Digital System Design

Lab # 03
Integrating Fast Adders with UART

Name	*Fawad Ahmed*
Reg #	*DV-DIC-C1-2025-007*
Instructor	*Junaid Khan*
Date	*20th June 2025*

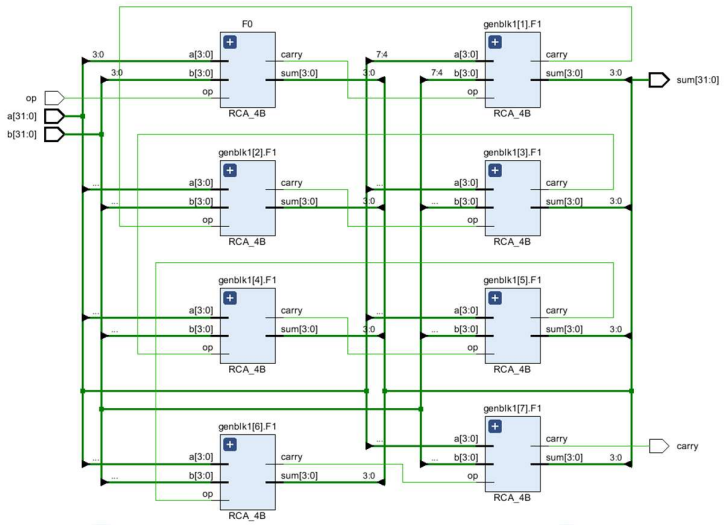
1. In-Lab Tasks:

Integrate Fast Adders with a UART that transmits & receives data from/to a PC. Also analyze the performance & utilization of each adder

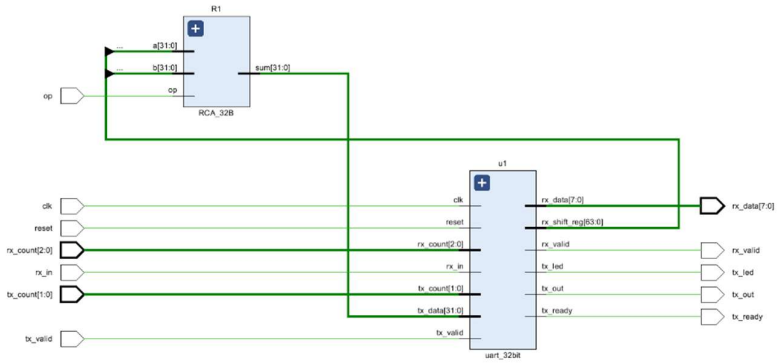
In this task, we will integrate fast adders like (RCA, CLA, CSVA, CNSA & CKSA) with a UART that receives operands of adders through UART and sends their result to the PC. We will discuss each adder which is given below:

i. **Ripple Carry Adders (RCA):**

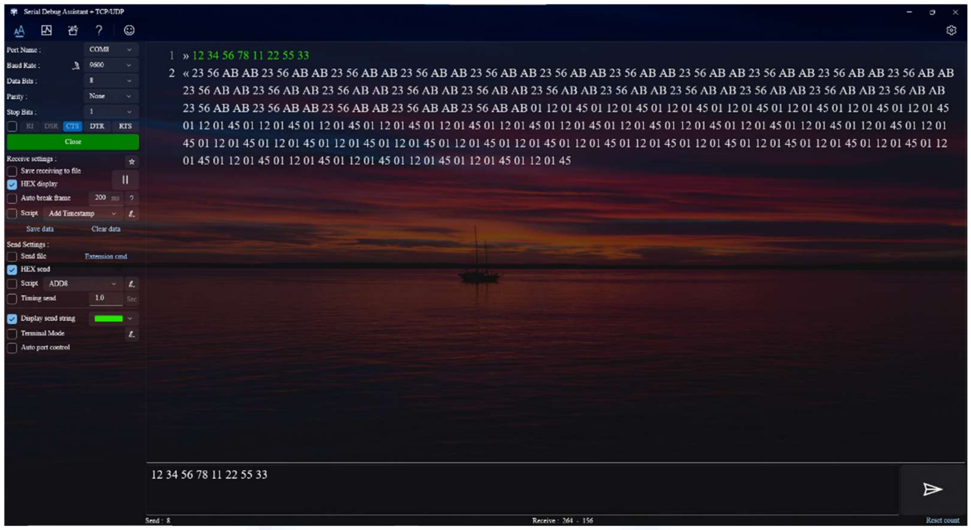
A Ripple Carry Adder (RCA) is a simple digital circuit that adds binary numbers by connecting full adders in series, where each adder passes its carry to the next. It is easy to design but slow for large inputs due to the sequential carry propagation delay. In the given task, we have to design this adder, whose size is 32-bit wide. This can be done by making a 4-bit RCA module, then instantiating 8 times to get a 32-bit RCA using the generate loop (generates multiple modules in given iterations). The diagram is given below:



In this block, it has 32-bit I/Os, and the (op) port is used for the selection bit, indicating whether B is negated or not, so that RCA is used as a subtraction. Now we integrate this adder with the UART, which is given below:



We observed that input (a & b) is received from the PC through the UART. This UART receives 64-bit data that splits into 2 operands & transmits its result of 32 bits to the PC. The result is given below:



We observed that by adding 0x12345678 to 0x11225533, we get a result of 0x2356ABAB. Similarly, by using subtraction, we get 0x01120145. Now we analyze the timing of RCA with UART, which is given below:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.985 ns	Worst Hold Slack (WHS):	0.052 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	339	Total Number of Endpoints:	339	Total Number of Endpoints:	204

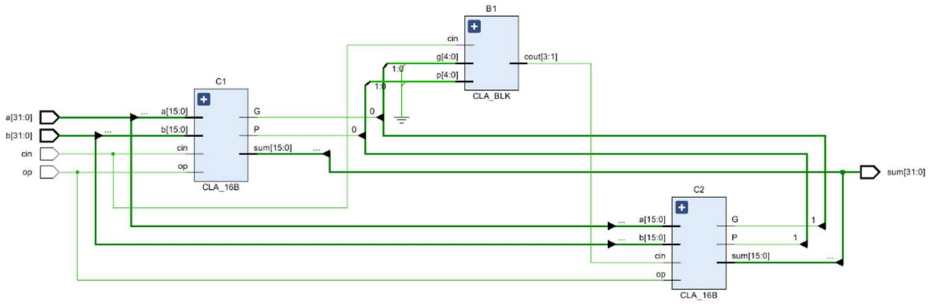
All user specified timing constraints are met.

The timing summary shows that your RCA integrated with UART meets all timing constraints, with no setup, hold, or pulse width violations. The Worst Negative Slack (WNS) of 0.985 ns and Worst Hold Slack (WHS) of 0.052 ns are both positive, meaning signals arrive within the allowed time. Pulse width slack is also safe at 4.5 ns. The resource utilization of this design is given below:

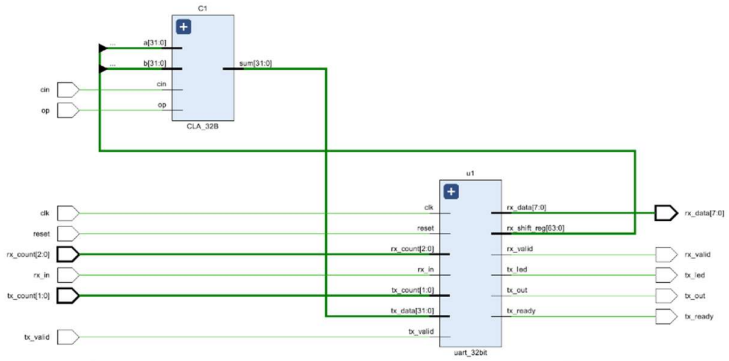
```
1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |  136 |     0 |     63400 |  0.21 |
|   LUT as Logic          |  136 |     0 |     63400 |  0.21 |
|   LUT as Memory         |    0 |     0 |     19000 |  0.00 |
| Slice Registers         |  203 |     0 |    126800 |  0.16 |
|   Register as Flip Flop |  203 |     0 |    126800 |  0.16 |
|   Register as Latch     |    0 |     0 |    126800 |  0.00 |
| F7 Muxes                |    8 |     0 |     31700 |  0.03 |
| F8 Muxes                |    0 |     0 |     15850 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

ii. Carry Look-Ahead Adder (CLA):

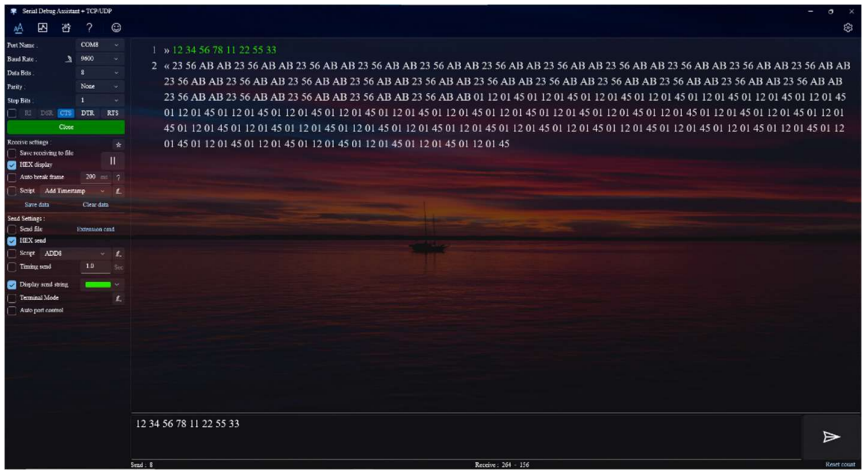
A Carry Look-Ahead Adder (CLA) speeds up addition by computing carry signals in advance using generate and propagate logic. This reduces delay compared to ripple carry adders, enabling faster arithmetic operations. In the given task, we made this adder as a 32-bit wide using a hierarchical approach. We first made 4-bit CLA & then we instantiated to make 16-bit CLA & finally 32-bit using 2 16-bit CLA blocks. The diagram is given below:



It has 2 32-bit I/Ps & 2 I/Ps of cin & op that are used to activate subtraction. Now we integrate with UART, which is given below:



We observed that input (a & b) is received from the PC through the UART. This UART receives 64-bit data that splits into 2 operands & transmits its result of 32 bits to the PC. The result is given below:



We observed that by adding 0x12345678 to 0x11225533, we get a result of 0x2356ABAB. Similarly, by using subtraction, we get 0x01120145. Now we analyze the timing of CLA with UART, which is given below:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.715 ns	Worst Hold Slack (WHS):	0.056 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	339	Total Number of Endpoints:	339	Total Number of Endpoints:	204

All user specified timing constraints are met.

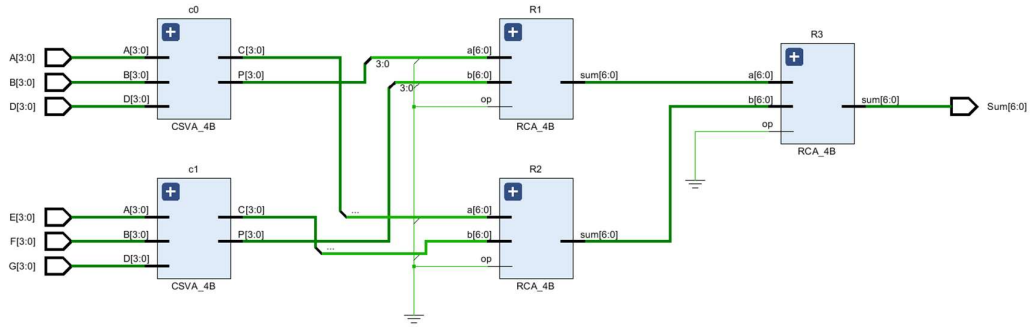
This timing summary for the CLA design shows no timing violations, meaning the design meets all constraints. The Worst Negative Slack (WNS) is 3.715 ns, and Worst Hold Slack (WHS) is 0.056 ns, both are positive, indicating that setup and hold timings are satisfied. The Pulse Width Slack is also safe at 4.5 ns. Overall, the CLA-based design is timing-clean and faster than the ripple carry version due to better WNS. The utilization of this adder is given below:

1. Slice Logic

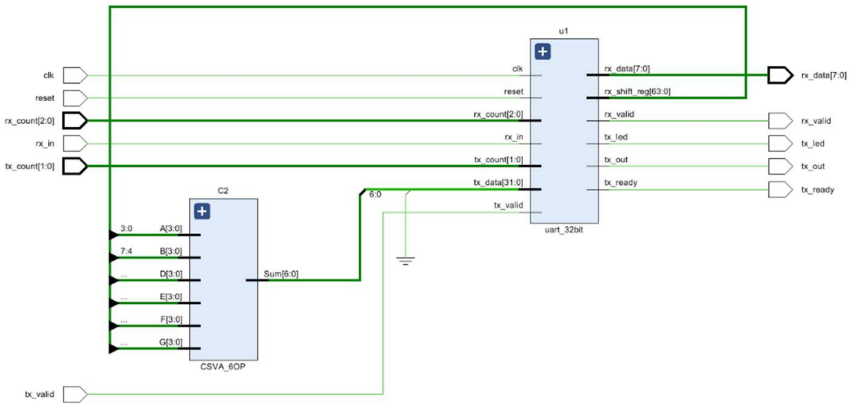
Site Type	Used	Fixed	Available	Util%
Slice LUTs*	184	0	63400	0.29
LUT as Logic	184	0	63400	0.29
LUT as Memory	0	0	19000	0.00
Slice Registers	203	0	126800	0.16
Register as Flip Flop	203	0	126800	0.16
Register as Latch	0	0	126800	0.00
F7 Muxes	8	0	31700	0.03
F8 Muxes	0	0	15850	0.00

iii. Carry Save Adder (CSVA):

A Carry Save Adder (CSA) adds three or more binary numbers simultaneously by saving intermediate carry bits instead of propagating them. It outputs a sum and carry vector separately, which are added later using a fast adder. CSAs are commonly used in multipliers and multi-operand addition for high-speed arithmetic. In the given task, we designed this adder that takes 6 operands with a 4-bit width. In this design, we use 2 CVSA that reduce 3 operands into 2 operands that move to 2 6-bit RCA's that compute their sum & then the results of the 2 RCA's are fed into the final 6-bit RCA that computes both sums. The diagram is given below:



Now we integrate with UART which is given below:



In this design, the CSA receives data from the PC through the UART and transmits its result to the PC. By using an FPGA, we add 1 + 2 + 3 + 4 + 5 + 6 & result is 21. The timing summary of this design is given below:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.336 ns	Worst Hold Slack (WHS):	0.138 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	292	Total Number of Endpoints:	292	Total Number of Endpoints:	177

All user specified timing constraints are met.

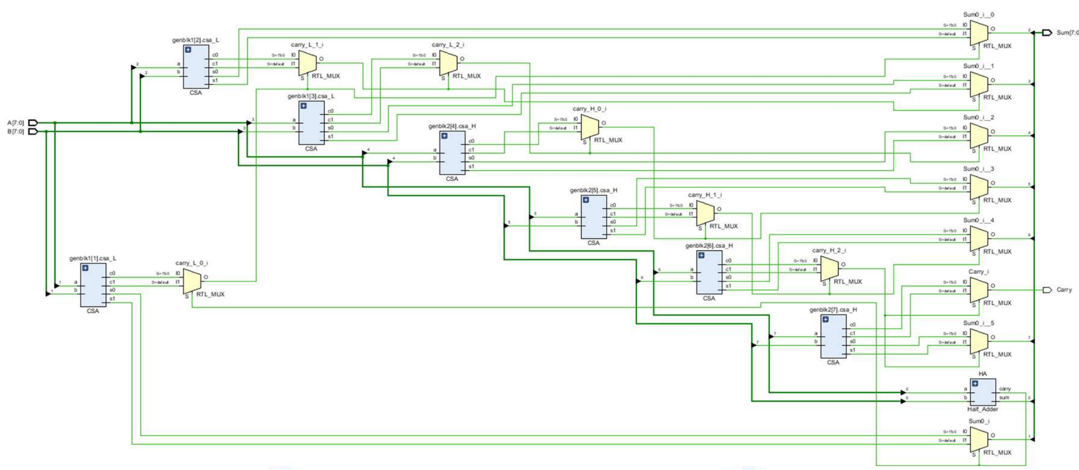
The timing summary for the Carry Save Adder (CSA) shows excellent performance with no timing violations. The Worst Negative Slack (WNS) is 5.336 ns, which is significantly positive, indicating fast setup timing. The Worst Hold Slack (WHS) is 0.138 ns, also positive, ensuring safe data capture. With a pulse width slack of 4.5 ns and zero failing endpoints, the CSA design is fully timing-clean and highly efficient. The resource utilization is given below:

1. Slice Logic

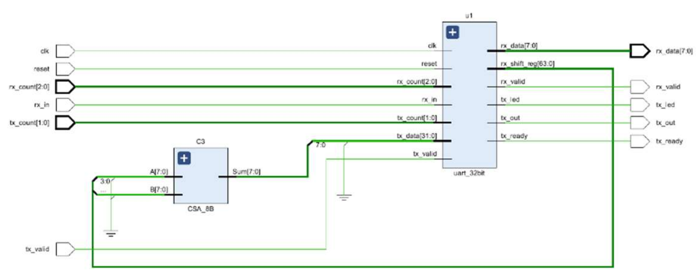
Site Type	Used	Fixed	Available	Util%
Slice LUTs*	117	0	63400	0.18
LUT as Logic	117	0	63400	0.18
LUT as Memory	0	0	19000	0.00
Slice Registers	176	0	126800	0.14
Register as Flip Flop	176	0	126800	0.14
Register as Latch	0	0	126800	0.00
F7 Muxes	8	0	31700	0.03
F8 Muxes	0	0	15850	0.00

iv. Conditional Sum Adder (CNSA):

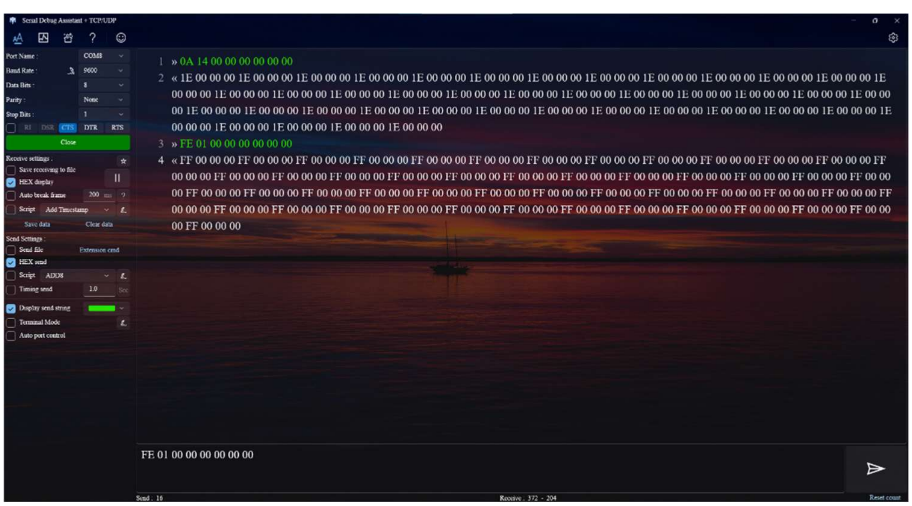
A Conditional Sum Adder (CNSA) speeds up addition by computing sums for both possible carry-in values (0 and 1) in parallel. A multiplexer then selects the correct result once the actual carry is known. This parallelism reduces delay and makes it faster than ripple carry adders. In the given task, we design this adder with 8-bit I/P & O/P using a hierarchical approach. The design is given below:



Now we integrate this design with UART which is given below:



This design doesn't support subtractions. So, this design task 8-bit input that receives from UART and transmits its result to PC. The output is given below:



By adding 0A with 14, and FE with 01, we get results of 1E and FF respectively. The timing summary of this design is given below:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.306 ns	Worst Hold Slack (WHS):	0.154 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	299	Total Number of Endpoints:	299	Total Number of Endpoints:	180

All user specified timing constraints are met.

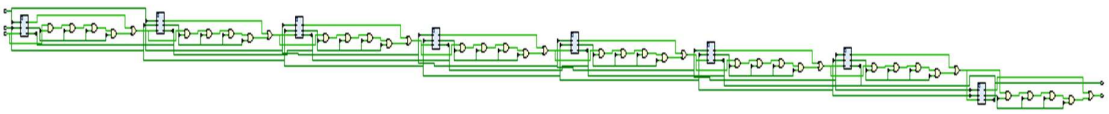
The timing summary of the CNSA demonstrates excellent performance, with a Worst Negative Slack (WNS) of 6.306 ns, indicating a fast and efficient setup timing. The Worst Hold Slack (WHS) is 0.154 ns, which is positive and ensures stable data capture. The Pulse Width Slack of 4.5 ns confirms that clock pulses are sufficiently wide. Overall, the design is timing-clean and optimized for speed. The utilization of this design is given below:

1. Slice Logic

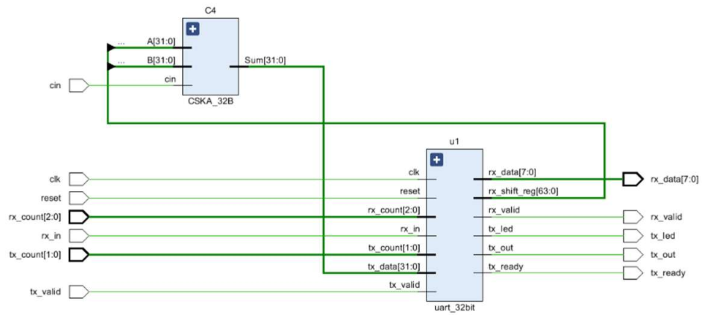
Site Type	Used	Fixed	Available	Util%
Slice LUTs*	113	0	63400	0.18
LUT as Logic	113	0	63400	0.18
LUT as Memory	0	0	19000	0.00
Slice Registers	179	0	126800	0.14
Register as Flip Flop	179	0	126800	0.14
Register as Latch	0	0	126800	0.00
F7 Muxes	8	0	31700	0.03
F8 Muxes	0	0	15850	0.00

v. Carry Skip Adder (CSKA):

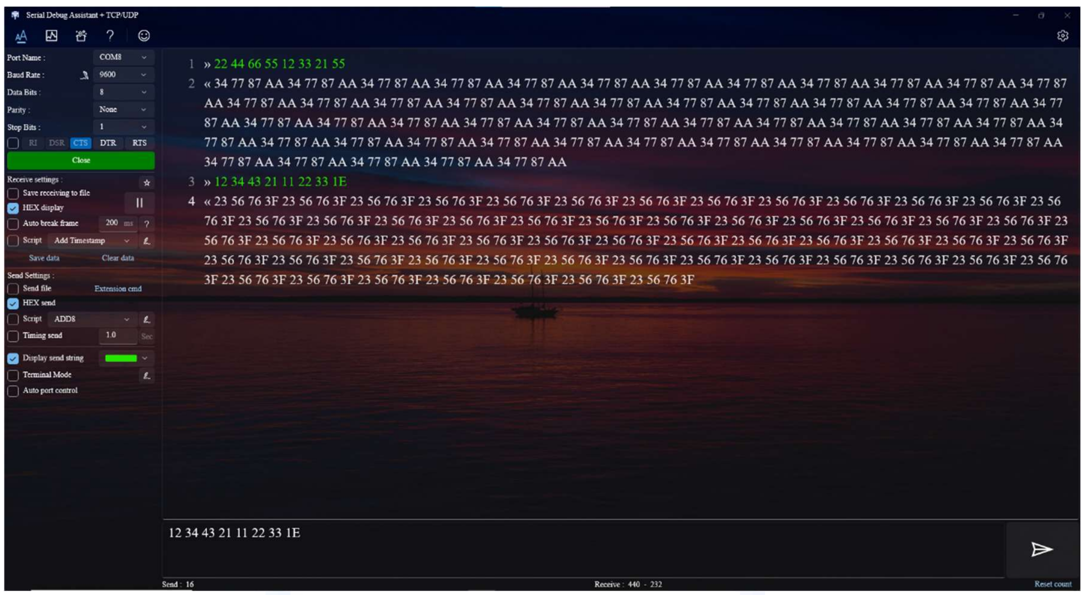
A Carry Skip Adder (CSKA) improves speed by allowing the carry to bypass certain blocks of adders when possible, using skip logic based on propagate signals. This reduces carry delay compared to ripple carry adders. It offers a balance between speed and hardware complexity. In the given task, we designed this adder with 32-bit I/Os. Each carry is selected based on Propagate (a&b). We use a 4-bit RCA to make a 32-bit CKSA. The design is given below:



Now we integrate with UART which is given below:



In this design, it takes a 31-bit I/P from the UART & transmits its result to PC. The cin port is used for subtractions. The output is given below:



We observed that by adding these I/Ps, we get the desired output. The timing summary of this design is given below:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	1.102 ns	Worst Hold Slack (WHS):	0.139 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	339	Total Number of Endpoints:	339	Total Number of Endpoints:	204

All user specified timing constraints are met.

The timing summary of the CSKA shows that all timing constraints are met with a Worst Negative Slack (WNS) of 1.102 ns, indicating acceptable setup timing. The Worst Hold Slack (WHS) of 0.139 ns is positive, ensuring data stability after clock edges. The Pulse Width Slack is 4.5 ns, confirming adequate clock pulse duration. The utilization of this design is given below:

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	136	0	63400	0.21
LUT as Logic	136	0	63400	0.21
LUT as Memory	0	0	19000	0.00
Slice Registers	203	0	126800	0.16
Register as Flip Flop	203	0	126800	0.16
Register as Latch	0	0	126800	0.00
F7 Muxes	8	0	31700	0.03
F8 Muxes	0	0	15850	0.00

vi. Comparison of Fast Adders:

Fast Adder	Performance	Area
RCA	Simple but slow due to sequential carry delay	Smallest, as it covers minimal logic & interconnects
CLA	Faster than RCA using parallel carry computation	Moderate to Higher, as it requires generate/propagate and complex carry logic that grows with bit width.
CSVA	High-speed, as it handles multiple operands efficiently	Moderate as it saves area over full adders when summing multiple operands, but needs post-add.
CNSA	Very fast, as parallel sum paths are selected via MUX	Highest as it duplicates adder blocks for carry=0 and carry=1; uses MUXes for selection.
CSKA	Moderate speed as it optimized carry skipping logic	Small to Moderate, as it adds skip logic per block, slightly increasing the area over RCA.

2. Critical Analysis:

In this lab, we analyze the performance and area of fast adders. Each fast adder is integrated with a UART that transmits/receives the required data. The Ripple Carry Adder (RCA) offers a simple, area-efficient design but suffers from high latency, making it less suitable for high-speed UART communication. Carry Look-Ahead and Conditional Sum Adders significantly improve timing performance due to their fast carry computation, though at the cost of increased area and design complexity. Carry Save Adders are ideal for multi-operand addition scenarios, offering fast intermediate results but requiring a final adder stage, which slightly complicates integration. The Carry Skip Adder provides a balanced compromise, improving speed over RCA with moderate hardware overhead. Overall, the optimal choice depends on application needs, whether low power and area (RCA) or high-speed transmission (CLA or Conditional Sum) is prioritized.