



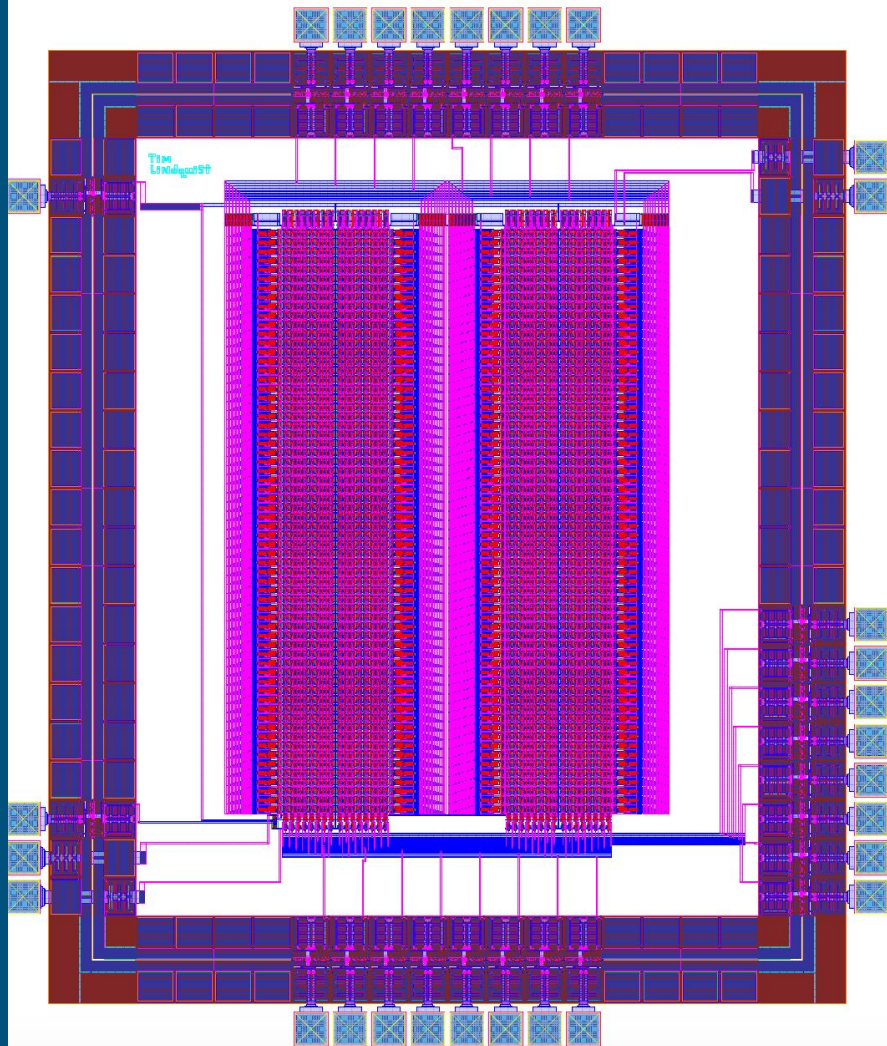
Processor + SRAM



By: Jakub Hladik, Tim Lindquist



The SRAM



SRAM

REQUIREMENTS:

- 256x8bit
 - 6T process
 - Read/Write capability
 - Data line precharging
 - 1MHz CLK
-

The 6T Cell

6T=6 transistors

Store 1 bit (0 or 1)

- Require 2048 cells

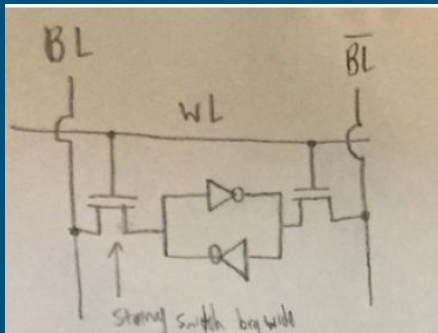
Sizing ratios

- PD transistors $8/2 \lambda$
- Access transistors $4/2 \lambda$
- PU transistors $3/3 \lambda$

PD:4x

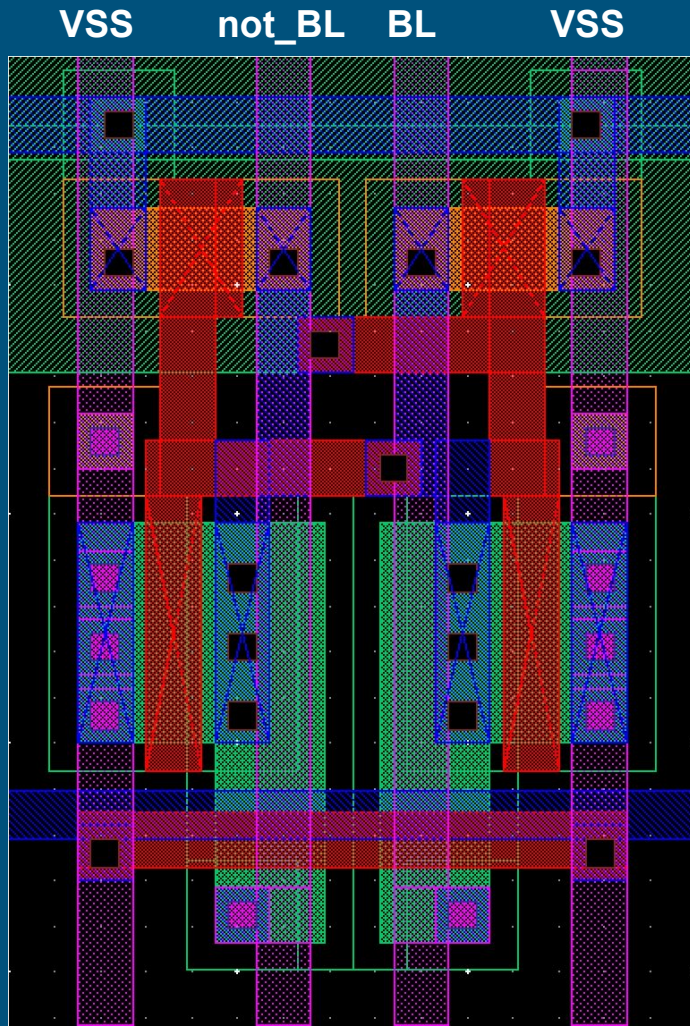
Access: 2x

PU:1x



VDD →

WL →



The Write Drivers

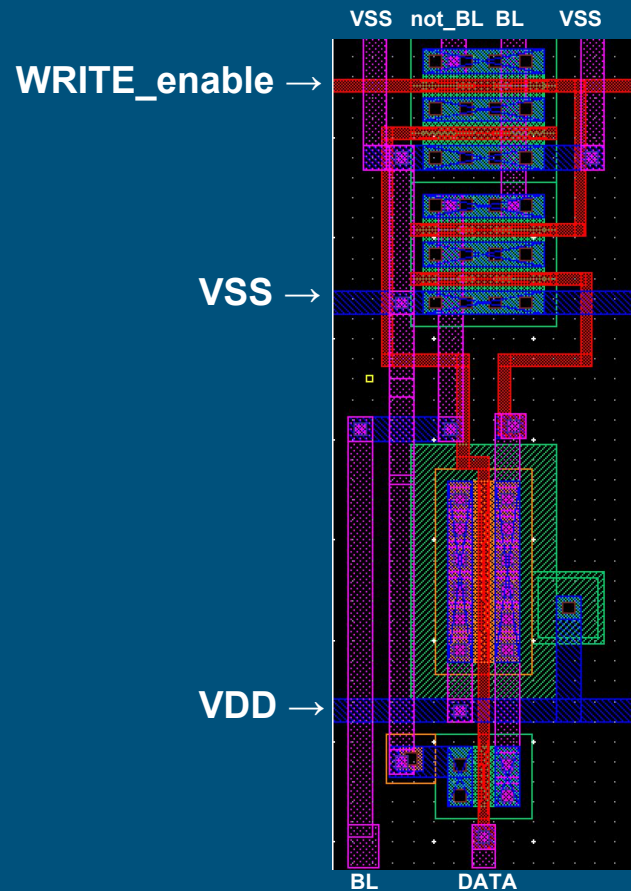
Placement:

- Under 6T cell
- 1 row x 8 column

Function

- Write data to BL's
- Read data from BL's

Strong Transistors



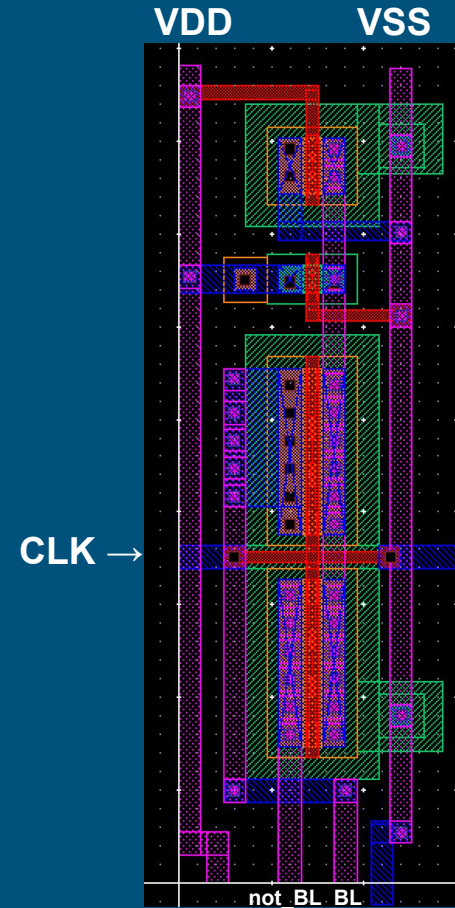
Precharge

Placement

- Above 6T cell
- 1row x 8 column

Function

- Charge line to VDD when CLK is low
- Conditions lines

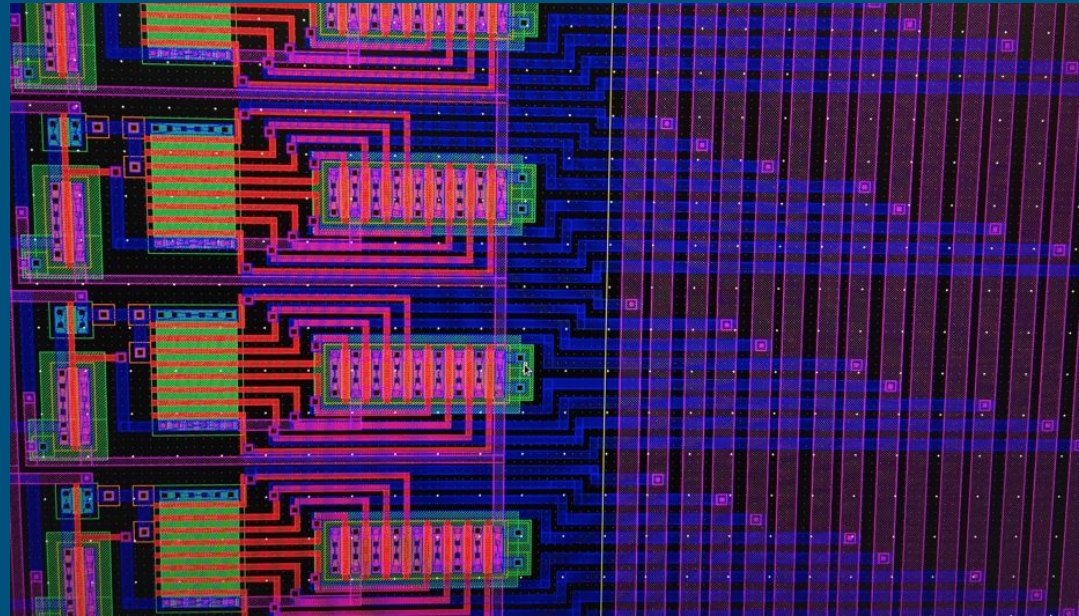


8 x 256 Decoder

8 input AND gate

Addressable 16 line config

256 different addresses



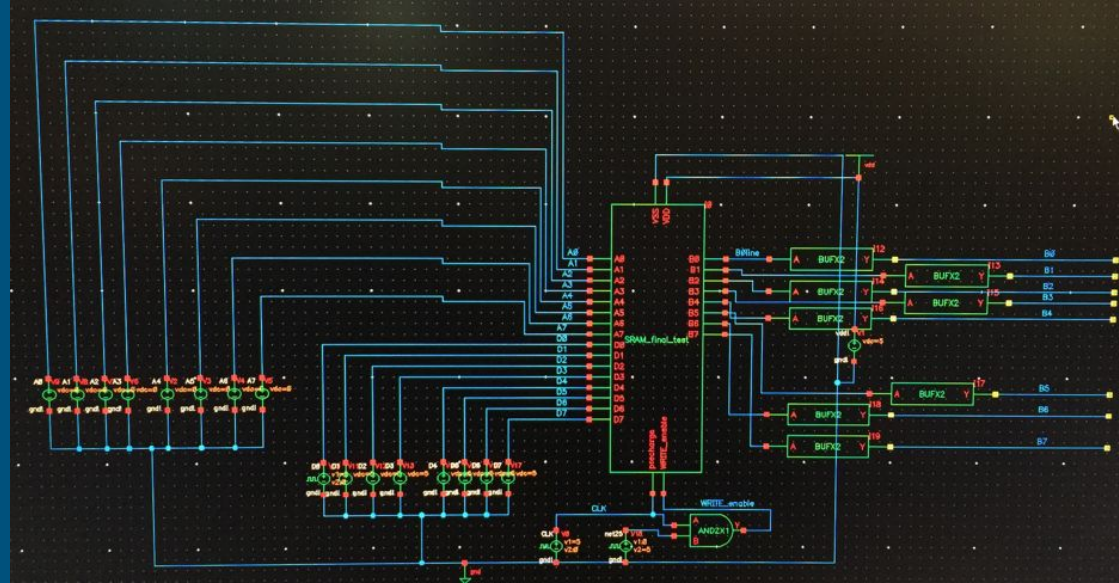
WRITE Operation

1. Write Drivers force BL and not_BL to desired state.
2. WL set to HIGH to turn on access transistors
3. Cell is written
4. WL drops LOW to save state

READ Operation

1. Precharge lines on lower CLK
2. CLK goes HIGH
3. WL set HIGH for desired address
4. Value read off BL
5. WL set to LOW

Environment= extracted parts

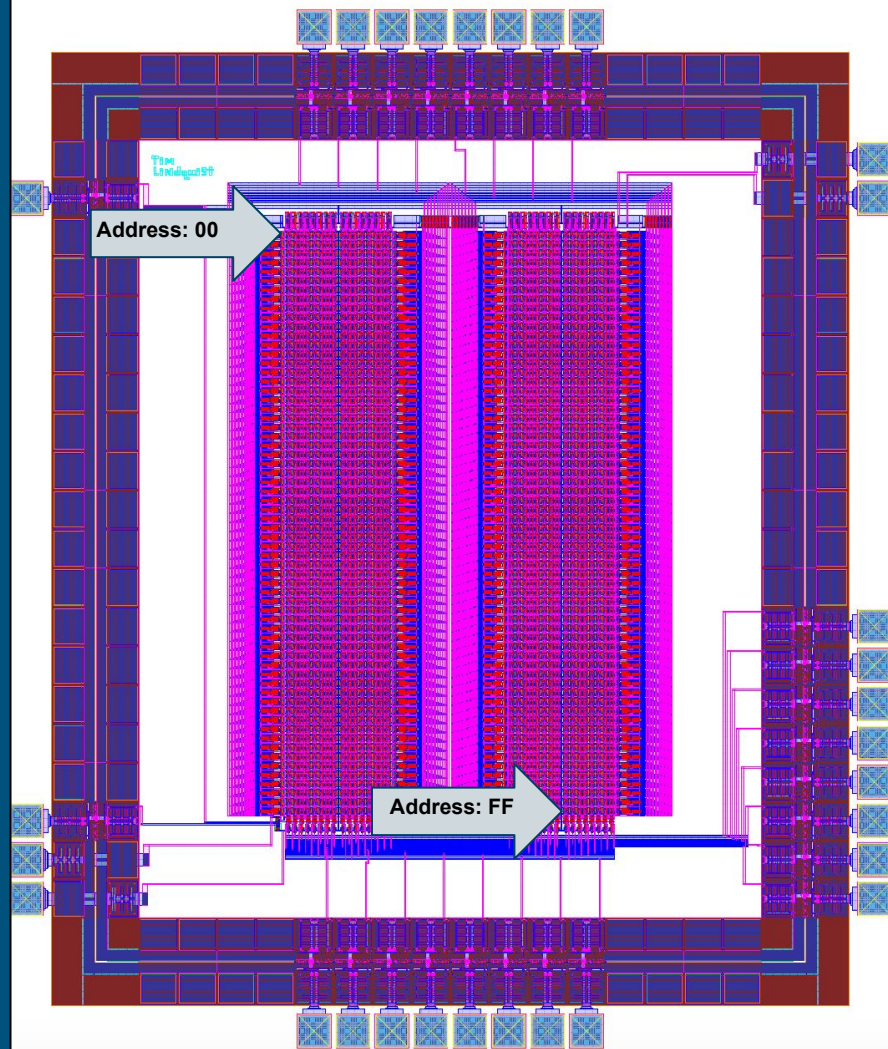


Results

Successfully read & wrote into addr 00:FF

00=0000 0000

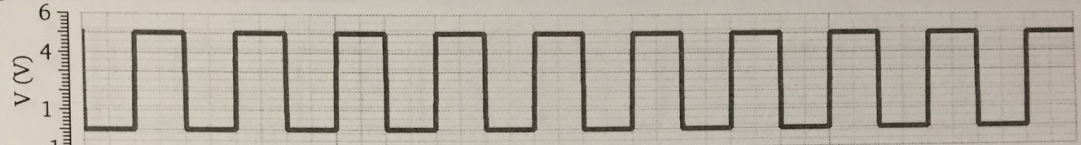
FF=1111 1111



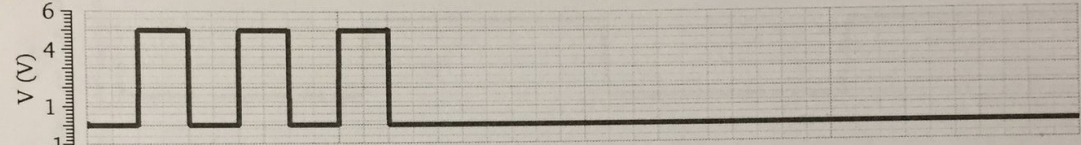
Transient Response

Name Vis

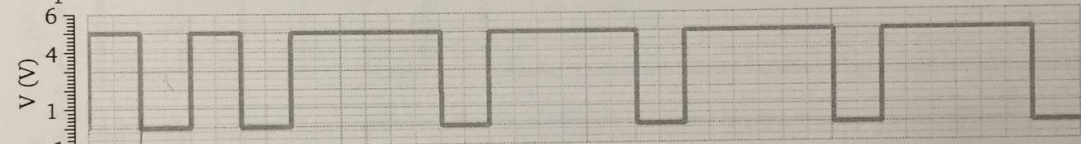
/CLK



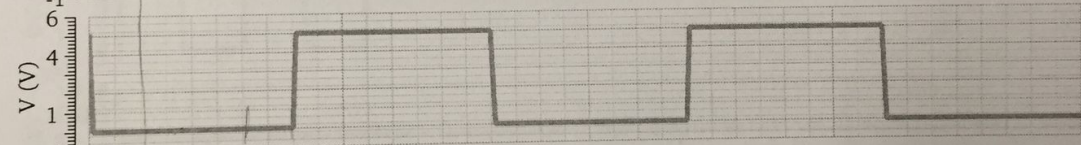
/WRITE_enable



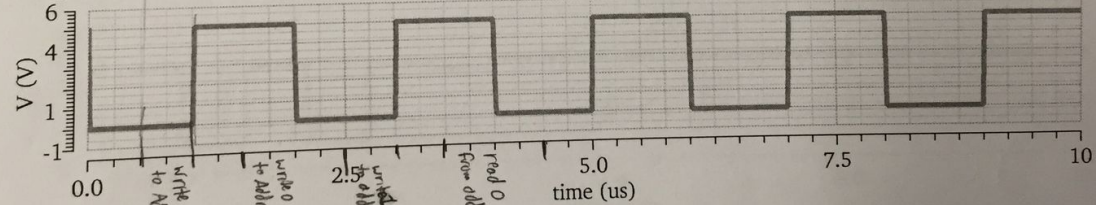
/B0



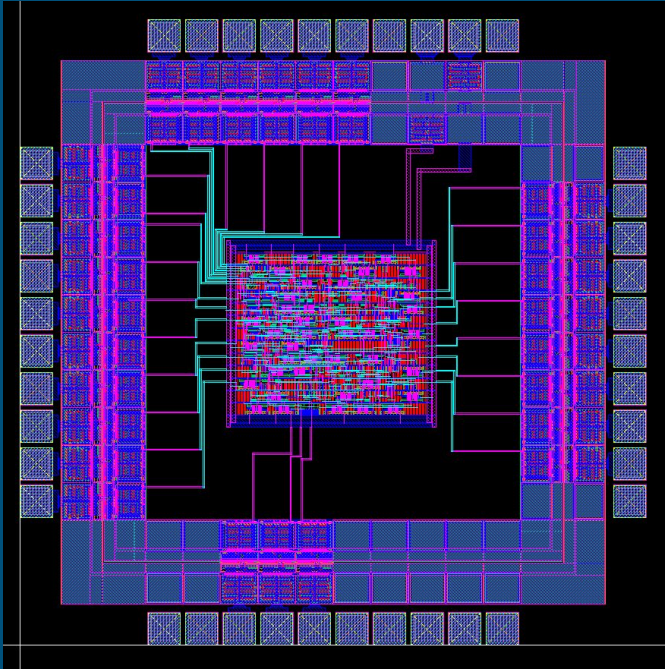
/D0



/A0



SUBLEQ: Single-Instruction Processor



SUBLEQ

2486 Transistors

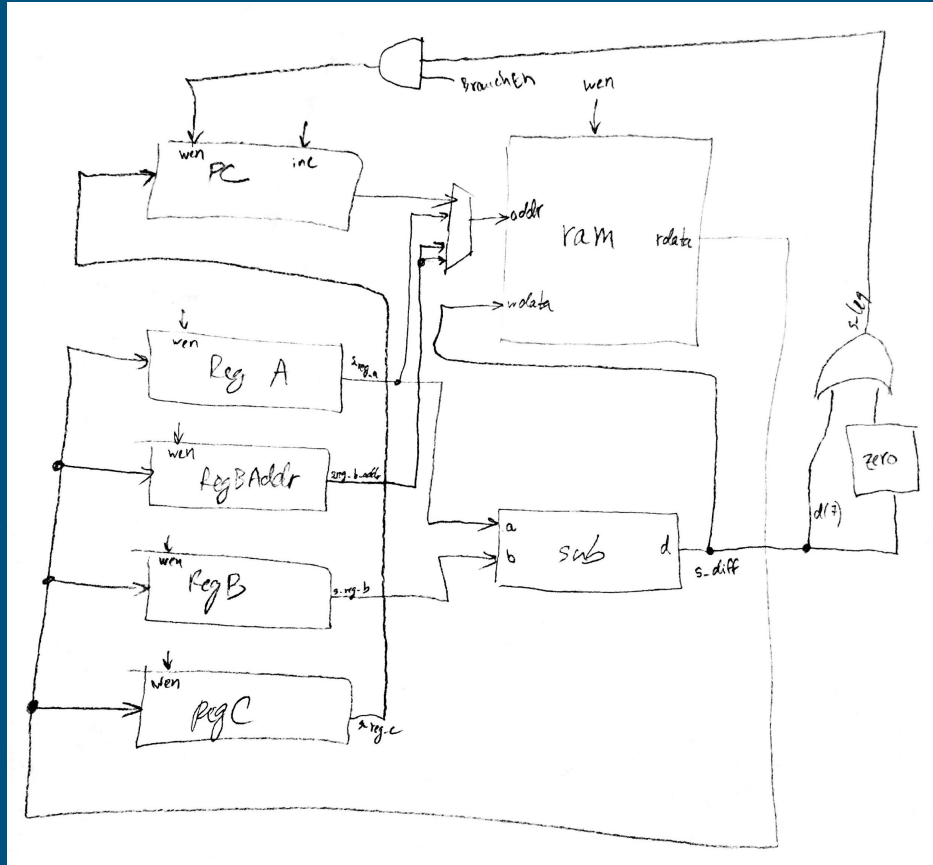
8-bit architecture

1 instruction

Low power

Extremely minimalistic computer architecture

Interesting concept



SUBLEQ

Subtract A from B and
Branch to C if result **LE**ss or
EQual to zero

Disable branch by making
value C the same as NextPC

Force Jump by subtracting
0-0

Add by subtracting a
negative number

Can solve any algorithmic
problem*

SUBLEQ: Where is the potential?

- Low component count has several effects
 - Lower propagation delays (higher clock speeds)
 - Low power
- Simplicity
 - Ideal where universal high performance not needed (ie. wrist watch)
 - Potential high performance in parallel setup

Total Power

```
-----
Total Internal Power:      1.836      67.16%
Total Switching Power:    0.8979     32.84%
Total Leakage Power:     3.78e-05    0.001382%
Total Power:              2.734
-----
```

Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage (%)
Sequential	1.382	0.124	2.292e-05	1.506	55.07
Macro	0	0	0	0	0
I/O	0	0	0	0	0
Combinational	0.339	0.4066	1.451e-05	0.7457	27.27
Clock (Combinational)	0.1154	0.3673	3.684e-07	0.4827	17.65
Clock (Sequential)	0	0	0	0	0
Total	1.836	0.8979	3.78e-05	2.734	100

Rail	Voltage	Internal Power	Switching Power	Leakage Power	Total Power	Percentage (%)
Default	5	1.836	0.8979	3.78e-05	2.734	100

Clock	Internal Power	Switching Power	Leakage Power	Total Power	Percentage (%)
i_clk	0.1154	0.3673	3.684e-07	0.4827	17.65
Total	0.1154	0.3673	3.684e-07	0.4827	17.65

```
-----
*      Power Distribution Summary:
*      Highest Average Power:          i_clk_L2_I0 (INVX8):          0.1928
*      Highest Leakage Power: control_logic_current_state_reg[0] (DFFSR): 5.33e-07
*      Total Cap:      2.0638e-11 F
*      Total instances in design: 222
*      Total instances in design with no power: 0
*      Total instances in design with no activity: 0
*
*      Total Fillers and Decap: 0
-----
```

SUBLEQ: FPGA DEMO

addr	data	PROGRAM
a 00	FB	} $i = i - 0$ if ($i \leq 0$) then JUMP
b 01	FD	
c 02	0C (exit loop)	
a 03	FE	} $a = a + 2$
b 04	FF	
c 05	06	
a 06	FC	} $i = i - 1$
b 07	FD	
c 08	09	
09	FB	} Jump to 00
0A	FB	
0B	00	
0C	FB	} exit loop infinite loop
0D	FB	
0E	0C	
0F		
10		
11		
12		

```

a = 0
for (i = 10; i > 0; i--)
{
    a = a + 2;
}
while (1) {} // infinite loop

```

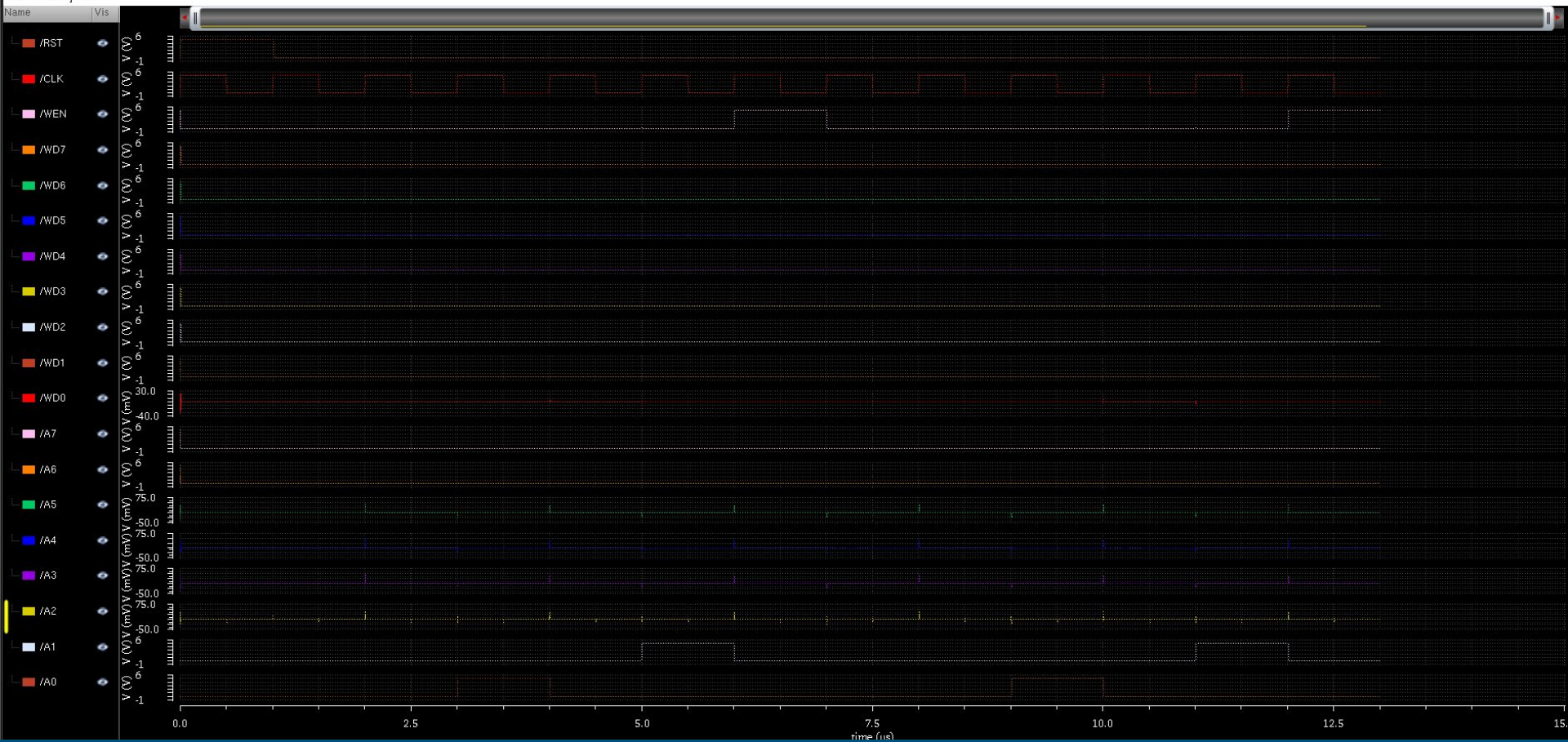
DATA

0	@	FF	(a)
-2	@	FE	const
+10	@	FD	(i)
+1	@	FC	const
0	@	FB	const

20 Bytes

final value of a
0x14
↓
20 (dec) ✓

Transient Response



Transient Response

