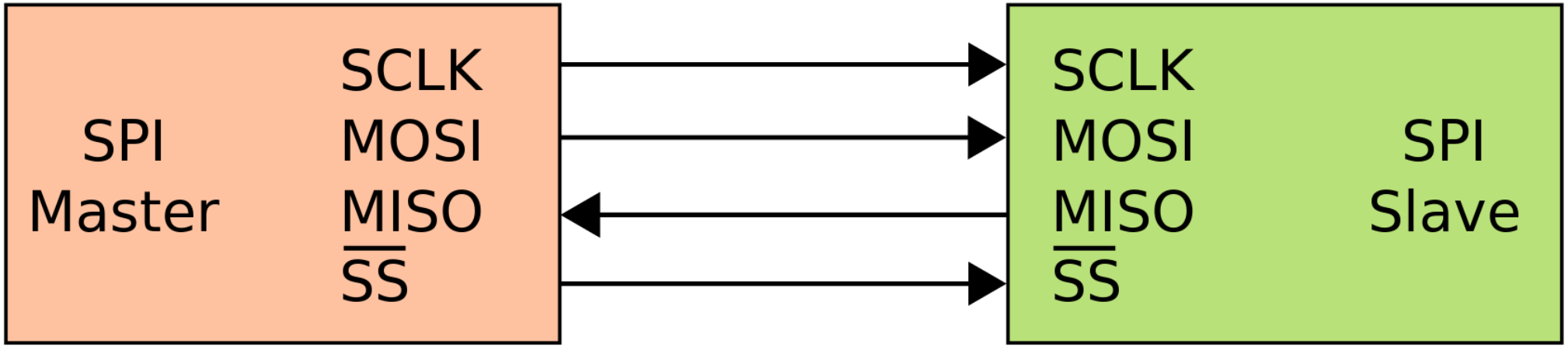
Serial connection

- screen /dev/ttyUSB0 115200
- Putty or anything similar

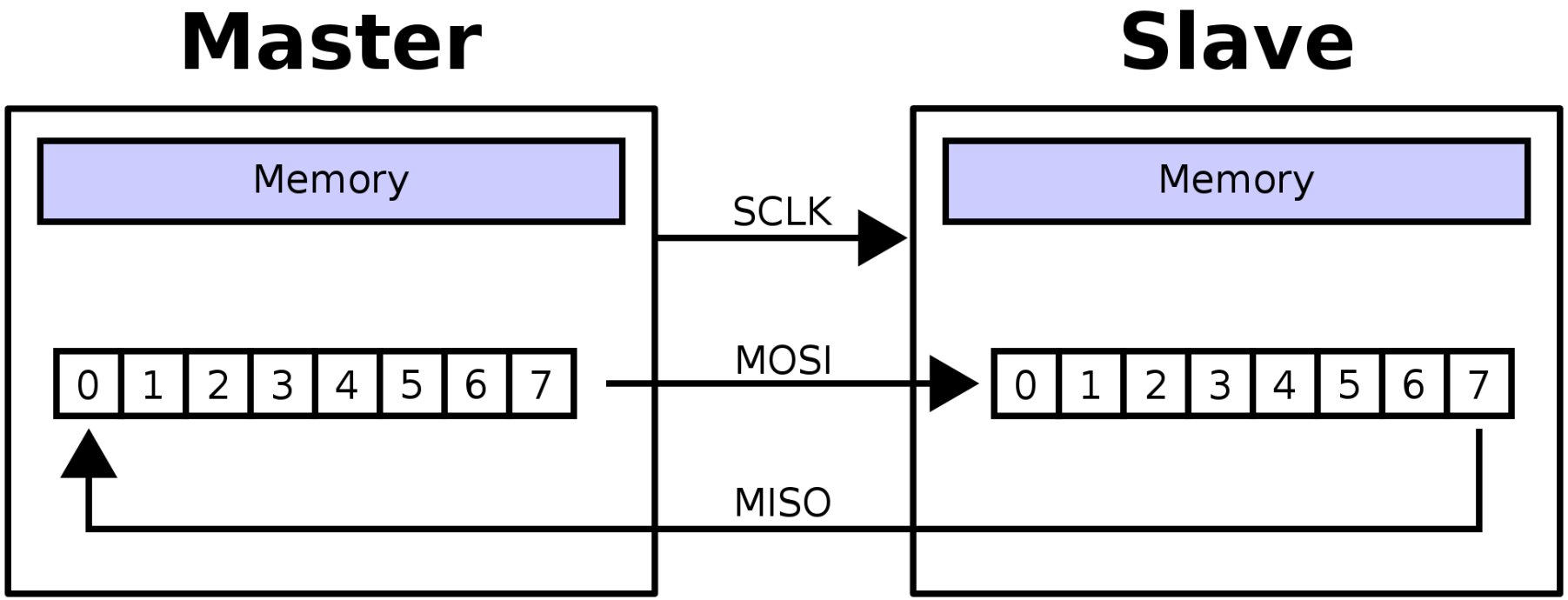


Echo





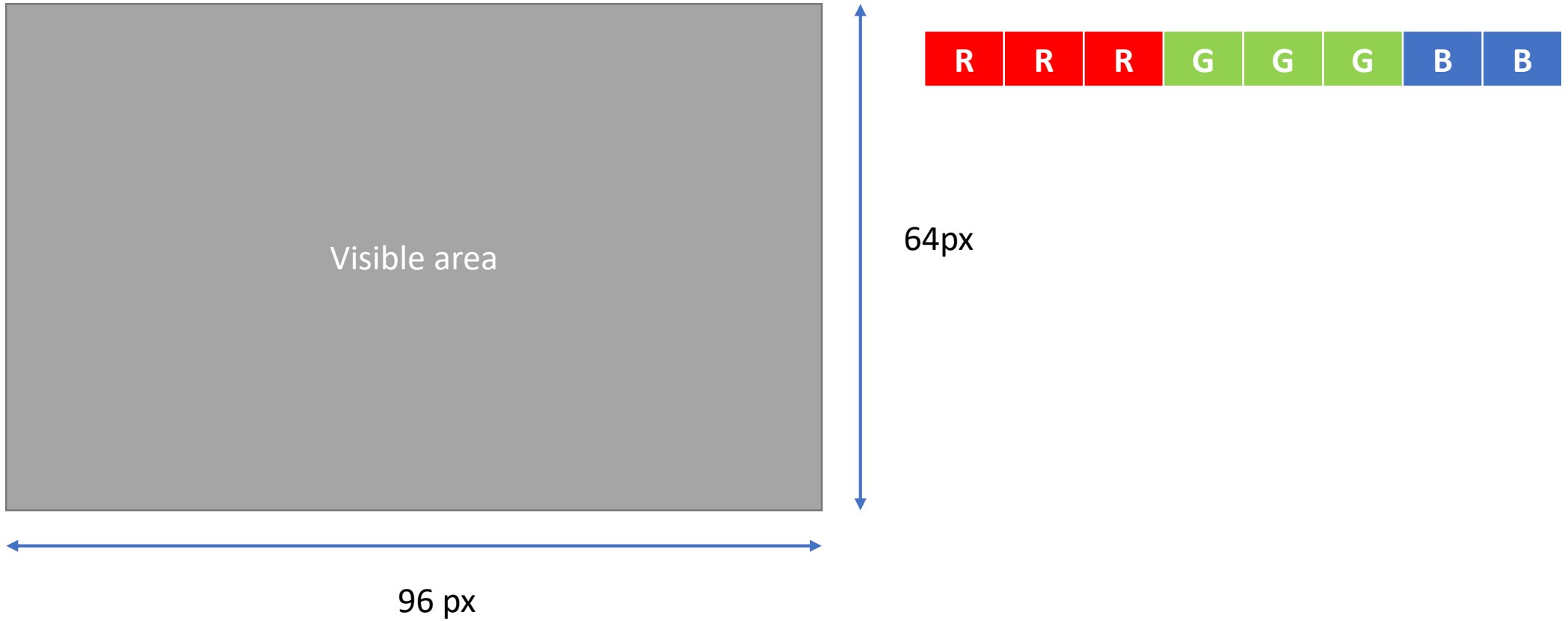
SPI



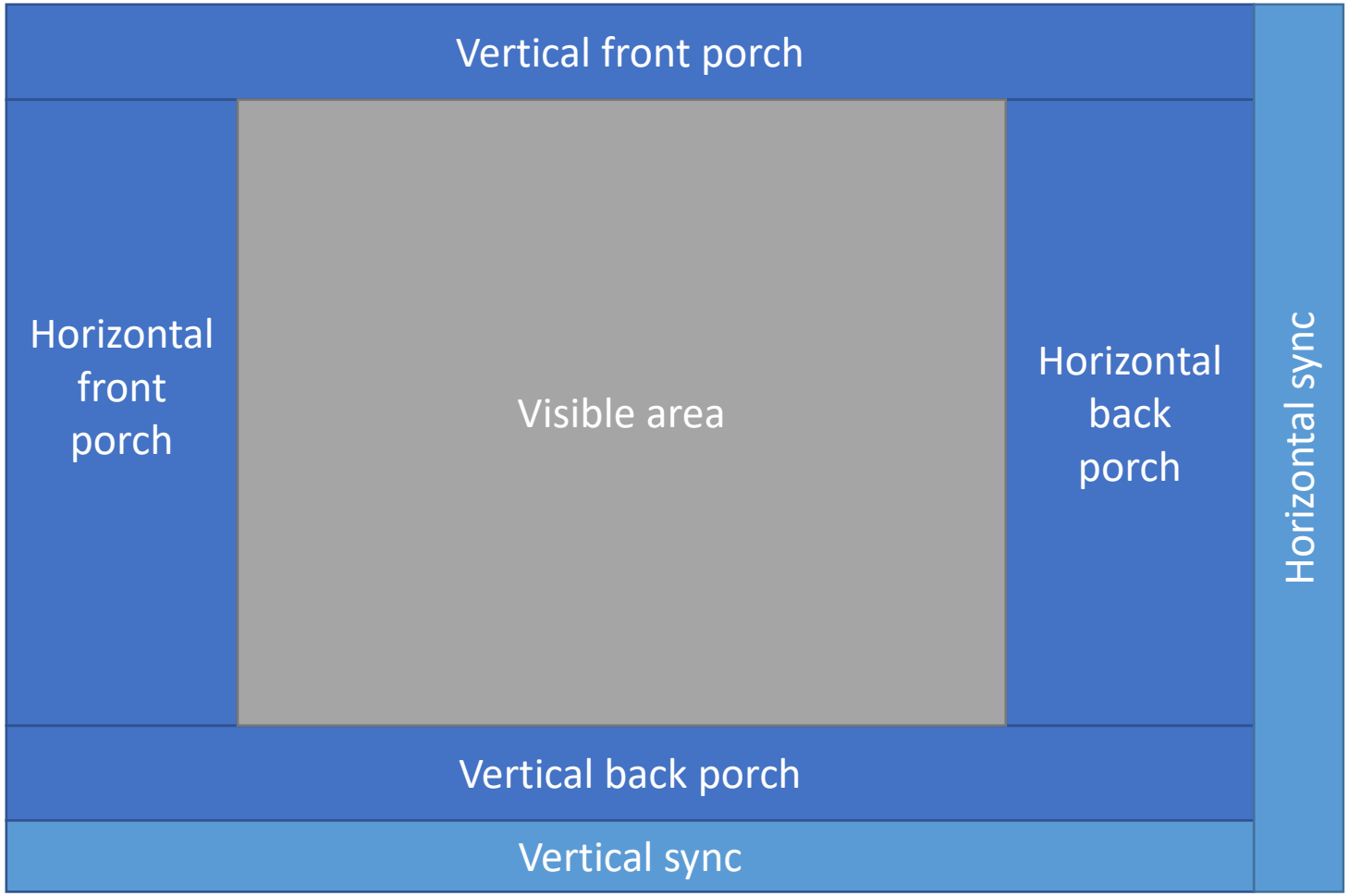
SPI data transfer

VIDEO

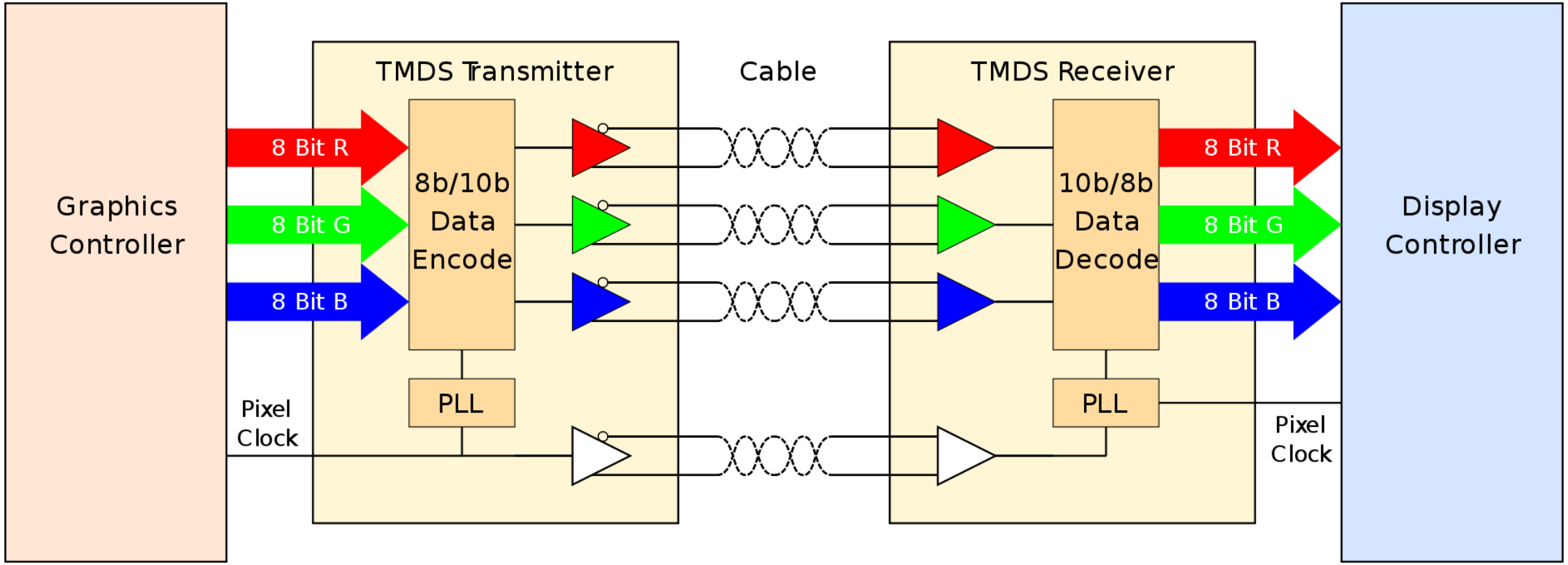
SPI Video



VGA Video


Vertical front porch
Horizontal front porch
Visible area
Horizontal back porch
Horizontal sync
Vertical back porch
Vertical sync

HDMI


Graphics Controller
TMDS Transmitter
8 Bit R
8 Bit G
8 Bit B
8b/10b Data Encode
PLL
Pixel Clock
Cable
TMDS Receiver
10b/8b Data Decode
PLL
8 Bit R
8 Bit G
8 Bit B
Pixel Clock
Display Controller

Video block

```
spi_video video(
  .clk(clk),
  .oled_csn(oled_csn),
  .oled_clk(oled_clk),
  .oled_mosi(oled_mosi),
  .oled_dc(oled_dc),
  .oled_resn(oled_resn),
  .x(x),
  .y(y),
  .color(color)
);
```

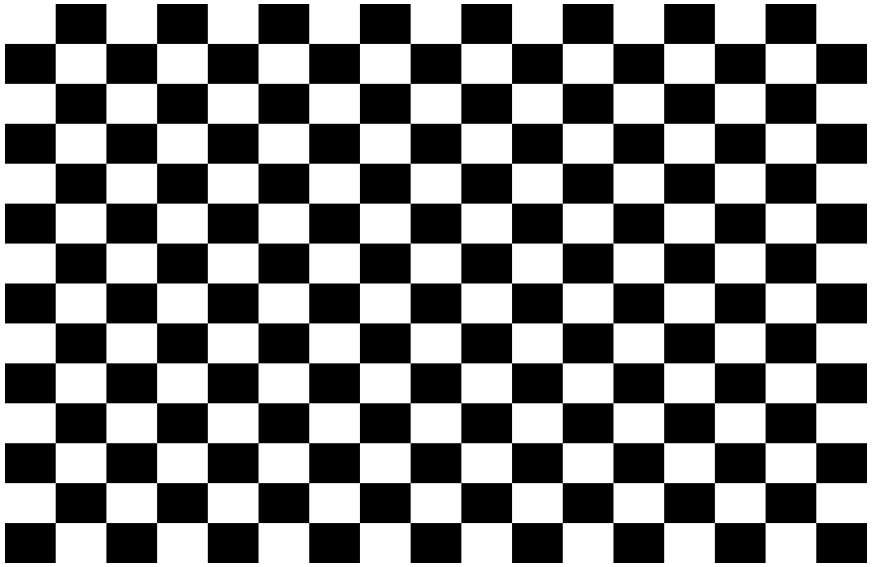
```
hdmi_video hdmi_video
(
    .clk_25mhz(clk_25mhz),
    .x(x),
    .y(y),
    .color(color),
    .gpdi_dp(gpdi_dp),
    .gpdi_dn(gpdi_dn)
);
```

One color screen

assign color = 8'h02;

assign color = 24’h0000ff;

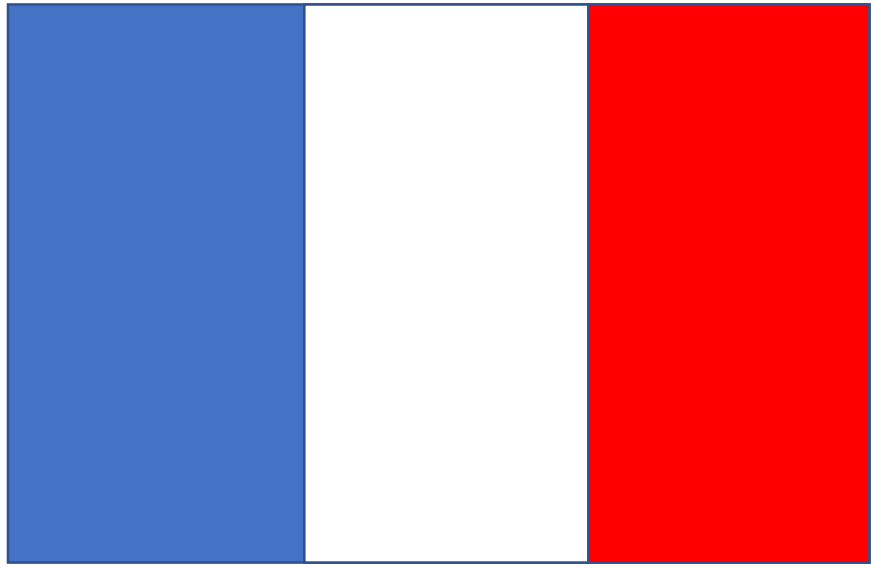
Checkers



```
assign color = x[3] ^ y[3] ? 8'hff : 8'h00;
```

```
assign color = x[4] ^ y[4] ? 24'hffffff : 24'h000000;
```

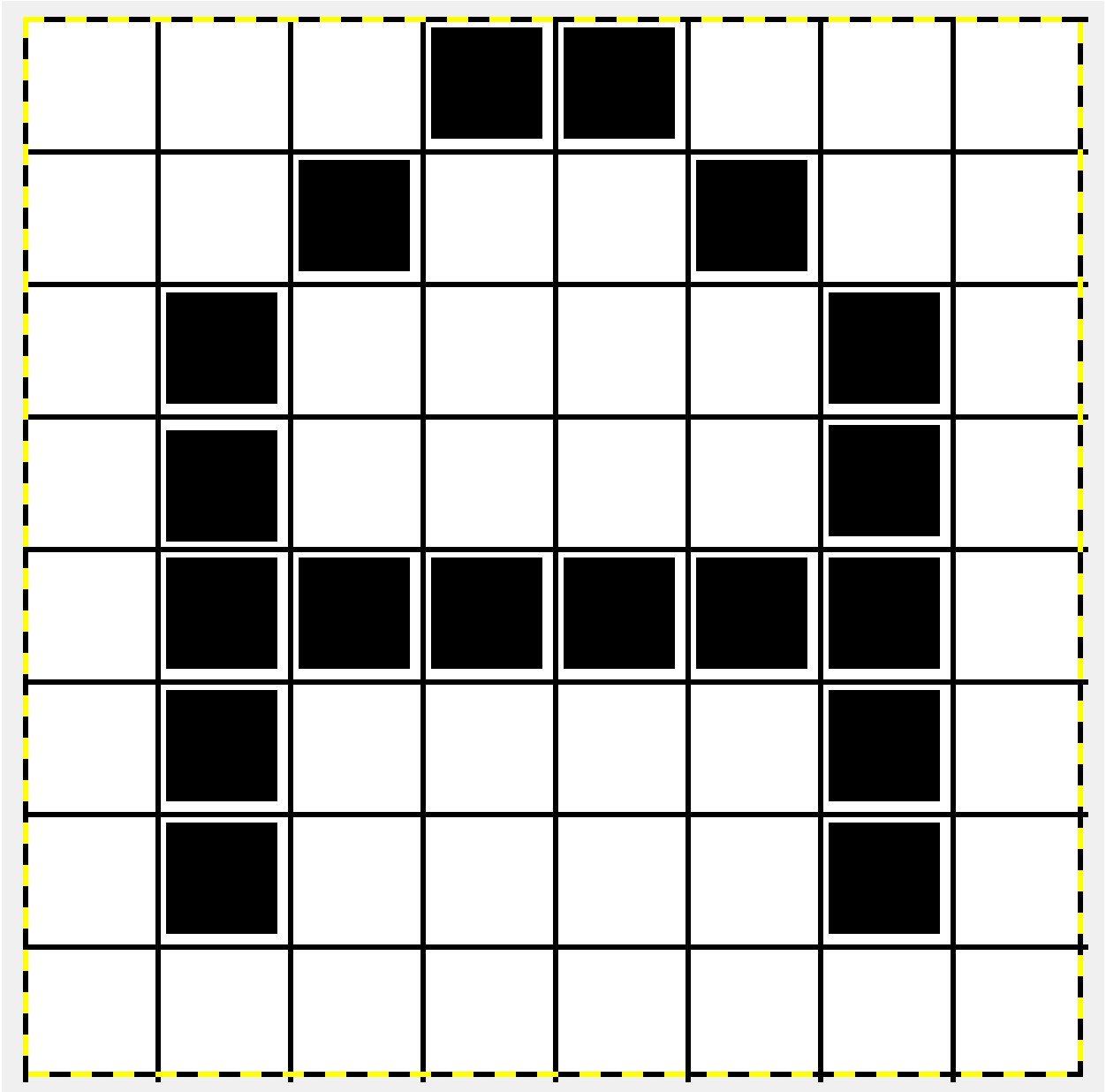
Tricolore



```
assign color = (x<32) ? 8'h02 : (x<64) ? 8'hff : 8'he0;
```

```
assign color = (x<213) ? 24'hff0000 : (x<426) ? 24'hffffff : 24'h0000ff;
```

Text



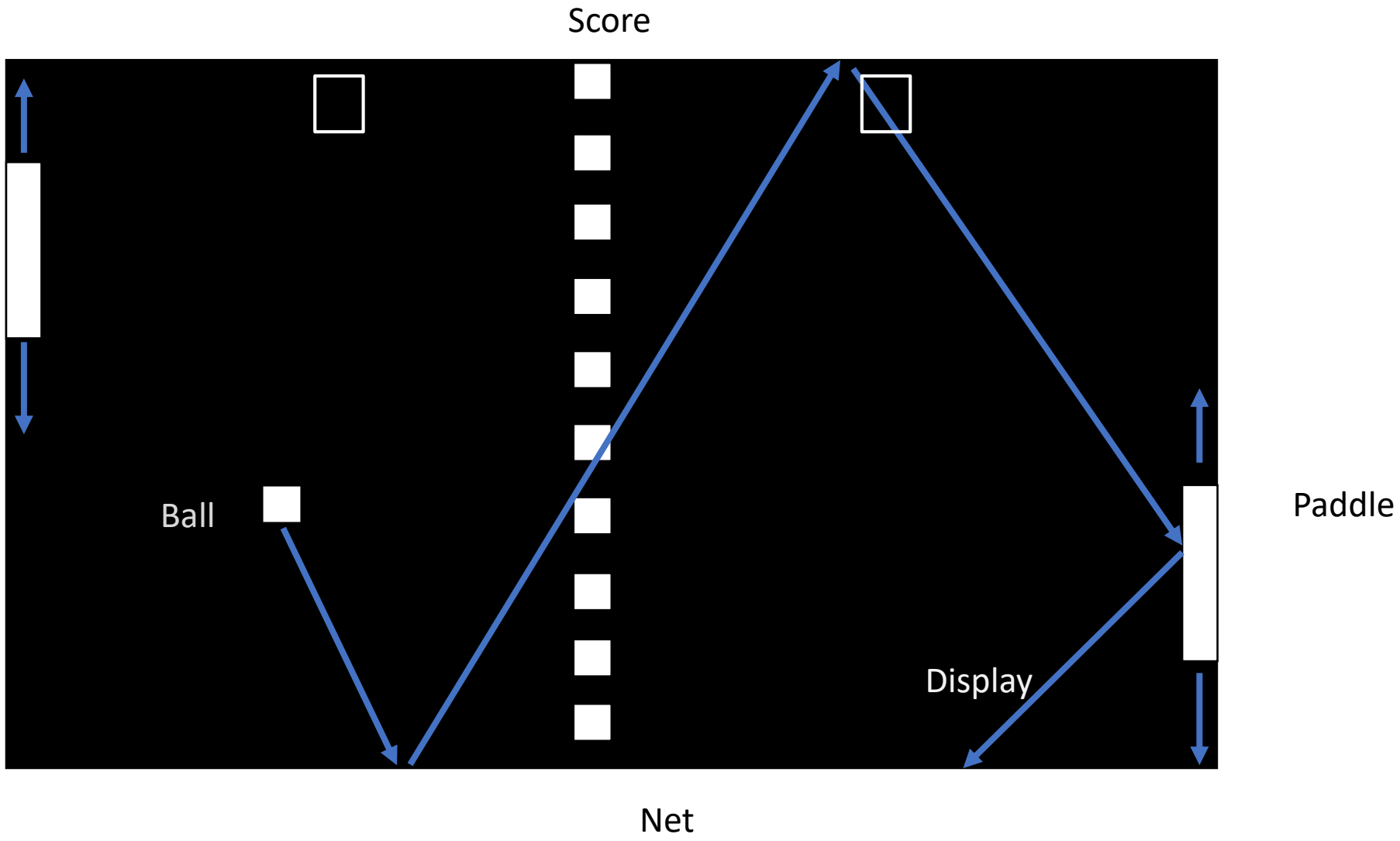
Text buffer

```
reg [7:0] mem [0:47];

integer k;

initial
begin
  for (k = 0; k < 48; k = k + 1)
    mem[k] <= 32;
  mem[0] <= 8'd65;
  mem[1] <= 8'd66;
  mem[2] <= 8'd67;
end
```

Terminal

PONG

Elements



CPU

Grom - 8

1. 8 bit CPU
2. 4 general purpose registers
3. 12 bit address bus and PC (program counter)
4. Code and data segment registers (CS,DS) (4 bit)
5. Stack pointer (SP) (12 bit)
6. 8-bit ALU with C-carry, Z-zero and S-sign flags

Instruction set (1/2)

IR			2nd	Instruction	Info
0000	dst	src		MOV dst, src	
0001	00	reg		ADD reg	r0 = r0 + reg
0001	01	reg		SUB reg	r0 = r0 - reg
0001	10	reg		ADC reg	r0 = r0 + reg + C
0001	11	reg		SBC reg	r0 = r0 - reg - C
0010	00	reg		AND reg	r0 = r0 and reg
0010	01	reg		OR reg	r0 = r0 or reg
0010	10	reg		NOT reg	r0 = not reg
0010	11	reg		XOR reg	r0 = r0 xor reg
0011	00	reg		INC reg	reg = reg + 1
0011	01	reg		DEC reg	reg = reg - 1
0011	10	reg		CMP reg	flags of r0 - reg
0011	11	reg		TST reg	flags of r0 and reg
0100	00	00		SHL	
0100	00	01		SHR	
0100	00	10		SAL	
0100	00	11		SAR	
0100	01	00		ROL	
0100	01	01		ROR	
0100	01	10		RCL	rotate with carry
0100	01	11		RCR	rotate with carry
0100	10	reg		PUSH reg	
0100	11	reg		POP reg	
0101	dst	src		LOAD dst, [src]	
0110	dst	src		STORE [dst], src	
0111	00	reg		MOV CS, reg	
0111	01	reg		MOV DS, reg	
0111	10	00		PUSH CS	
0111	10	01		PUSH DS	
0111	10	10		???	
0111	10	11		???	
0111	11	00		???	
0111	11	01		???	
0111	11	10		RET	
0111	11	11		HLT	

Instruction set (2/2)

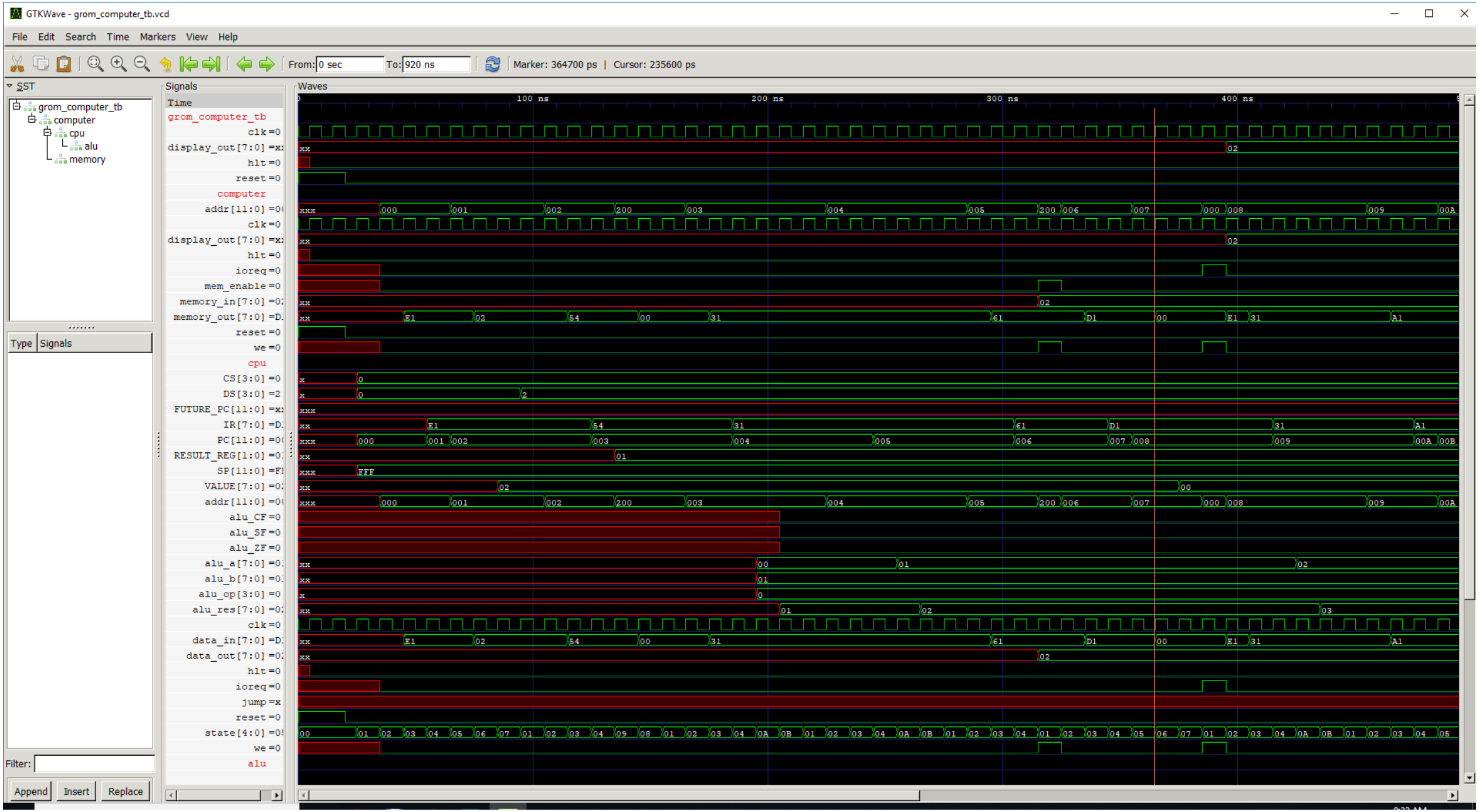
1000	00	00	val	JMP val	(unconditionally jump)
1000	00	01	val	JC val	(carry=1 jump)
1000	00	10	val	JNC val	(carry=0 jump)
1000	00	11	val	JM val	(sign=1 jump)
1000	01	00	val	JP val	(sign=0 jump)
1000	01	01	val	JZ val	(zero=1 jump)
1000	01	10	val	JNZ val	(zero=0 jump)
1000	01	11	val	???	
1000	10	00	val	JR val	(unconditionally jump)
1000	10	01	val	JRC val	(carry=1 jump)
1000	10	10	val	JRNC val	(carry=0 jump)
1000	10	11	val	JRM val	(sign=1 jump)
1000	11	00	val	JRP val	(sign=0 jump)
1000	11	01	val	JRZ val	(zero=1 jump)
1000	11	10	val	JRNZ val	(zero=0 jump)
1000	11	11	val	???	

1001	high		low	JUMP addr	
1010	high		low	CALL addr	
1011	high		low	MOV SP,addr	
1100	xx	reg	val	IN reg,[val]	
1101	xx	reg	val	OUT [val],reg	
1110	xx	00	val	MOV CS,val	
1110	xx	01	val	MOV DS,val	
1110	xx	10	val	???	
1110	xx	11	val	???	
1111	00	reg	val	MOV reg, val	
1111	01	reg	val	LOAD reg, [val]	
1111	10	reg	val	STORE [val], reg	
1111	11	xx		???	

ALU

```
ALU_OP_ADD :
    tmp = A + B;
ALU_OP_SUB :
    tmp = A - B;
ALU_OP_ADC :
    tmp = A + B + { 7'b0000000, CF };
ALU_OP_SBC :
    tmp = A - B - { 7'b0000000, CF };
ALU_OP_AND :
    tmp = {1'b0, A & B };
ALU_OP_OR :
    tmp = {1'b0, A | B };
ALU_OP_NOT :
    tmp = {1'b0, ~B };
ALU_OP_XOR :
    tmp = {1'b0, A ^ B};
```

```
ALU_OP_SHL :
    tmp = { A[7], A[6:0], 1'b0};
ALU_OP_SHR :
    tmp = { A[0], 1'b0, A[7:1]};
ALU_OP_SAL :
    // Same as SHL
    tmp = { A[7], A[6:0], 1'b0};
ALU_OP_SAR :
    tmp = { A[0], A[7], A[7:1]};
ALU_OP_ROL :
    tmp = { A[7], A[6:0], A[7]};
ALU_OP_ROR :
    tmp = { A[0], A[0], A[7:1]};
ALU_OP_RCL :
    tmp = { A[7], A[6:0], CF};
ALU_OP_RCR :
    tmp = { A[0], CF, A[7:1]};
```


GTKWave - grom_computer_tb.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 920 ns
Marker: 364700 ps | Cursor: 235600 ps
SST
grom_computer_tb
computer
cpu
alu
memory
Type Signals
Filter:
Append Insert Replace
Signals
Time
grom_computer_tb
clk =0
display_out[7:0] =x
hlt =0
reset =0
computer
addr[11:0] =0
clk =0
display_out[7:0] =x
hlt =0
ioreq =0
mem_enable =0
memory_in[7:0] =0
memory_out[7:0] =D
reset =0
we =0
cpu
CS[3:0] =0
DS[3:0] =2
FUTURE_PC[11:0] =x
IR[7:0] =D
PC[11:0] =0
RESULT_REG[1:0] =0
SP[11:0] =F
VALUE[7:0] =0
addr[11:0] =0
alu_CF =0
alu_SF =0
alu_ZF =0
alu_a[7:0] =0
alu_b[7:0] =0
alu_op[3:0] =0
alu_res[7:0] =0
clk =0
data_in[7:0] =D
data_out[7:0] =0
hlt =0
ioreq =0
jump =x
reset =0
state[4:0] =0
we =0
alu
Waves
100 ns
200 ns
300 ns
400 ns

Software running on our CPU

```
store[0] <= 8'b11100001; // MOV DS,2
store[1] <= 8'b00000010; //
store[2] <= 8'b01010100; // LOAD R1,[R0]
store[3] <= 8'b00110001; // INC R1
store[4] <= 8'b00110001; // INC R1
store[5] <= 8'b01100001; // STORE [R0],R1
store[6] <= 8'b11010001; // OUT [0],R1
store[7] <= 8'b00000000; //
store[8] <= 8'b00110001; // INC R1
store[9] <= 8'b10100001; // CALL 0x100
store[10] <= 8'b00000000; //
store[11] <= 8'b01111111; // HLT

store[256] <= 8'b11010001; // OUT [0],R1
store[257] <= 8'b00000000; //
store[258] <= 8'b01111110; // RET
```

COMPUTER

Altair

- M – Memory examine and change
- D – Dump memory
- J – Jump to user program

- J000000

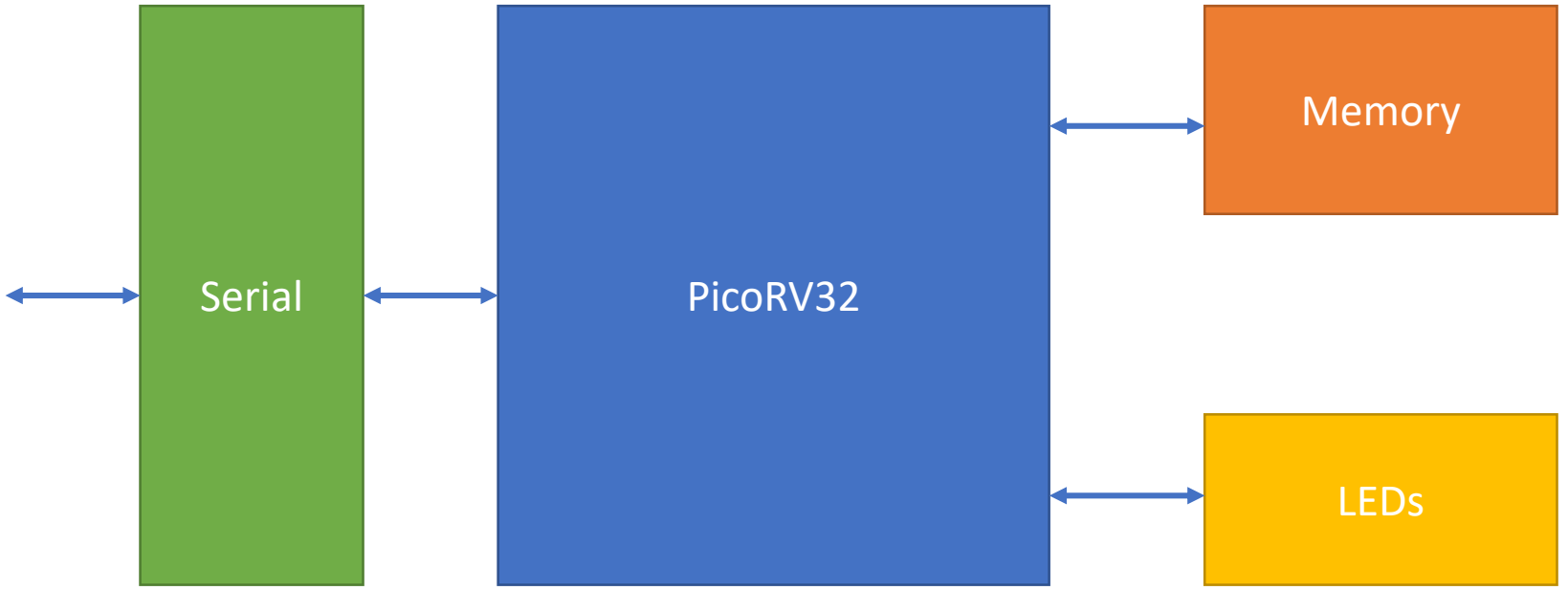
RISC-V

Risc-V ISA

- RV32I - Base Integer Instruction Set, 32-bit
- RV64I - Base Integer Instruction Set, 64-bit
- M - Standard Extension for Integer Multiplication and Division
- C - Standard Extension for Compressed Instructions

- A - Standard Extension for Atomic Instructions
- F - Standard Extension for Single-Precision Floating-Point
- D - Standard Extension for Double-Precision Floating-Point
- Q - Standard Extension for Quad-Precision Floating-Point

AttoSoC



Building

- make (to create firmware.hex)
- apio build
- apio upload

After change of files :

- make clean
- apio clean