

# LAHAF: Low-power, area-efficient, and high-performance approximate full adder based on static CMOS Implementation Using eSim

**Name of the Participant:** KARTHIKEYAN S

**Institute:** Loyola ICAM College of Engineering and Technology (LICET), Chennai.

**Title of the circuit:** LAHAF: Low-power, area-efficient, and high-performance approximate full adder based on static CMOS

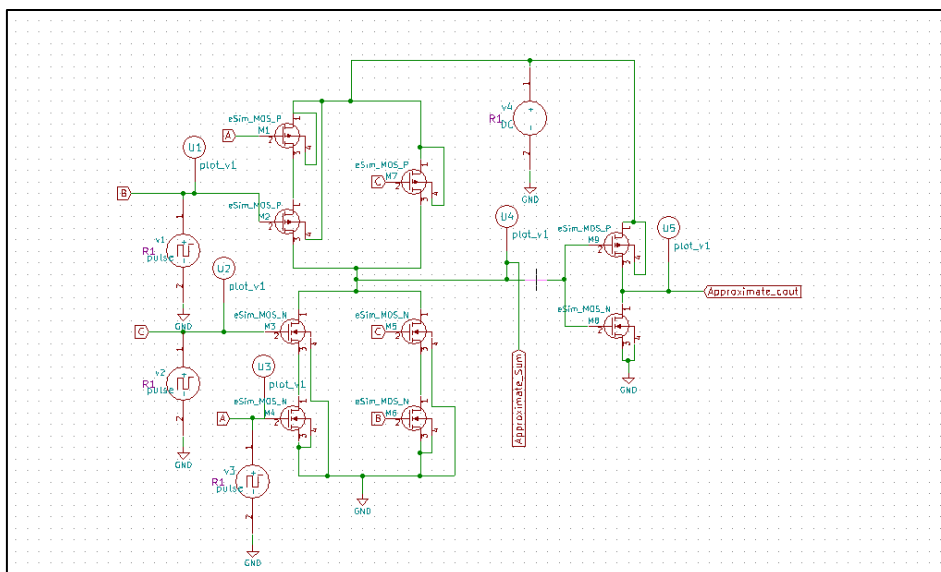
## Description:

In this article, a low-power, area-efficient full adder cell is designed, and its approximate output is presented. Techniques such as Static CMOS, Pass Transistor Logic, Transmission Gate Logic, and similar structures are employed to achieve a low-power design approach. In this work, the Static CMOS structure is used to design an approximate low-power full adder, where the Sum output is inverted to generate the Carry.

Approximate adders are suitable for complex applications such as image processing, machine learning, and data mining. Due to the human visual system's tolerance to minor inaccuracies, approximate computing can effectively be applied in processing images and videos. These adders offer significantly low power dissipation, reduced area, and acceptable performance, making them ideal for power-constrained systems.

To achieve low power consumption, adopting an area-efficient design is a key strategy, as it also contributes to reducing overall fabrication cost. The proposed design is simulated using the eSim tool, and the overall propagation delay, average power consumption, and output waveforms are analyzed. The simulation results using Ngspice illustrate the input signals (A, B, and C) and the corresponding outputs of the approximate low-power full adder (Sum and Carry).

## Circuit Diagram:



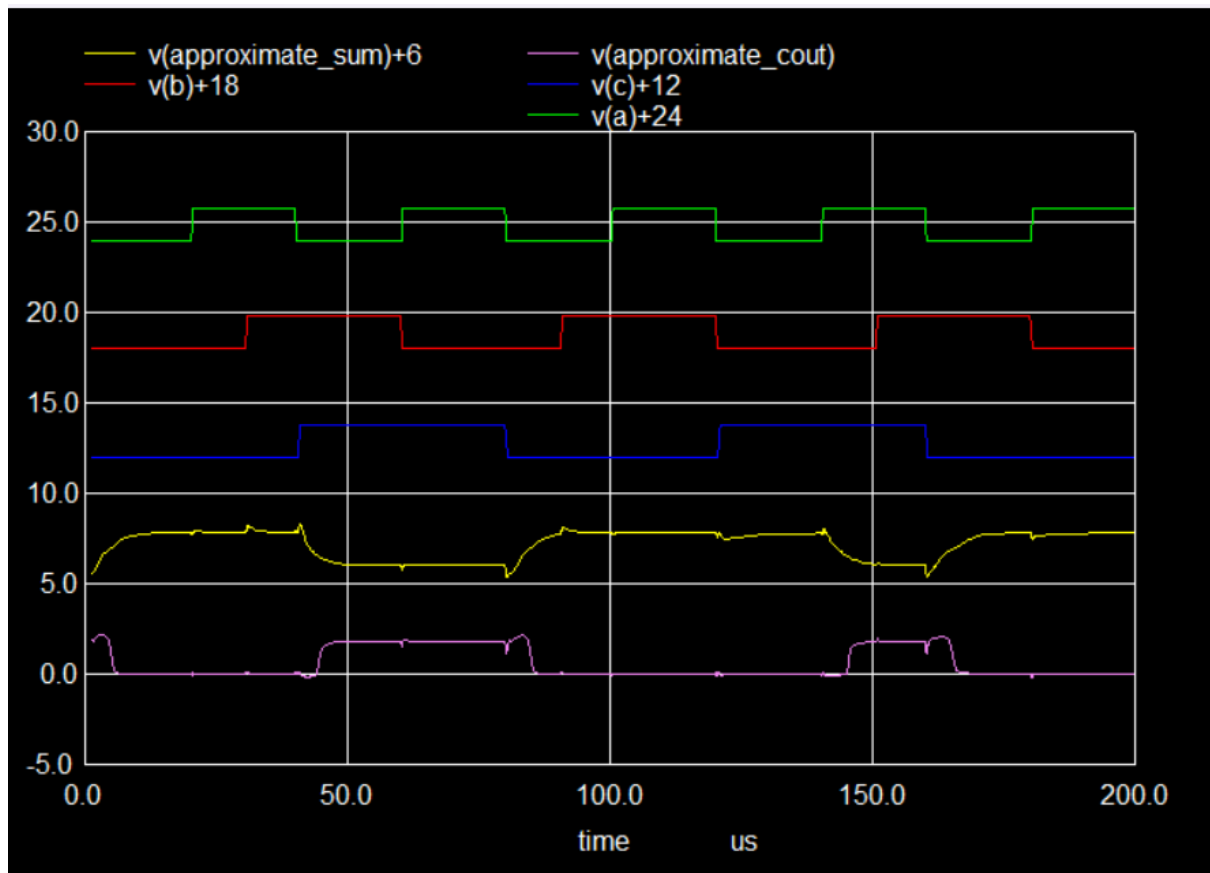
## Truth Table:

Table 1: truth table is adapted from Table 6 of the LAHAF journal [1].

The truth table of the exact full adder and the proposed approximate full adder.						
Ain	Bin	Cin	Exact Sum	Exact Cout	Approximate Sum	Approximate Cout
0	0	0	0	0	1 ×	0 ✓
0	0	1	1	0	1 ✓	0 ✓
0	1	0	1	0	1 ✓	0 ✓
0	1	1	0	1	0 ✓	1 ✓
1	0	0	1	0	1 ✓	0 ✓
1	0	1	0	1	0 ✓	1 ✓
1	1	0	0	1	1 ×	0 ×
1	1	1	1	1	0 ×	1 ✓

## Results:

**NGSPICE Outputs:** Inputs a,b,c and Outputs approximate sum and approximate carry



## Propagation Delay Using NGSPICE:

```
ngspice 35
ngspice 5 -> meas tran tpLH_B_Sum trig v(b) val=0.9 fall=2 targ v(sua) val=0.9 rise=2
tpLH_B_Sum = -3.532897e-05 targ= 8.492103e-05 trig= 1.202500e-04
ngspice 6 -> let tp_B_Sum = (tpHL_B_Sum+tpLH_B_Sum)/2
ngspice 7 -> meas tran tpHL_C_Sum trig v(c) val=0.9 rise=2 targ v(sua) val=0.9 fall=2
tpHL_C_Sum = 2.270371e-05 targ= 1.434537e-04 trig= 1.207500e-04
ngspice 8 -> meas tran tpLH_C_Sum trig v(c) val=0.9 fall=2 targ v(sua) val=0.9 rise=2
tpLH_C_Sum = -7.532897e-05 targ= 8.492103e-05 trig= 1.602500e-04
ngspice 9 -> let tp_C_Sum = (tpHL_C_Sum+tpLH_C_Sum)/2
ngspice 10 -> meas tran tpHL_A_Cout trig v(a) val=0.9 rise=2 targ v(cout) val=0.9 fall=2
tpHL_A_Cout = 2.487064e-05 targ= 8.524554e-05 trig= 6.037500e-05
ngspice 11 -> meas tran tpLH_A_Cout trig v(a) val=0.9 fall=2 targ v(cout) val=0.9 rise=2
tpLH_A_Cout = 6.490479e-05 targ= 1.450298e-04 trig= 8.012500e-05
ngspice 12 -> let tp_A_Cout = (tpHL_A_Cout+tpLH_A_Cout)/2
ngspice 13 -> meas tran tpHL_B_Cout trig v(b) val=0.9 rise=2 targ v(cout) val=0.9 fall=2
tpHL_B_Cout = -4.594463e-06 targ= 8.524554e-05 trig= 9.075000e-05
ngspice 14 -> meas tran tpLH_B_Cout trig v(b) val=0.9 fall=2 targ v(cout) val=0.9 rise=2
tpLH_B_Cout = 2.477979e-05 targ= 1.450298e-04 trig= 1.202500e-04
ngspice 15 -> let tp_B_Cout = (tpHL_B_Cout+tpLH_B_Cout)/2
ngspice 16 -> meas tran tpHL_C_Cout trig v(c) val=0.9 rise=2 targ v(cout) val=0.9 fall=2
tpHL_C_Cout = -3.550446e-05 targ= 8.524554e-05 trig= 1.207500e-04
ngspice 17 -> meas tran tpLH_C_Cout trig v(c) val=0.9 fall=2 targ v(cout) val=0.9 rise=2
tpLH_C_Cout = -1.522021e-05 targ= 1.450298e-04 trig= 1.602500e-04
ngspice 18 -> let tp_C_Cout = (tpHL_C_Cout+tpLH_C_Cout)/2
ngspice 19 ->
Interrupted once
Warning: clearing control structures
ngspice 19 -> let tp_Sum_max = max(tp_A_Sum, tp_B_Sum, tp_C_Sum)
Warning: the user-defined function max has 2 args
Error: no function as max with that arity
Error: RHS "max(tp_A_Sum, tp_B_Sum, tp_C_Sum)" invalid
ngspice 20 ->
Interrupted once
Warning: clearing control structures
ngspice 20 -> print tp_A_Sum
tp_a_sum = 4.393737e-05
ngspice 21 -> print tp_b_sum
tp_b_sum = 8.687370e-06
ngspice 22 -> print tp_c_sum
tp_c_sum = -2.63126e-05
ngspice 23 -> print tp_a_cout
tp_a_cout = 4.488767e-05
ngspice 24 -> print tp_b_cout
tp_b_cout = 9.637664e-06
ngspice 25 -> print tp_c_cout
tp_c_cout = -2.53623e-05
ngspice 26 -> ngspice 19 -> let tp_Sum_max1 = max(tp_A_Sum, tp_B_Sum)
ngspice: no such command available in ngspice
ngspice 27 -> print tp_A_Sum tp_B_Sum tp_C_Sum
tp_a_sum = 4.393737e-05
tp_b_sum = 8.687370e-06
tp_c_sum = -2.63126e-05
ngspice 28 -> print tp_A_Cout tp_B_Cout tp_C_Cout
tp_a_cout = 4.488767e-05
tp_b_cout = 9.637664e-06
tp_c_cout = -2.53623e-05
ngspice 29 ->
```

### Overall Propagation Delay:

#### Maximum Delays:

- tp\_Sum\_max = max(4.393737e-05, 8.687370e-06, -2.63126e-05)

tp\_Sum\_max = 4.393737e-05

- tp\_Cout\_max = max(4.488767e-05, 9.637664e-06, -2.53623e-05)

#### Overall Propagation Delay:

- tp\_overall = max(tp\_Sum\_max, tp\_Cout\_max)  
= max(4.393737e-05, 4.488767e-05)

Overall Delay = 4.488767e-05 seconds

## Average Power using NGSPICE:

```
ngspice 35

No compatibility mode selected!

Circuit: * d:\esimprojects\approxadder\approxadder.cir
Warning: Model issue on line 1 :
model cmosp pmos (level=8 version=3.2 tnom=27 tox=4.1e-9 xj=1e-7 nch=4. ...
unrecognized parameter (+) - ignored
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000

Checking parameters for BSIM 3.2 model cmosp
Warning: Pd = 0 is less than W.
Warning: Ps = 0 is less than W.

Checking parameters for BSIM 3.2 model cmosn
Warning: Pd = 0 is less than W.
Warning: Ps = 0 is less than W.
Warning: v1: no DC value, transient time 0 value used
Warning: v2: no DC value, transient time 0 value used
Warning: v3: no DC value, transient time 0 value used

Initial Transient Solution
-----
Node                                Voltage
-----
net__m3-pad3_                       1.12176e-08
a                                    1.8
net__m5-pad3_                       1.12176e-08
b                                    1.8
sum                                 2.24351e-08
c                                    1.8
cout                                1.8
net__m1-pad3_                       1.75765
net__m1-pad1_                       1.8
v1#branch                           0
v2#branch                           0
v3#branch                           0
v4#branch                          -1.88333e-11

No. of Data Rows : 466
ngspice 1 -> meas tran curr_inte integ v3#branch from=1u to=200u
curr_inte = -2.48248e-10 from= 1.07831e-06 to= 2.00000e-04
ngspice 2 -> let power_average=(curr_inte*1.8)/(200u-1u)
ngspice 3 -> print power_average
power_average = -2.24546e-06
ngspice 4 ->
```

## References:

- 1.LAHAF: Low-power, area-efficient, and high-performance approximate full adder based on static CMOS: <https://www.sciencedirect.com/science/article/pii/S2210537921000226>
- 2.CMOS Inverter Tutorial: <https://www.youtube.com/watch?v=R2y5bXrKBXw>
- 3.Project tutorials: <https://youtube.com/playlist?list=PLOE9jhuDlj9r-XIIgx5PPjpogx7ThS5CB&si=FOjbl6b--Gh2Ag4q>
- 4.Delay calculation for full adder circuit: [https://community.cadence.com/cadence\\_technology\\_forums/f/custom-ic-design/29194/delay-calculation-for-full-adder-circuit](https://community.cadence.com/cadence_technology_forums/f/custom-ic-design/29194/delay-calculation-for-full-adder-circuit)
5. CMOS Full Adder: [https://www.youtube.com/watch?v=AXU\\_J4wr\\_yA](https://www.youtube.com/watch?v=AXU_J4wr_yA)