NYU | TANDON

SHA-256 ASIC for Bitcoin Mining

ECE-GY 9433 Special Topics: Agile System-on-Chip Design

Devashish Gawde
Anish Miryala
Navin Nadar

Agenda

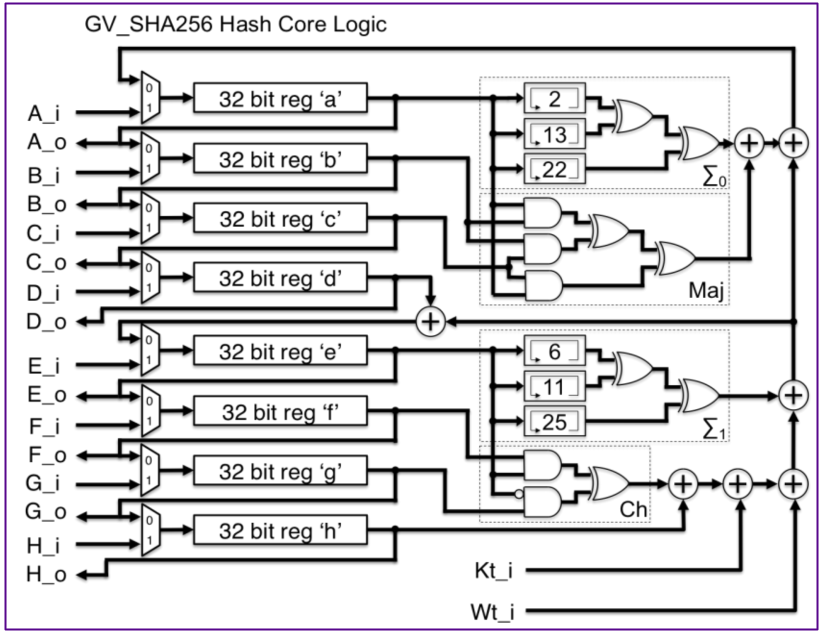
- Motivation
- Introduction
- Architectural Overview
- Challenges
- Verification
- Evaluation
- Conclusion
- References

Motivation

- Cryptocurrency (Bitcoin) paving way towards decentralized currency (future of currency).
- Demands high computation – driven by cryptographic hash functions. (SHA)
- Mining ASICs are both costly and often inaccessible due to limited availability.
- Need to make it more accessible – "**democratize** mining hardware".
- Unlike proprietary ASICs, we would like to contribute to open-source the design of a Bitcoin Mining ASIC.
- Provide a middle ground - a platform that is both affordable and capable of outperforming traditional mining hardware (GPUs, CPUs and also some ASICs!).
- Added advantage – might also contribute to environmental impact!

Introduction – Secure Hash Algorithm

- Takes an input (message) of any size and produces a fixed-size 256-bit (32-byte) output, known as a hash. (ONE WAY FUNCTION!!!)
- Adders become a bottleneck in the design.
- Beneficial to implement a custom adder to optimize for speed and area.
- Chosen adders for analysis:
- Brent-Kung,
- Han-Carlson,
- Kogge-Stone,
- Default '+' Operator.



NYU TANDON

Introduction - Adders

- SHA requires a large number of addition operations.
- Designing an adder involves trade-offs – There is no single optimal circuit.
- More Wire tracks – Capacitance – Energy
- More Fanout – Load – Delay
- More Logic levels – Area
- Use of adder generators – synth_opt adders, VLSIFFRA

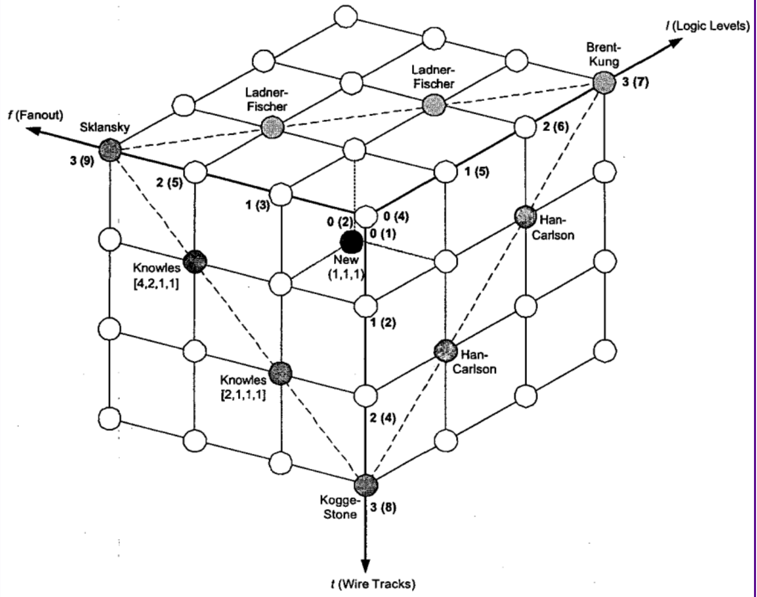
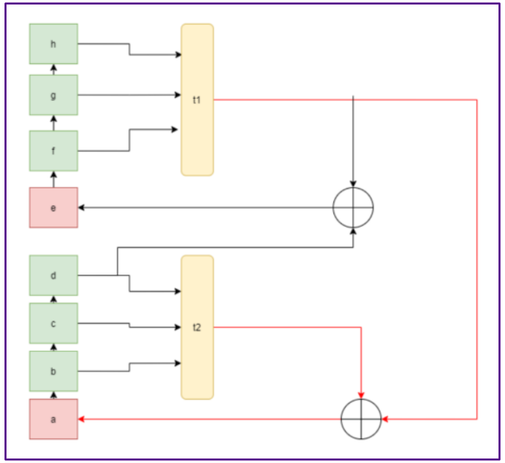


Fig. 3. Taxonomy of prefix graphs

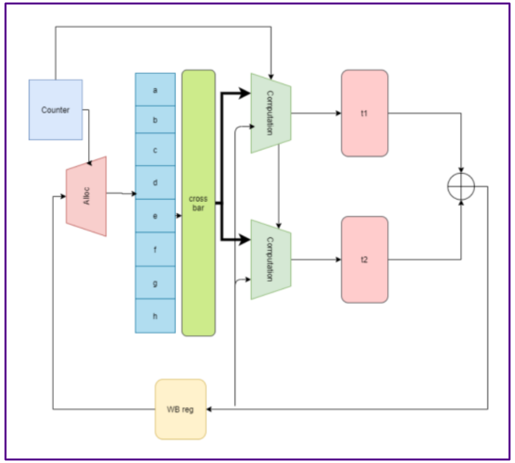
D. Harris, “A taxonomy of parallel prefix networks,” in Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, Nov. 2003, pp. 2213–2217

Architectural Overview

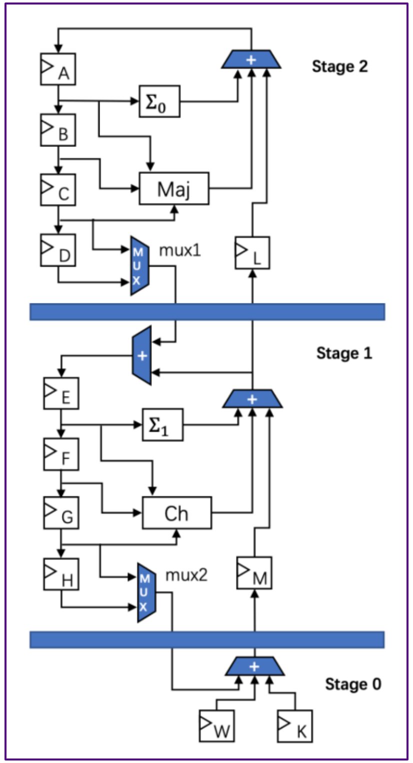
- SHA 256 algorithm can be implemented using various designs.
- Performance metrics depend on the design you choose to implement.



Baseline Naïve Design



Novel Counter Based Design

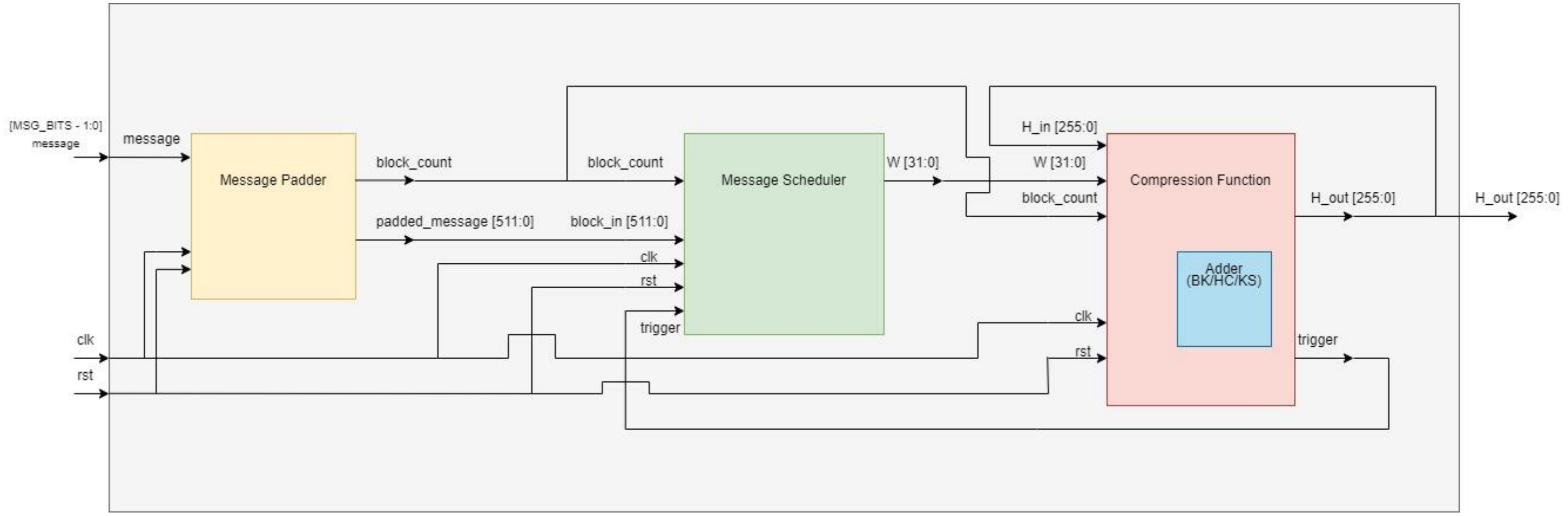


Pipeline Design

NYU TANDON

Architectural Overview

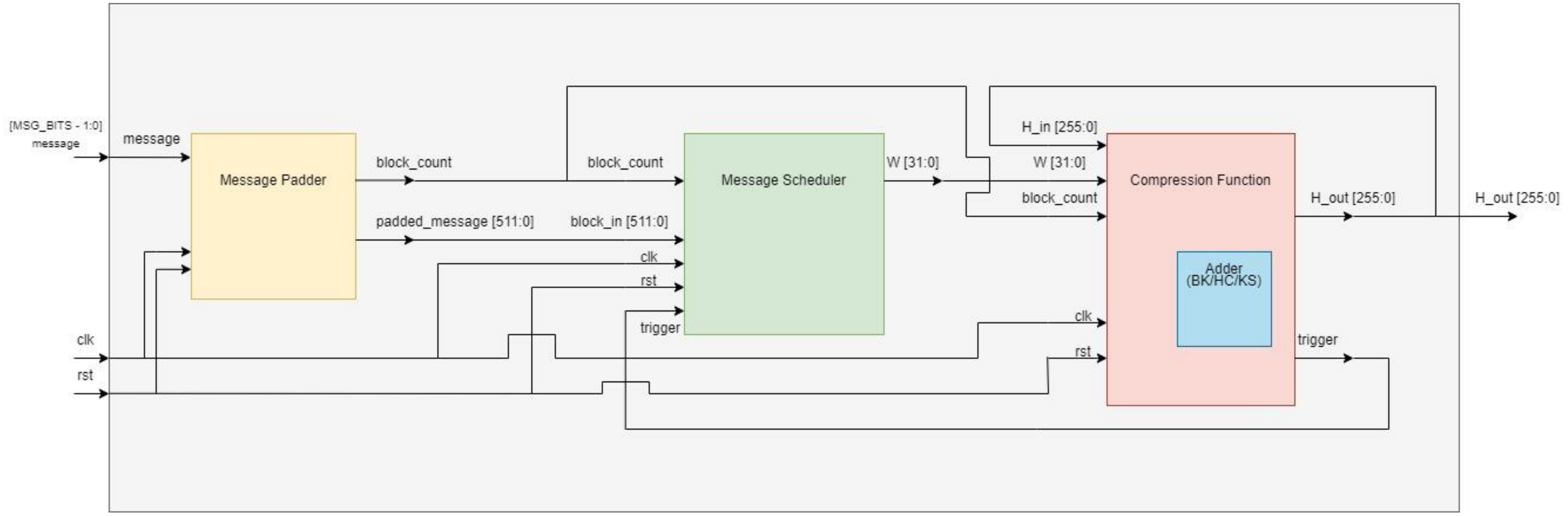
- The message padder in the SHA-256 algorithm ensures input messages are properly formatted by appending a single '1' bit followed by '0's and the message length, producing proper 512-bit blocks for hashing.



NYU | TANDON

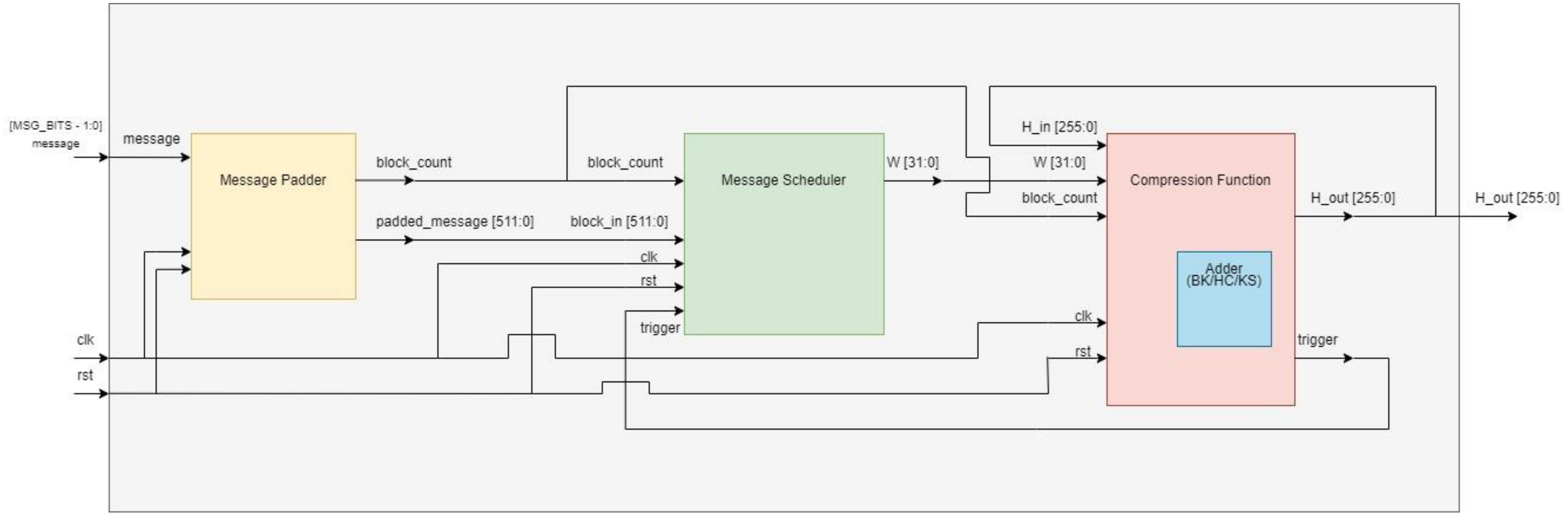
Architectural Overview

- The message scheduler in the SHA-256 algorithm organizes the input message into set of 64 32-bit words for processing, incorporating a set of bitwise operations and constants in each step.



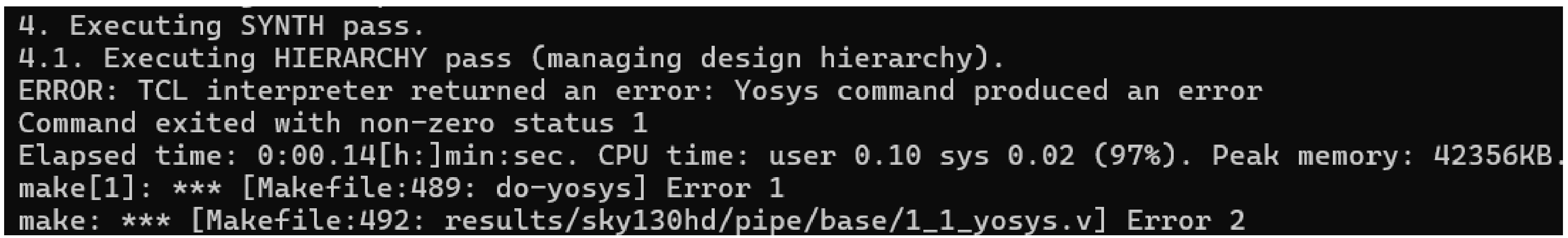
Architectural Overview

- The compression function in the SHA-256 algorithm combines the current hash value with the input data block (words) and a set of constant values through a series of bitwise operations to produce the next hash value.



Challenges

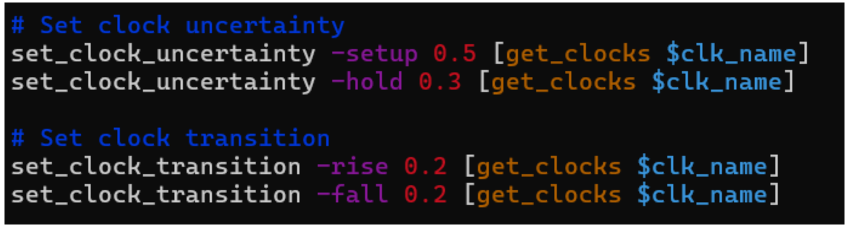
- The architectural designs lack functional accuracy and appropriate test-benches.
- Yosys Synthesis failure.



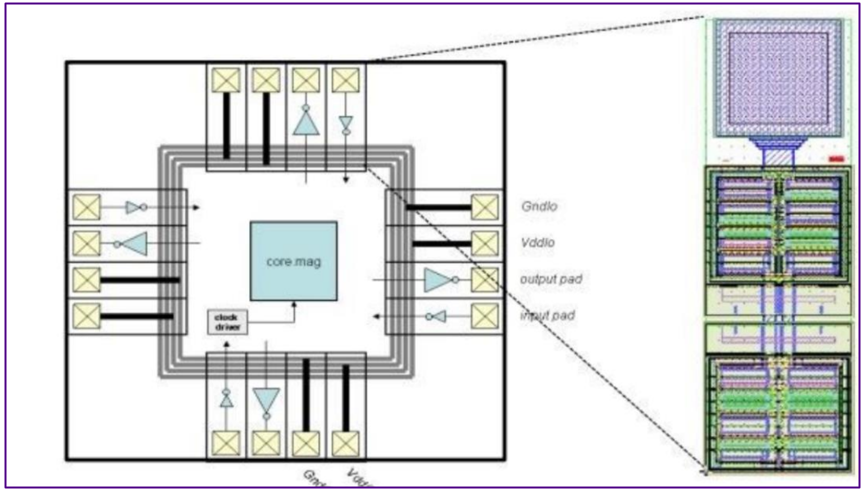
```
4. Executing SYNTH pass.
4.1. Executing HIERARCHY pass (managing design hierarchy).
ERROR: TCL interpreter returned an error: Yosys command produced an error
Command exited with non-zero status 1
Elapsed time: 0:00.14[h:]min:sec. CPU time: user 0.10 sys 0.02 (97%). Peak memory: 42356KB.
make[1]: *** [Makefile:489: do-yosys] Error 1
make: *** [Makefile:492: results/sky130hd/pipe/base/1_1_yosys.v] Error 2
```

Challenges

- GDS-II of chip exhibited inefficient designs.
- I/O Cell Drivers.
- Constraints.

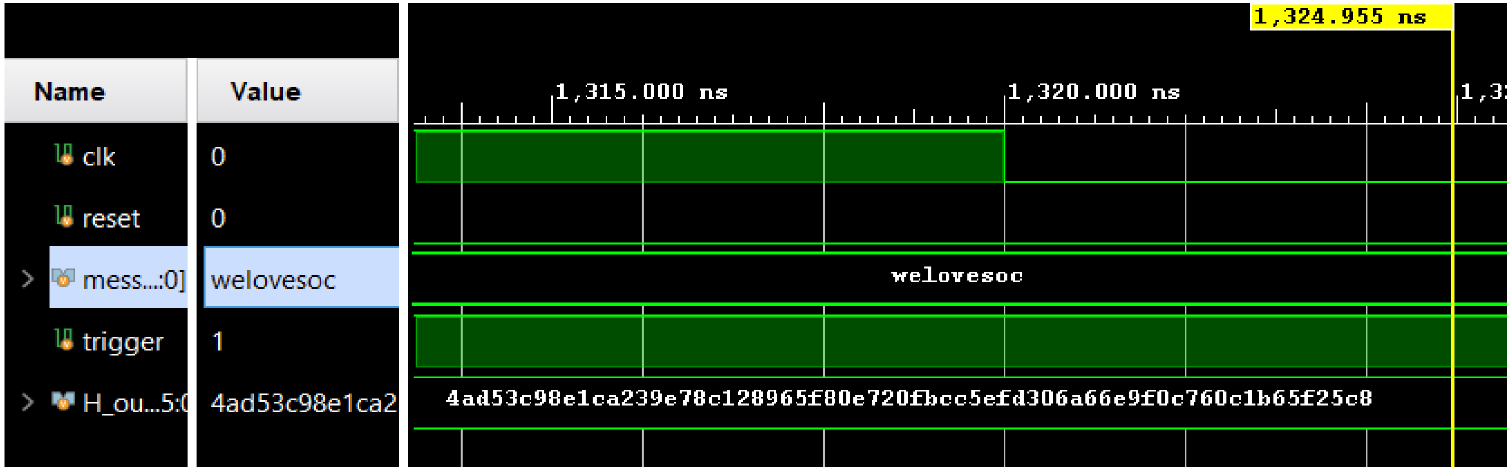


```
# Set clock uncertainty
set_clock_uncertainty -setup 0.5 [get_clocks $clk_name]
set_clock_uncertainty -hold 0.3 [get_clocks $clk_name]

# Set clock transition
set_clock_transition -rise 0.2 [get_clocks $clk_name]
set_clock_transition -fall 0.2 [get_clocks $clk_name]
```



Verification – Functional

- Conducted Design Verification using Xilinx Vivado.



Name	Value
clk	0
reset	0
mess...:0]	welovesoc
trigger	1
H_ou...5:0	4ad53c98e1ca2

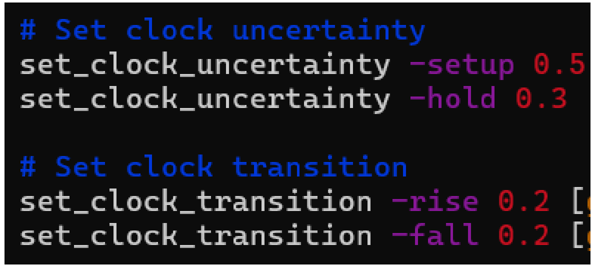
```
welovesoc
4ad53c98e1ca239e78c128965f80e720fbcc5efd306a66e9f0c760c1b65f25c8
```

NYU | TANDON

Verification – Timing

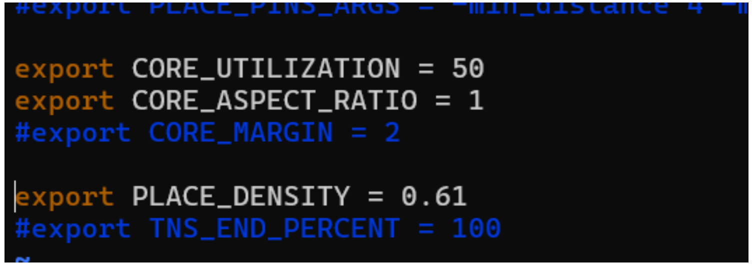
Constraints

- Base clock - 100 MHz (10 ns clock period).
- Top module did not support this latency. Lot of Timing failures, clock period had to be increased to get some slack.
- Used synchronous resets as it simplifies timing considerations to some extent



```
# Set clock uncertainty
set_clock_uncertainty -setup 0.5
set_clock_uncertainty -hold 0.3

# Set clock transition
set_clock_transition -rise 0.2 [
set_clock_transition -fall 0.2 [
```

Constraints.sdc

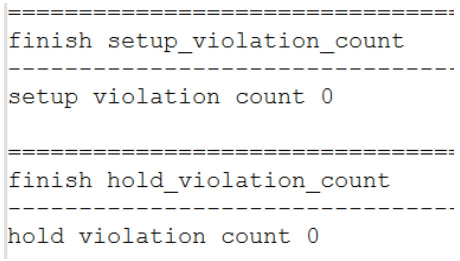


```
#export PLACE_PINS_ARGS = -min_distance 4 -m

export CORE_UTILIZATION = 50
export CORE_ASPECT_RATIO = 1
#export CORE_MARGIN = 2

export PLACE_DENSITY = 0.61
#export TNS_END_PERCENT = 100
~
```

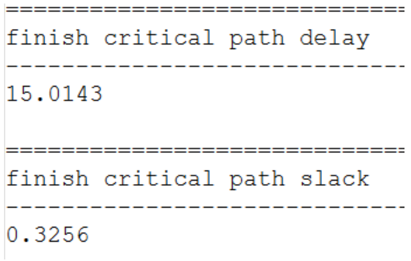
Confik.mk

Verification – Timing

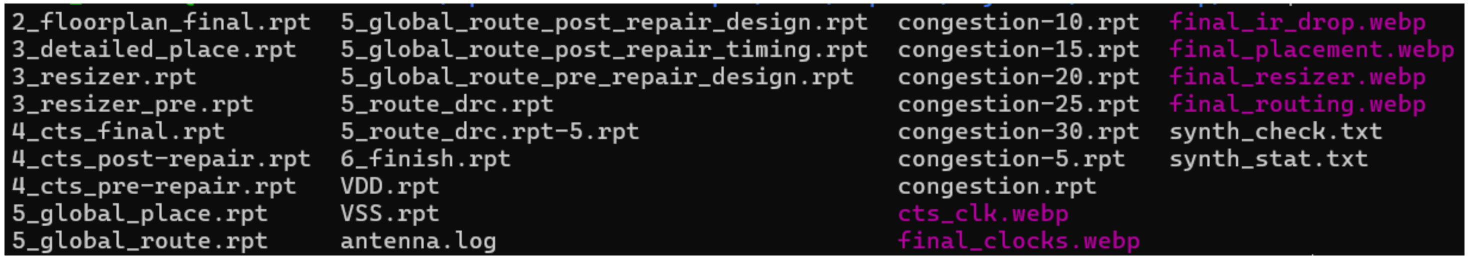
- **Report**



```
==================================
finish setup_violation_count
----------------------------------
setup violation count 0

==================================
finish hold_violation_count
----------------------------------
hold violation count 0
```

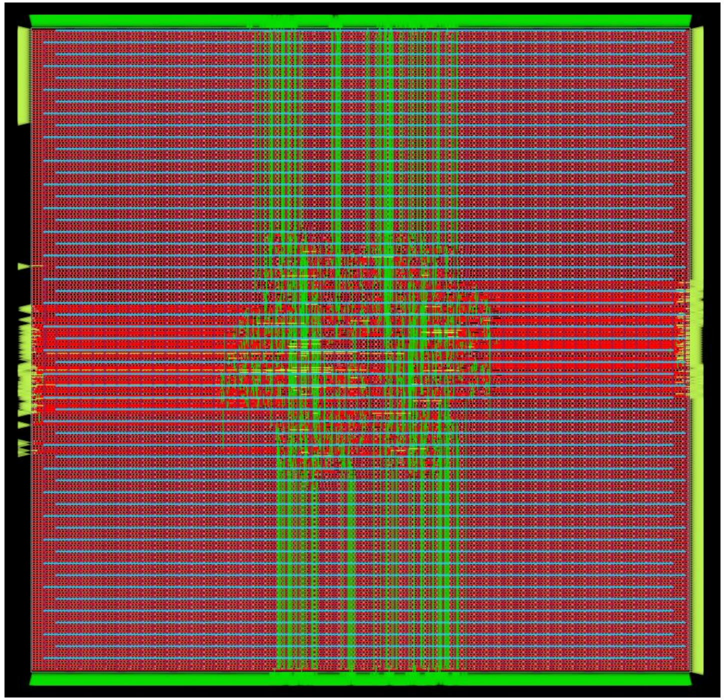


```
==============================
finish critical path delay
------------------------------
15.0143

==============================
finish critical path slack
------------------------------
0.3256
```

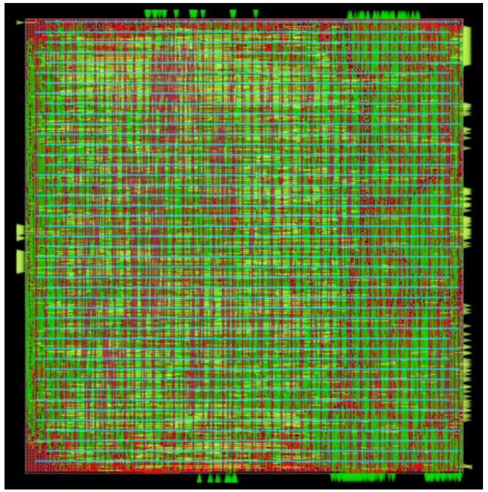


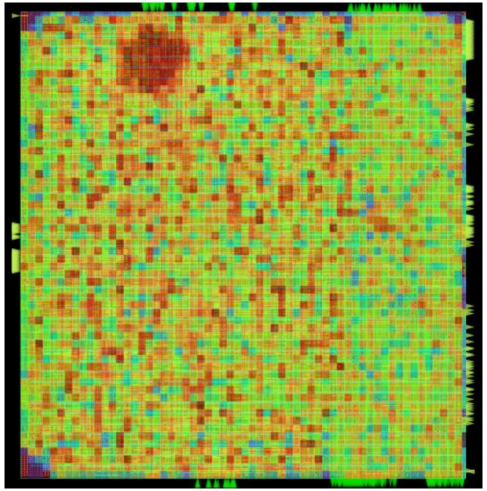
```
2_floorplan_final.rpt    5_global_route_post_repair_design.rpt  congestion-10.rpt  final_ir_drop.webp
3_detailed_place.rpt     5_global_route_post_repair_timing.rpt  congestion-15.rpt  final_placement.webp
3_resizer.rpt            5_global_route_pre_repair_design.rpt   congestion-20.rpt  final_resizer.webp
3_resizer_pre.rpt        5_route_drc.rpt                        congestion-25.rpt  final_routing.webp
4_cts_final.rpt          5_route_drc.rpt-5.rpt                  congestion-30.rpt  synth_check.txt
4_cts_post-repair.rpt    6_finish.rpt                           congestion-5.rpt   synth_stat.txt
4_cts_pre-repair.rpt     VDD.rpt                                congestion.rpt
5_global_place.rpt       VSS.rpt                                cts_clk.webp
5_global_route.rpt       antenna.log                            final_clocks.webp
```

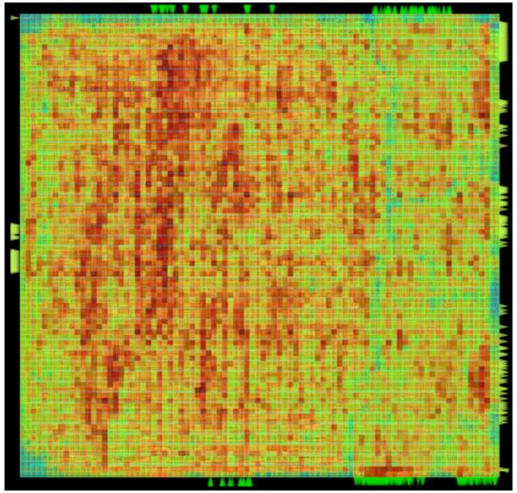
GDS-II


Initial Design GDS-II

GDS-II

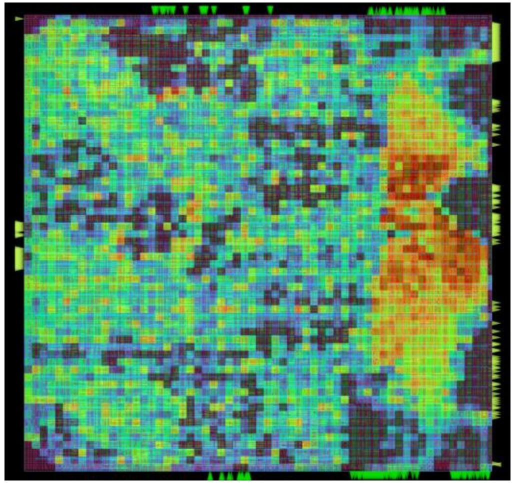

GDS-II

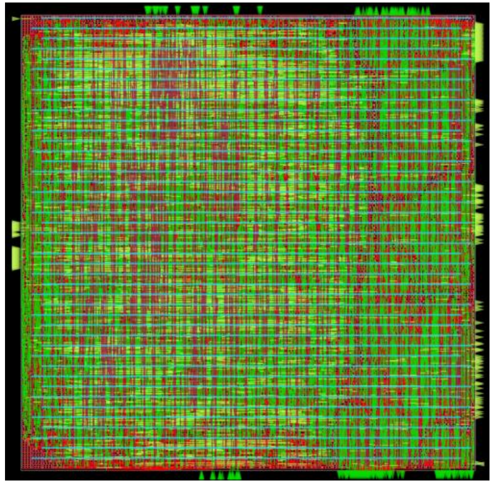

Placement-Density

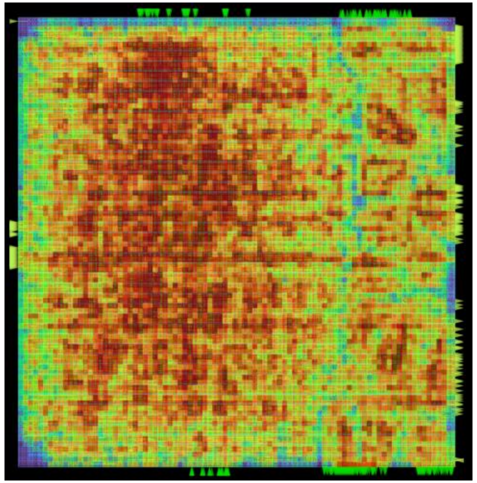

Routing-Congestion

NYU | TANDON

GDS-II


Power-Density


IR-Drop


Estimated-Congestion

NYU | TANDON

Evaluation - Core

- Same "Performance"?
- Keeping performance similar, compare other metrics for different adders.
- Core with the Brent-Kung adder consumes least power.

Naïve Core	Latency(ns)	Area(u^2)
Reference paper	9.7	229453
BK-adder	10	45467
HC-adder	10	52577
KS-adder	10	65952

Table. Synthesis Results

Naïve Core	GHash/s	Power (W)	MHash/J
Reference paper	0.82	12.8	64
BK-adder	0.80	10.5	76
HC-adder	0.80	13.6	59
KS-adder	0.80	18.1	44

Table. Performance Metrics

NYU TANDON

Evaluation - Top Module

- Compare to CPU and GPU?

	GHash/s	Power (W)	MHash/ J
Xeon Gold 6240	0.14	165	0.83
Tesla V100	1.33	250	3.69

Naïve Top	Latency(ns)	Area(u^2)
'+'	14	189353
BK-adder	14	200600
HC-adder	14	206234
KS-adder	14	215845

Table. Synthesis Results

Naïve Top (512 HE)	GHash/s	Power (W)	MHash/ J
'+'	0.60	12.6	48
BK-adder	0.60	20	30
HC-adder	0.60	14.5	41
KS-adder	0.60	24.5	24

Table. Performance Metrics

NYU | TANDON

Conclusion

- We successfully designed and verified Baseline Naïve design and implemented it using OpenROAD Flow-Scripts (ORFS); almost matched the paper performance.
- We successfully designed and verified Pipeline design.
- We faced multiple challenges during our implementation process.
- We evaluated various performance metrics for our design.

Future Work

- Implement Pipeline design with all the adders using ORFS.
- I/O cells incorporation.

References

[1] Alex De Vries, Ulrich Gallersdörfer, Lena Klaaßen, and Christian Stoll. Revisiting bitcoin's carbon footprint. Joule, 6(3):498–502, 2022.

[2] Camilo Mora, Randi L Rollins, Katie Taladay, Michael B Kantar, Mason K Chock, Mio Shimada, and Erik C Franklin. Bitcoin emissions alone could push global warming above 2 c. Nature Climate Change, 8(11):931–933, 2018.

[3] Yiqui Sun, Haichao Yang, Wentao Zhang, and Yufeng Gu. Asic design for bitcoin mining. 2021.

[4] L. Dadda, M. Macchetti, and J.Owen. The design of a high speed asic unit for the hash function sha-256(384,512). In Proceedings Design, Automation and Test in Europe Conference and Exhibition, volume 3, pages 70–75 Vol.3, 2004.

[5] Matthew Vilim, Henry Duwe, and Rakesh Kumar. Approximate bitcoin mining. In 2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), pages 1–6. IEEE, 2016.

[6] Rahul P. Naik. Optimising the sha256 hashing algorithm for faster and more efficient bitcoin mining. 2013.

References

[7] Marco Macchetti and Luigi Dadda. Quasi-pipelined hash circuits. In 17th IEEE Symposium on Computer Arithmetic (ARITH'05), pages 222–229. IEEE, 2005.
[8] D. Harris. A taxonomy of parallel prefix networks. In Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, Nov. 2003, pp. 2213–2217.
[9] Secure hash standard (SHS). Federal information processing standards publication. Fips pub 180-4.

Thank You

NYU | TANDON