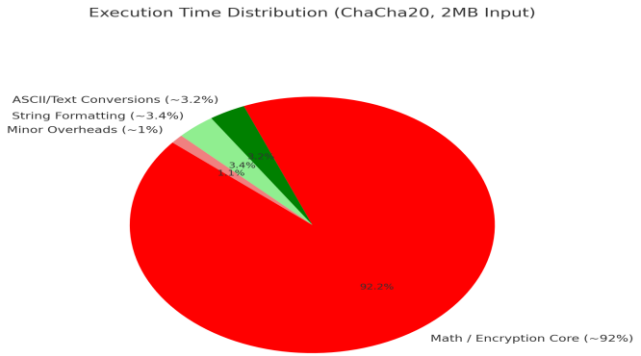
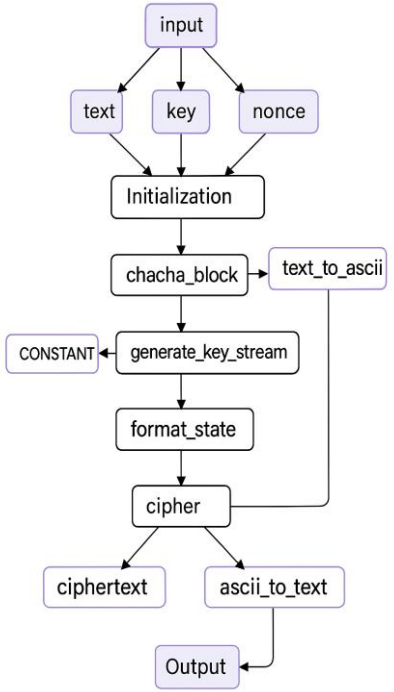
Cryptographic Algorithm (ChaCha20)

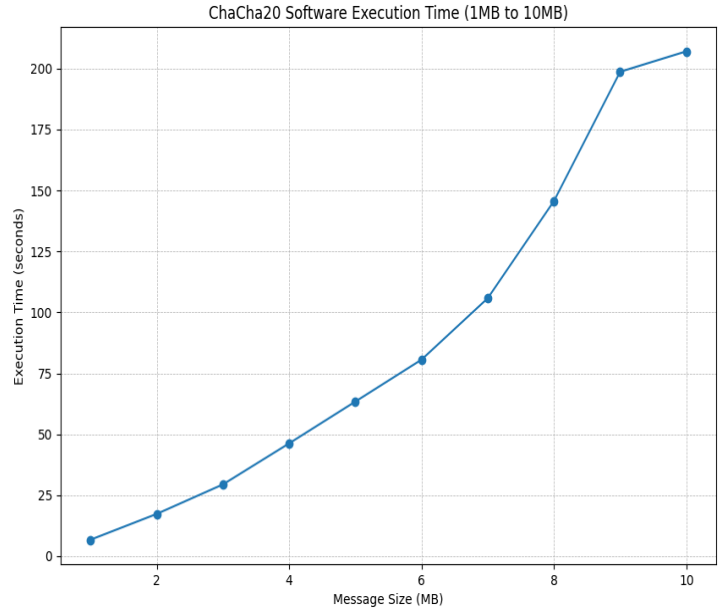


What are you trying to do? I.e., what AI/ML algorithm/workload did you pick?)
I plan to implement a hardware-software co-design for the ChaCha20 stream cipher, a lightweight encryption algorithm commonly used in AI/ML edge applications for securing data streams.
It is a supportive cryptographic function integrated in AI/ML systems for privacy-preserving inference, secure model updates, or data transfer.

How have others implemented and/or accelerated this algorithm?
ChaCha20 has been widely implemented in optimized C/C++ libraries such as OpenSSL and Libsodium for CPU/GPU platforms, and also as RTL-based IP cores for FPGAs and ASICs in secure low-power systems. Some hybrid approaches use embedded systems to combine software and hardware acceleration.

What are you doing differently/better/etc.?
I am implementing and accelerating the ChaCha20 stream cipher algorithm using hardware-software co-design. The goal is to offload the core ChaCha20 block to a hardware accelerator written in SystemVerilog while retaining the control and ASCII/hex conversion logic in Python.





What have you accomplished so far?
I have completed the Python reference implementation of ChaCha20, extracted and translated the core cryptographic block into SystemVerilog, built a cocotb testbench, and benchmarked the software-only implementation on large input sizes up to 10MB. I also generated visual data-flow diagrams and profiling reports to guide hardware mapping. I have also generated a input file of 2MB for my algorithm

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
3	0.001	0.000	52.806	17.602	d:\PSU 3rd Term\HW for AI & ML\Project\Test.py:52(run_test_for_option)
3	0.003	0.001	52.771	17.590	d:\PSU 3rd Term\HW for AI & ML\Project\ChaCha20.py:160(main)
3	0.000	0.000	26.710	8.903	c:\Users\Lokarjun R\AppData\Local\Programs\Python\Python313\Lib_pylong.py:214(int_from_string)
3	0.000	0.000	26.703	8.901	c:\Users\Lokarjun R\AppData\Local\Programs\Python\Python313\Lib_pylong.py:187(_str_to_int_inner)
24573/3	24.860	0.001	24.860	8.287	c:\Users\Lokarjun R\AppData\Local\Programs\Python\Python313\Lib_pylong.py:202(inner)
3	0.041	0.014	22.133	7.378	d:\PSU 3rd Term\HW for AI & ML\Project\ChaCha20.py:134(cipher)
6	0.093	0.016	19.199	3.200	c:\Users\Lokarjun R\AppData\Local\Programs\Python\Python313\Lib_pylong.py:142(int_to_decimal_string)
6	0.001	0.000	19.105	3.184	c:\Users\Lokarjun R\AppData\Local\Programs\Python\Python313\Lib_pylong.py:103(int_to_decimal)
1572858/6	18.020	0.000	18.020	3.003	c:\Users\Lokarjun R\AppData\Local\Programs\Python\Python313\Lib_pylong.py:121(inner)
3	0.014	0.005	11.016	3.672	d:\PSU 3rd Term\HW for AI & ML\Project\ChaCha20.py:40(ascii_to_text)
154	0.201	0.001	9.987	0.065	d:\PSU 3rd Term\HW for AI & ML\Project\ChaCha20.py:13(to_hex)
183	0.020	0.000	9.631	0.053	{built-in method builtins.print}
133	0.861	0.006	3.904	0.029	{method 'join' of 'str' objects}
6000003	2.556	0.000	3.043	0.000	d:\PSU 3rd Term\HW for AI & ML\Project\ChaCha20.py:44(<genexpr>)
9	2.927	0.325	2.927	0.325	c:\Users\Lokarjun R\AppData\Local\Programs\Python\Python313\Lib_pylong.py:51(compute_powers)
2	1.703	0.852	2.641	1.321	d:\PSU 3rd Term\HW for AI & ML\Project\ChaCha20.py:36(text_to_ascii)
4000157	0.577	0.000	0.577	0.000	{method 'zfill' of 'str' objects}
6000000	0.488	0.000	0.488	0.000	{built-in method builtins.chr}
4000000	0.333	0.000	0.333	0.000	{built-in method builtins.ord}
1	0.014	0.014	0.041	0.041	d:\PSU 3rd Term\HW for AI & ML\Project\ChaCha20.py:32(hex_to_bin)

What will you do next and what remains to be done until you can declare success?
The next steps include completing hardware-software co-simulation, comparing performance between software and RTL execution, and finalizing documentation and benchmarks. Success will be defined by demonstrating correct, cycle-accurate behavior of the RTL block and performance improvement over the software baseline.