International Journal of Modern Developments in Engineering and Science
Volume 4, Issue 7, July 2025
https://www.ijmdes.com | ISSN (Online): 2583-3138

Adiabatic Logic and Subthreshold CMOS Circuit Design Using Opensource Software

Avijit Jutla[1], Justin Xu[2], Yinxuan Qiao[3], Daniel Abreu[4], Ziyuan Zheng[5], James T. Doyle[6*]
[1,2,3,4,5,6]*Department of Electrical Engineering, Arizona State University, Tempe, Arizona, United States*

***Abstract*: Adiabatic circuits use a varying voltage source allowing for the energy to flow back and be reused, reducing power consumption. A subthreshold circuit is a circuit which operates at below transistor threshold voltages, which is the voltage it takes for the transistor to switch on. Multiple blocks using subthreshold CMOS and adiabatic were designed and simulated in ltspice and Xscheme using the opensource Skywater 130nM PDK. These blocks include baseline subthreshold D Flipflop, 1 Mhz subthreshold integrated oscillator, sub adiabatic modulo 10 divider (order of magnitude lower power than conventional Boolean approach), adiabatic phase generator, low power relaxation oscillator. These blocks can be combined to form a lower power system.**

***Keywords*: Adiabatic, Subthreshold CMOS Logic, Opensource Integrated circuit design software.**

1. Introduction

Being successful with the open-source tools requires knowing how to use all the tools in tandem with each other. For example, most tools used in university for schematics also have a simulator. With the docker container, Xschem and NGspice are separate (however closely tied together). A similar thing can be said about the open-source layout tools that come with the container, Magic and Klayout. Although the two tools can be used separately, using them together could prove useful for speeding up workflow. Magic allows for the automatic generation of devices that are ready to be routed in a design. Klayout is closer to Cadence Virtuoso than magic and is a bit simpler for routing designs. Klayout also supports scripts made in Python and Ruby that can speed up workflow. One such example is an automatic via generator created by the open-source ASIC community.

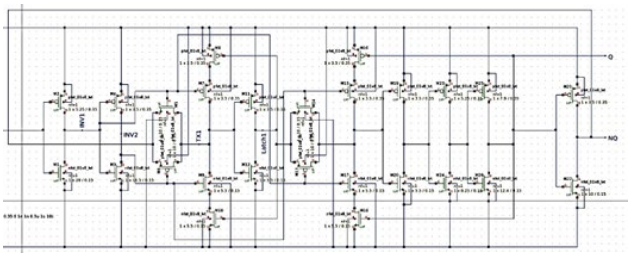

Fig. 1. Low power D Flipflop in the divide by two configurations

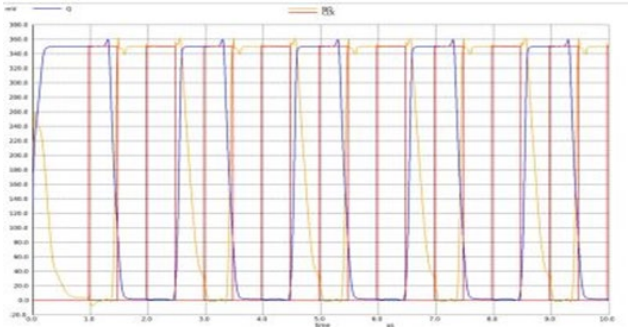

Fig. 2. Output of the low power D Flip flop in the divide by two configurations, Ngspice 130nM PDK

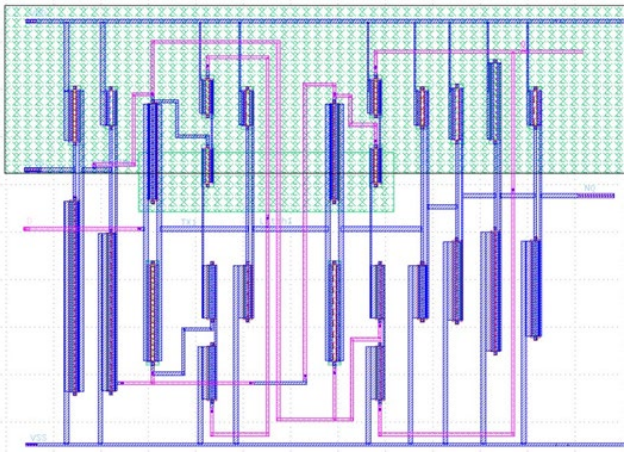

Fig. 3. Low power D Flip flop roughly ~50um wide

Devices Generated in Magic Design routed in Klayout:

Another circuit that was designed is the sub adiabatic modulo circuit. This circuit is able to do an arbitrary division based on changing resistance and capacitance value, allowing high divisions at significantly lower power than drawn by a divisor made by toggle flip-flops.

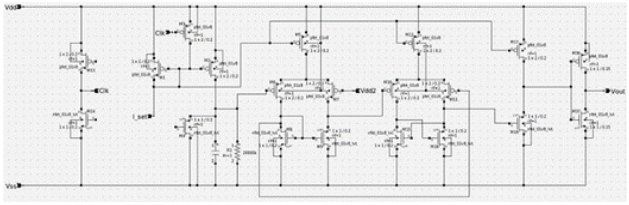

Fig. 4. Xschem schematic of sub adiabatic modulo circuit

*Corresponding author: jbkkdoyle1@gmail.com

In addition, a partially adiabatic D flip-flop was designed for low-power applications, using energy recovery techniques. It incorporates clock-controlled transmission gates and latches for state retention with minimal heat dissipation. The lower flip-flop section, positioned toward the bottom, functions as a storage element during clock transitions, aiding in reversible logic and charge recycling. The use of stacked inverters and complementary pass-transistor logic supports dynamic and energy-efficient switching behavior in thermally aware systems.

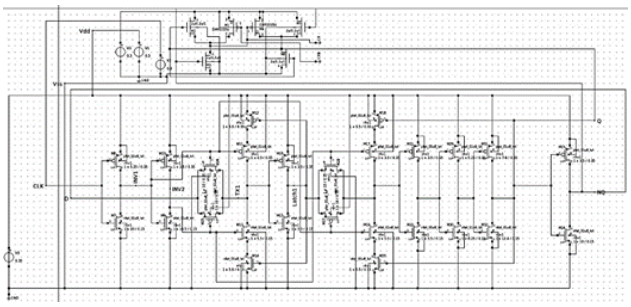

Fig. 5. Xschem schematic of partially adiabatic flipflop

2. Testing

The previous team of the ASU capstone had designed two circuits which were taped out to the Tiny Tapeout 5 and Tiny Tapeout 6 chips. The project numbers are 328 on Tiny Tapeout 5 and 423 on Tiny Tapeout 6. On Tiny Tapeout 5 the circuit is an 8 bit gray code counter. A gray code counter is a counter that switches only one bit at a time from number to number therefore decreasing the possibilities of an error. The counter on the Tiny Tapeout chip was tested for speed which showed that at 10Mhz the output waveform started to distort and at 50Mhz the clock of the input waveform no longer matched the output waveform. The Cyclic Redundancy check, on the Tiny Tapeout 6 chip, was tested for functionality. There were 3 ways the chip was verified, the first of which was using the Wokwi design done by the original team and the other two were using an Arduino to generate 128 bits of random data and then add a bit flip to simulate an error. In the functionality test it was found both the Wokwi design and Arduino simulation were able to match the design very well. Results published on Github Opensource projects (see references).

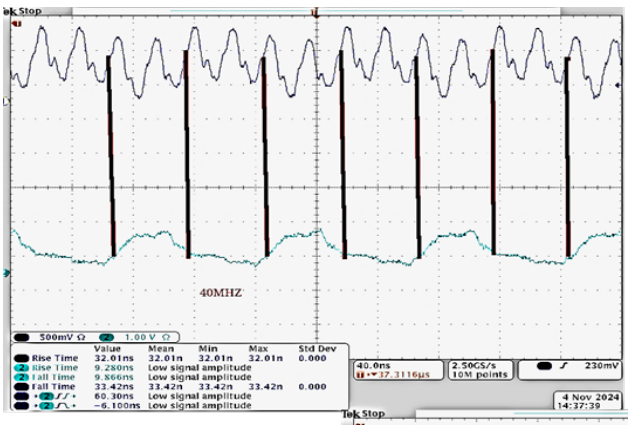

Fig. 6. Gray code 40MHz distorted waveforms

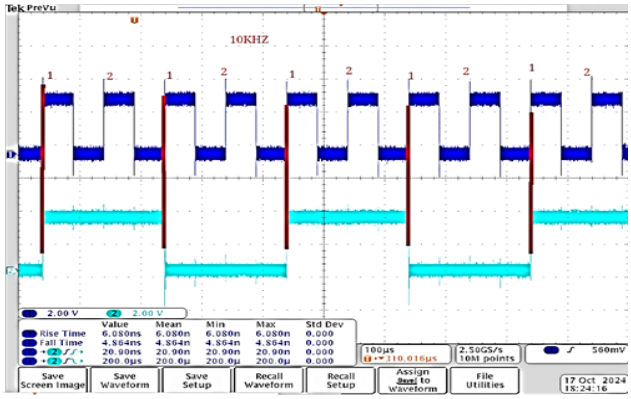

Fig. 7. Gray code 10KHz waveforms

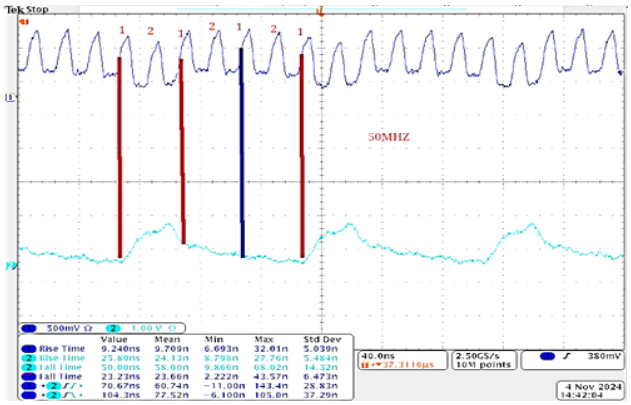

Fig. 8. Gray code 50 Mhz (Unmatched clock and output)

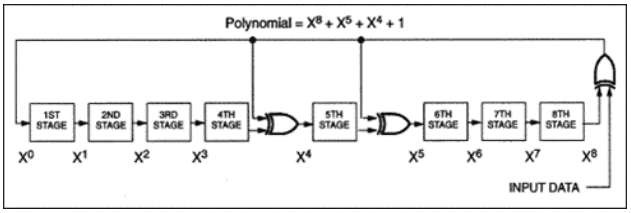

Fig. 9. 8-bit CRC design

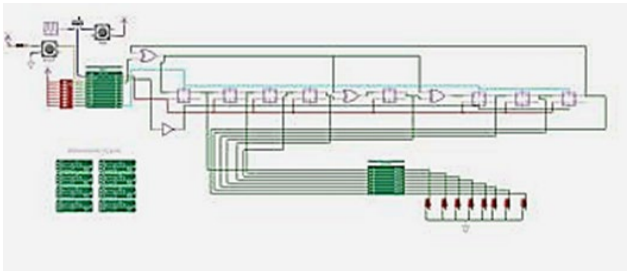

Fig. 10. 8-bit CRC Wokwi design

Fig. 11. 8-bit CRC Arduino code

3. Results

Table 1

Design	Average Power
Baseline FlipFlop	20.82/40.18 nW
1 Mhz LP Oscillator	2.9 uW
Subadiabatic Modulo 10	13.893 nW
Partially Adiabatic Generator	24.664 nW
Relaxation Osscilator	31.561 nW

4. Conclusion

This paper presented a study on adiabatic logic and subthreshold CMOS circuit design using opensource software.

References

[1] D. Maksimovic, V. G. Oklobdfija, B. Nikolic, and K. W. Current, "Clocked CMOS Adiabatic Logic with Integrated Single-Phase Power Clock Supply, Experimental Results," ACM, Inc. pp. 323-327. August 1997.

[2] C. Luo and 1. Hu, "Single-phase adiabatic flip-flops and sequential circuits using improved CAL circuits," IEEE ASICON'07, Guilin, China, pp. 126-129. 2007.

[3] H. Ni, 1. Hu,"Near-Threshold Flip-Flops Using Clocked Adiabatic Logic in Nanometer CMOS Processes,", Key Engineering Materials Journal, vols.460-461, pp 837-842, 2011.

[4] D. Bol, D. Kamel, D. Flandre, and I. Legat, "Nanometer MOSFETeffects on the minimum-energy point of 45nm subthreshold logic," Proceedings of the 14th ACM IEEE international symposium on Lowpower electronics and design, August 2009.

[5] D. Markovic, C. C. Wang" Alarcon, L.P. Alarc6n, T. Liu, 1. M. Rabaey, "Ultralow-Power Design in Near-Threshold Region," Proceedings of the IEEE vol. 98(2), pp.237-252, 2010.

[6] Minakshi Sanadhya, M. Vinoth Kumar. "Recent Development in Efficient Adiabatic Logic Circuits and Power Analysis with CMOS Logic" 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015).

[7] Ragh Kuttappa, Steven Khoa, Leo Filippini, Vasil Pano, and Baris Taskin "Comprehensive Low Power Adiabatic Circuit Design with Resonant Power Clocking" Drexel University, Philadelphia, PA, USA 2020.

[8] Generic Structures: First-Order Positive Feedback Produced for the System Dynamics in Education Project MIT System Dynamics Group.

[9] Chi-Chia Sun and Cheng-Chih Wang. "Ultra-low power circuit design based on Adiabatic Logic". 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing.

[10] https://github.com/ariz207/tt05_GrayCounter

[11] https://github.com/ariz207/tt06-CRC8bit

Addendum: Python Power calculation for LTspice

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Load the data file
file_path"/users/asj2021/Downloads/Capstone_LTspice/Normal_FlipFlo
ps/lowpowerdflipflop2.txt"
df pd.read_csv(file_path, delim_whitespace=True)
# Extract the current and time to arrays
current = df['I(V1)'].to_numpy()
time = df.iloc[:, 0].to_numpy()
#Calculate integral of current
current_integral = np.trapz(current, time)
#Find average current and print it
avg_current = current_integral / (10**-5)
print(f"Average current (charge/seconds) {avg_current}")
print(f"In nanoamps = {avg_current * (10**9)} nA")
print("")
######## Below here is commented out average power calculation
##Calculate power
df['Power'] = df['V(n001)'] * df['I(V1)']
power_values = df['Power'].to_numpy()
##Compute time-weighted average power
##Take integral of power and then divide by time (10us)
average_power = np.trapz(power_values, time) / (10**-5)
##Convert power to nano watts and print
average_power_nanowatts = average_power * (10**9)
print(f"Time-weighted Average Power in nanowatts:
{average_power_nanowatts}")
```