VLSI Design

Project Report

Chetanya Goyal - 2021112009

The directory structure of the submitted file is of the form-

- project
  - and
    - delays_and
    - dumps_and
    - modded_and
    - power_and
  - full_adder
    - delays_full
    - dumps_full
    - modded_full
    - power_full
  - half_adder
    - delays_half

- dumps_half
- modded_half
- power_half

multiplier

- delays_mult
- dumps_mult
- modded_mult
- power_mult

multiplier_layout

- delays_mult_layout
- dump_mult_layout
- modded_mult_layout
- power_mult_layout

codes

- scripts
- c++ files

layouts and .spice

verilog

Additional files used -

```
- TSMC_180nm.txt
- 22nm_MGK.pm
- SCNM_DEEP.09.tech27
```

- All the script files ($.sh$) were run on the bash terminal and were used to compile and run the ngspice dump and modded files created
  - The dump files contain the terminal outputs for all the simulations
  - The modded files are a copy of the original $.cir$ file, except with $.probe$ statements and a control block with only the transient command. These were used to run separate simulations for power analysis
  - The script files run the modded files in bash in a loop and suppress the output to the specified dump files, from which the final power outputs are extracted
  - layouts and spice files have been added to a separate folder so that the cells can be revealed in the multiplier by selecting the required cell and pressing X. If they were in that circuit's folder, the other subcells would have not been revealed
- All the codes were included. To run them from scratch the process was -

  ```
  // move the required script and the cpp files to the circuit to be run's folder
  // they have been kept in a separate folder to avoid clutter
  $ g++ final_read_(circuit).cpp
  $ ./a.out
  $ chmod +x script_(circuit).sh
  $ ./script_(circuit).sh
  $ g++ final_write_(circuit).cpp
  $ ./a.out
  ```

  Where $(\mathrm{circuit})$ signifies what circuit the code is being run for, i.e. multiplier, and gate, etc.

  This will cpp file generates the modified circuits with the $.probe()$ statement added and the second file generates the dumps and the power extracted
- The worst case delays for the AND gate from the ngspice simulation was approximately $2.02814 \times 10^{-10} s$ and the delay for the layout was $3.03867 \times$

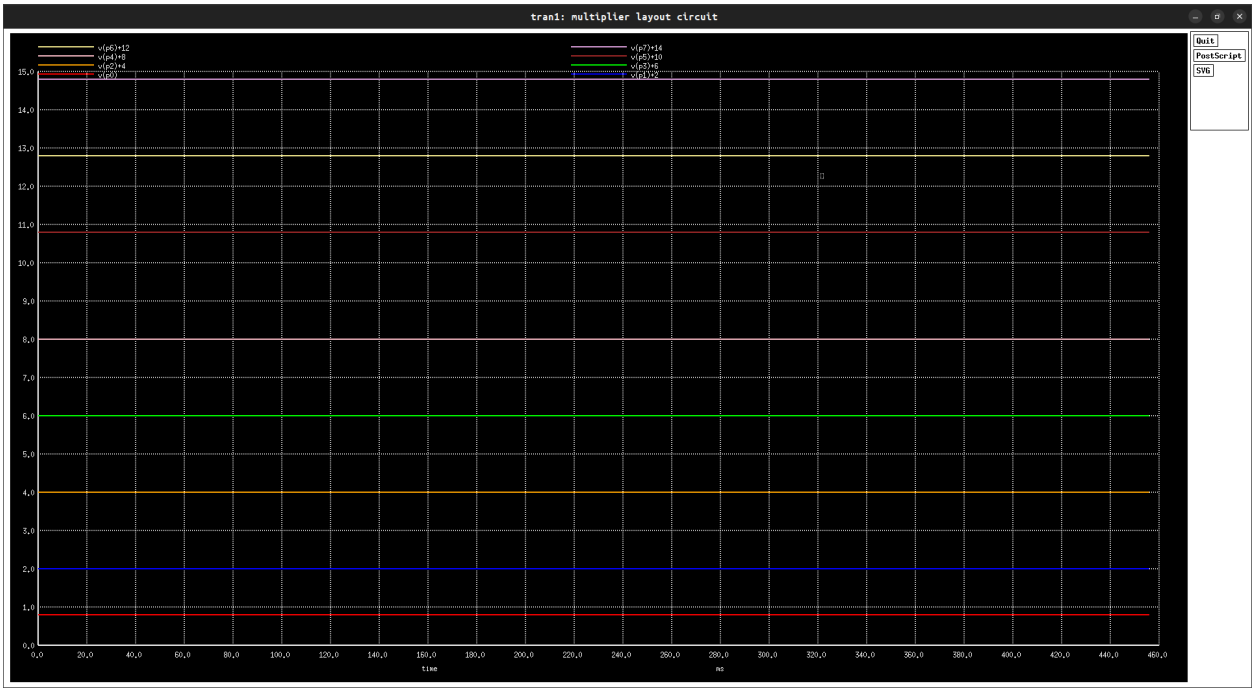
$10^{-9}s$, which is approximately $2.9ns$ worse. This was due to the higher value of intrinsics

- The power for the AND gate for layout was almost 200 times as large as the power for the ngspice version. It is know that power is dependent on the leakage current, which was seen to be higher when the dump files were compared for the two types of circuits.
- The worst case delays for the half adder were around $2.4ns$ worse due to the increased number of parasitics present in the circuit and inefficiency in arrangement of the transistors, leading to longer channels.
- The power for the half adder was found to be much better, however due to use of CMOS technology in the xor gate. CMOS circuits allow an almost neglible amount of leakage current, due to which the power was much better.
- The delays for the full adder layout were worse by approximately $5ns$ for the same reasons as the half adder
- The static power consumption was better for the layout on average by approximately 1000 times for some inputs combinations, but for the rest, the power of the ngspice full adder was better by 10^{10} orders of magnitude due to the CMOS nature of the ngspice circuit versus the cascaded half adder nature of the magic layout.
- The worst case delays were included in separate files for each circuit and it was found that the delays for the layout created using magic were worse by approximately $42ns$. This is assumed to be due to the extra capacitances added by the messy distributions of $poly$ and $m1$, used to connect nodes. The transistors in the layout were not sized such that the low-to-high and high-to-low delays would be equal, but they were designed for minimum delay.
- The power was also much worse in the layout, as compared to the ngspice multiplier. $32.198083 \times 10^{-8}W$ approximately was the power obtained for the top rightmost AND gate in the layout (and_0) and $6.68423 \times 10^{-8}W$ was the power obtained for the $X1$ subcircuit in the ngspice multiplier layout (the and gate corresponding to and_0 from the layout). It can be extrapolated that the power for the larger subcircuits (half adder and full adder and multiplier in general) would increase manifold. For example, the power in $X10$, a full adder, was found to be $3.6403 \times 10^{-7}W$ and 16.508421×10^{-7} for the layout version of the multiplier.

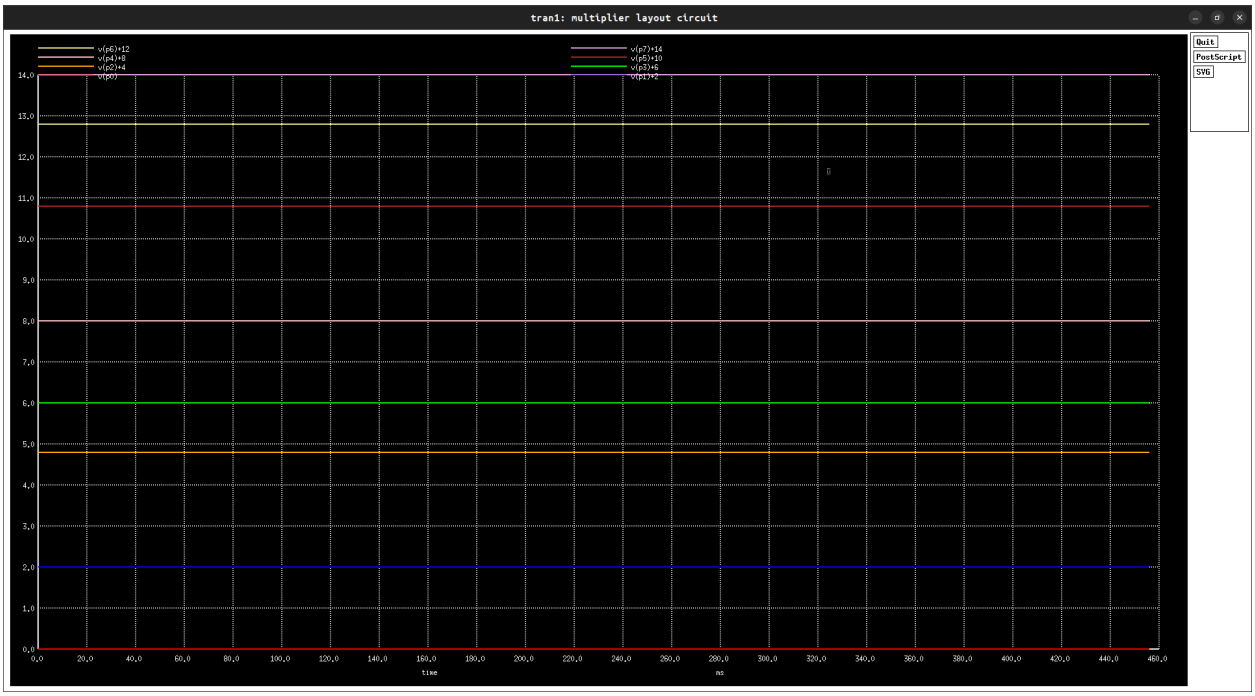
The main reason for this is that -

- The ngspice version uses the CMOS versions of all subcircuits, whereas the layout in magic uses a cascaded half-adder with an or gate in making a full adder, which leads to much higher static power consumption
- This leads to a lower leakage current in the ngspice circuit, which in turn, decreases the static power consumption of the circuit

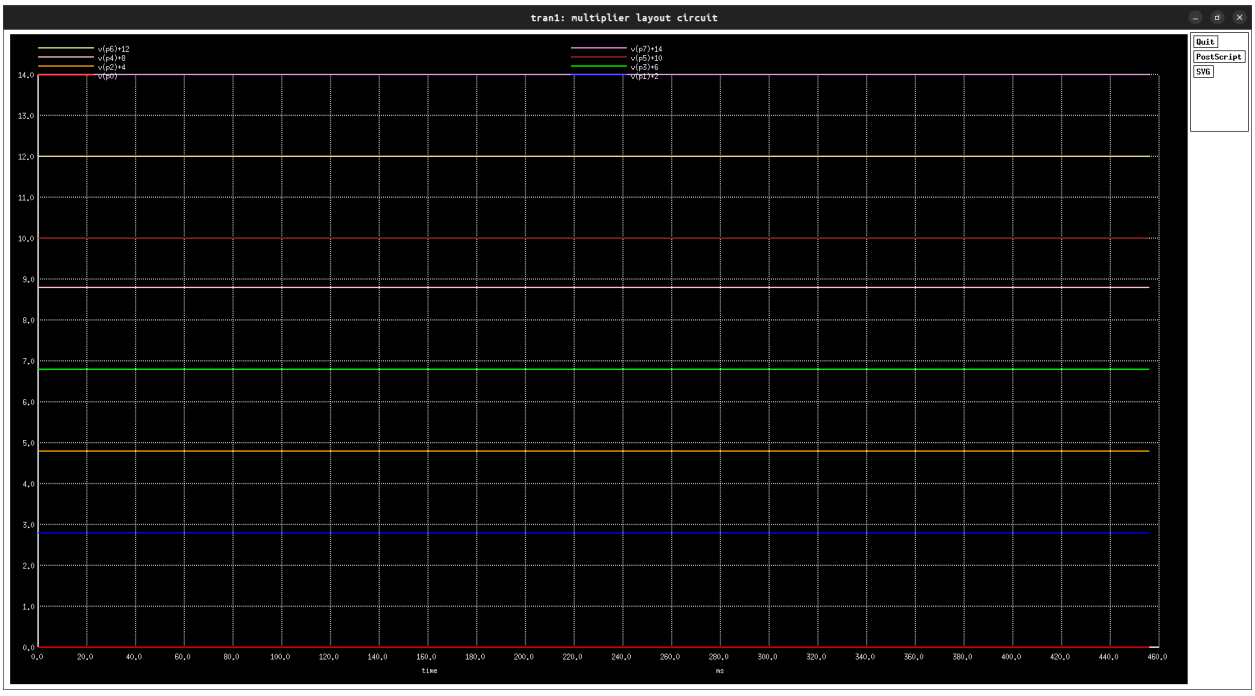
- One difference was the decrease in the number of spikes seen in the transient spice simulations. The output plot for the layout's spice simulation was cleaner and contained less spikes as compared to the ngspice multiplier.



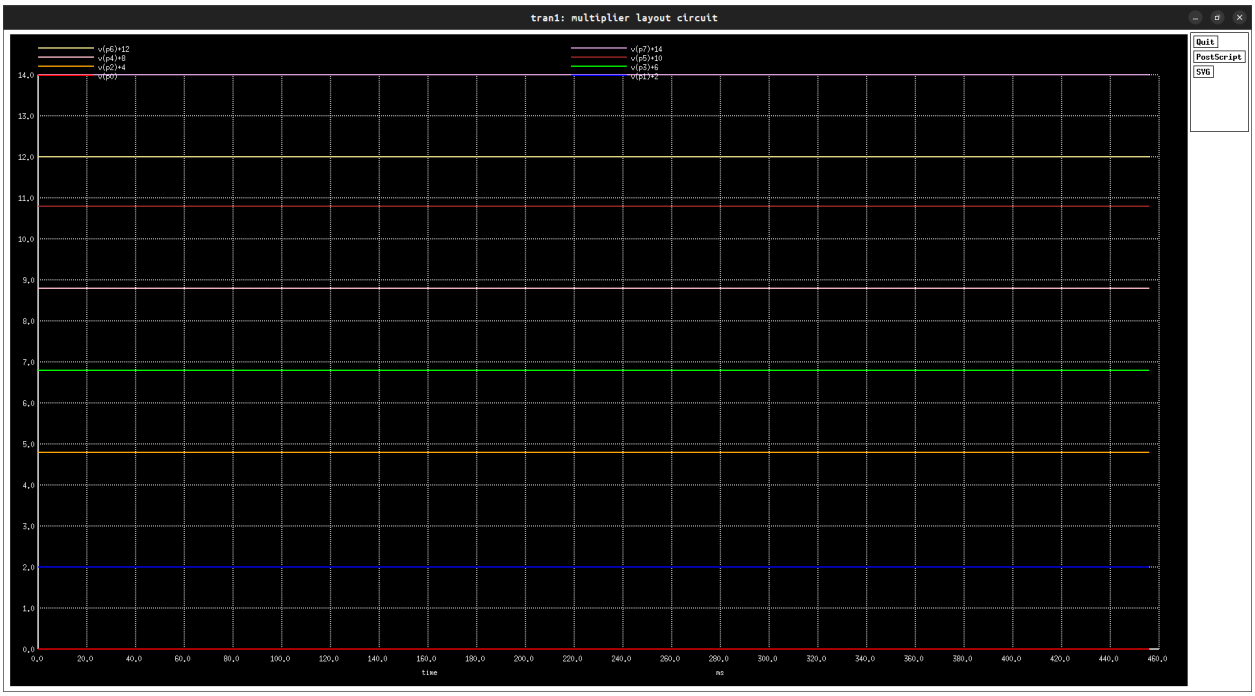
Multiplier layout output for $B = \{1111\}$ and $A = \{1111\}$



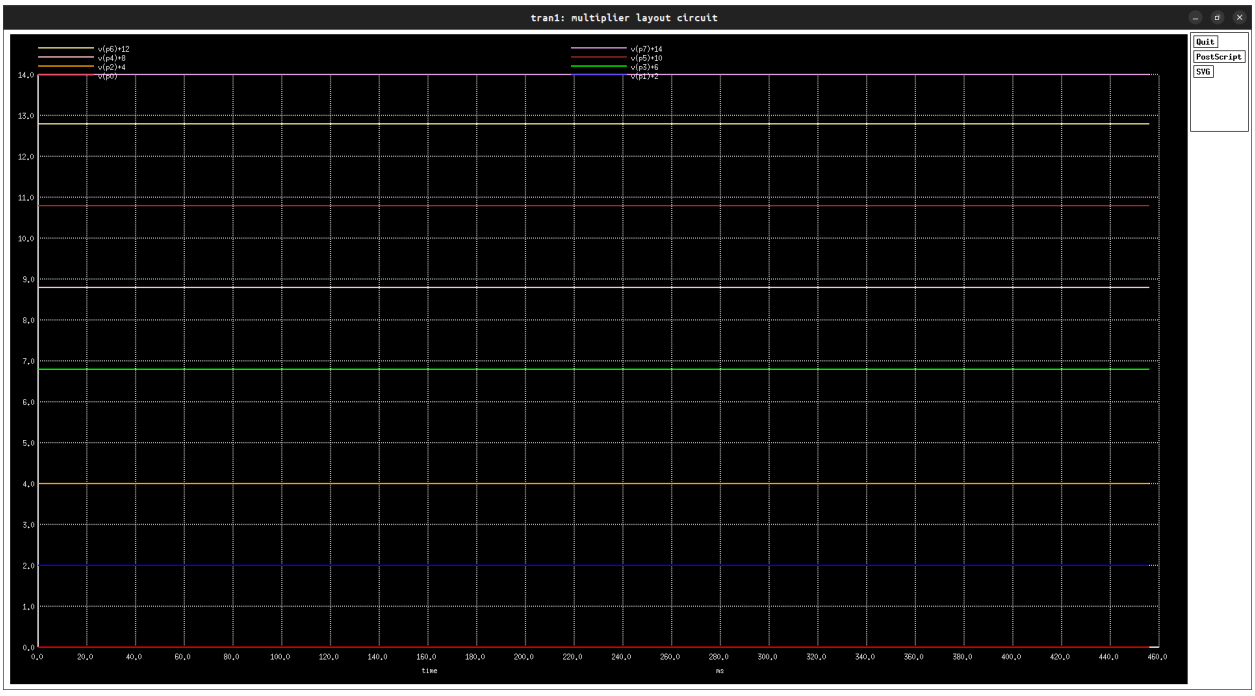
Multiplier layout output for $B = \{1010\}$ and $A = \{1010\}$



Multiplier layout output for $B = \{1111\}$ and $A = \{0010\}$



Multiplier layout output for $B = \{1111\}$ and $A = \{0100\}$



Multiplier layout output for $B = \{1111\}$ and $A = \{1000\}$