Rapid Synchronization Recovery from Single Event Effects in the Aurora 64b/66b Protocol

Anatoliy Martynyuk, Hongjiang Cai, Scott Hauck, Timon Heim, Shih-Chieh Hsu, and Geoffrey Jones

***Abstract*—The Aurora 64b/66b line encoding is a Xilinx communication protocol utilized for high-speed serial communications. The protocol offers several useful features but suffers from a relatively slow resynchronization scheme that can be triggered by periodic single event effects (SEEs), particularly in high radiation environments. To reduce data losses using the Aurora protocol we developed a replacement recovery scheme for the receiver which drastically improves resynchronization speeds with reasonable costs in device resources. Our new resynchronization scheme, HSn, can help the system recover from an SEE rapidly by evaluating multiple possible header positions simultaneously, and then confirming the true header position. Compared to the existing resynchronization scheme, the new scheme with HS8 reduces block loss by a factor of 98x while only using 0.23% LUTs and 0.13% FFs of the target FPGA [10].**

***Index Terms*—64b/66b Protocol, FPGA, Single Event Effect, Synchronization Recovery.**

Anatoliy Martynyuk was with Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: anatoliym2@gmail.com)

Hongjiang Cai is with Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: c1003183598@gmail.com)

Scott Hauck is with Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: hauck@uw.edu)

Timon Heim is with Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA (e-mail: theim@lbl.gov)

Shih-Chieh Hsu is with Department of Physics, University of Washington, Seattle, WA 98195 USA (e-mail: schsu@uw.edu)

Geoffrey Jones is with Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: geoffhjones@msn.com)

I. INTRODUCTION

Data communications, especially in loss-prone environments, require special data encoding protocols to provide high speed, good reliability, and good physical layer properties. High reliability can be achieved at a cost in bandwidth, requiring a careful balancing of these two constraints. However, any realistic system will simply statistically reduce the chance of data loss, and systems must still be in place for recovery when data loss occurs.

At the Large Hadron Collider, we utilize a large number of Xilinx FPGAs, and high-speed data links, aggregating the huge data flows generated by this particle accelerator experiment. This environment requires efficient protocols to handle the data-rates and communication errors. Although the radiation environment within such a high-energy physics experiment is more extreme than in many other environments, the issue of error recovery is important in many such contexts.

Error recovery in traditional systems has tended to focus on flipped bits. However, radiation can induce SEEs (Single Event Effects), such as glitches on the clock line, that can effectively cause bits to be added or deleted, thus misaligning the subsequent communications stream [5][6][11]. In this paper, we consider techniques to quickly adapt to such insertions and deletions and demonstrate practical techniques to quickly recover from these effects.

II. AURORA PROTOCOL

The Aurora Protocol is a link layer protocol developed by Xilinx for high-speed communications. The protocol is available in two data formats: 8b/10b and 64b/66b. The protocol format determines the bit length of blocks, and the number of data bits vs. “header” bits in each block. The 8b/10b variation utilizes 10-bit blocks with 8 payload bits. The 64b/66b variation

similarly has 64 payload bits in each 66-bit block. This corresponds to an overhead of 25% in the 8b/10b variation and 3.125% in the 64b/66b variation [1][2].

Both variations are designed to provide enough state changes to allow clock recovery, stream alignment, and DC balancing during transmission. The payload bits of each block are scrambled before transmission. The scrambling generally increases the transition density and DC balance of the stream. The two "header" bits, however, are left unscrambled and used for stream alignment. The high transition density and short length additionally allow for clock recovery at the receiver [1][2].

Each protocol variant offers tradeoffs in performance, with overall throughput depending on the number of transceivers and the target line rate of selected transceivers. The 8b/10b protocol supports throughputs up to 84.48 Gb/s, while the 64b/66b variant supports throughputs up to 400 Gb/s. Latencies of the 8b/10b variation generally hovers around 40 cycles of latency, whereas the 64b/66b has a latency of around 55 cycles [3][4].

III. RESYNCHRONIZATION AT THE LARGE HADRON COLLIDER

Electronics development for upgrades to the ATLAS experiment at the Large Hadron Collider (LHC) demonstrates how SEEs disrupt communication in a high radiation environment; similar effects can happen in space applications (though at significantly lower rates), and increasingly in terrestrial systems as chip feature sizes shrink. The LHC ATLAS experiment contains a pixel detector made up of sensing instrumentation. At its innermost layer will be an array of custom silicon pixel readout chips called the ITkPix[12]. ITkPix collects charge data from subatomic collisions within the experiment. This information is organized into frames and forwarded to an FPGA-based data acquisition device (DAQ) called YARR [7].

Communication between the pixel array and the DAQ uses 64b/66b line encoding. This protocol was chosen because it has lower bandwidth overheads than the 8b/10b protocol, and we are willing to tolerate the corresponding occasional data loss.

The 64b/66b protocol has a built-in synchronization scheme. Synchronization is achieved by determining the packet boundary in the receiver's incoming stream. This boundary is indicated by the header bits, which are differentiated from the remaining data by always containing a 0 to 1 or 1 to 0 transition. The receiver monitors the stream to verify that this transition occurs exactly every 66 bits. If such transitions appear consistently the boundary is considered to be found, the sender and receiver are synchronized, and the receiver will correctly distinguish blocks within the stream. However, if the bits checked in some blocks do not contain a transition (i.e., they are from the data field rather than the header) then the DAQ needs to change which position it is assuming is the header. This is accomplished by "slipping" the stream, effectively dropping a bit from the stream to change the checked indices. With enough slips, the receiver can scan through all 66 possible bit indices. Since header bits are guaranteed to always have a transition, and other bits are scrambled to ensure essentially a 50%/50% chance of transition or not, observing the stream for multiple cycles can reliably find the correct header position in a well-formed stream. The synchronization process is defined by the block sync FSM in IEEE 802.3ae [10].

The DAQ has to resolve data corruption and loss of synchronization as a result of SEEs under high radiation environments. As the LHC moves to a higher intensity mode, it is anticipated to see an increase in the number of SEEs. As a result, we developed a more efficient resynchronization scheme to detect and recover from SEEs more quickly [9].

DEVICE COMPONENT	SEU RATE ESTIMATION IN HL-LHC
PIXEL UNPROTECTED LATCH MEMORY	4.6 bit flips per second per FE chip
PIXEL TMR LATCH (WITHOUT CORRECTION)	7 bit flips per minute per FE chip
PIXEL TMR LATCH (WITH CORRECTION)	0.2 bit flips per hour per FE chip
CLOCK & DATA RECOVERY	4 SEEs per minute per chip (RD53CDR proto)

Table1: SEU Rate Estimates for Future High Luminosity LHC[8]

IV. YARR'S INTRINSIC RECOVERY SCHEME

The existing YARR DAQ has its own recovery scheme built in the Aurora-style single RX lane. It leverages the "bitslip module" in ISERDESE2, one of Xilinx primitives used to build the deserializer in YARR, to effectively recover from bit drops and bit adds caused by SEEs [10].

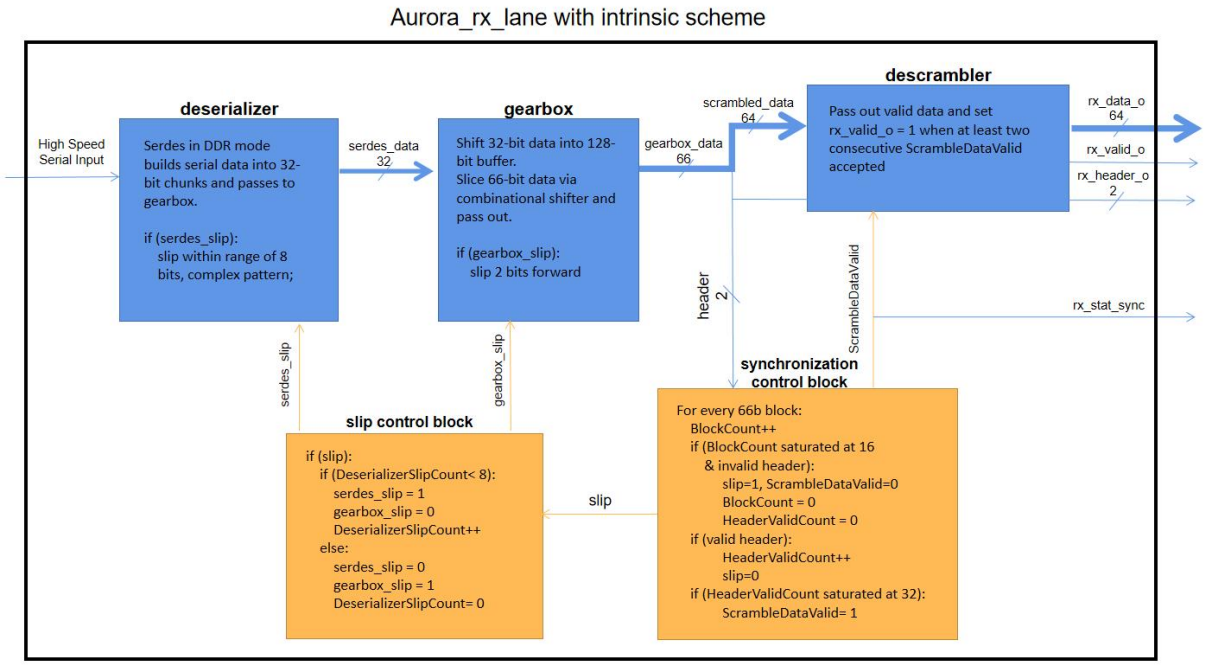

Figure 1: Block diagram of YARR's 64b/66b intrinsic recovery scheme.

The diagram (Fig. 1) above illustrates the mechanism of YARR's intrinsic recovery scheme, based on the Xilinx Aurora protocol. Serial data sent from ITkPix is first passed to the deserializer and built into a 32-bit chunk. This 32-bit chunk is then passed into the 128-bit buffer in the gearbox. The gearbox will then send the 66-bit data, including the 2-bit header and 64-bit scrambled data, from the 128-bit buffer.

Assume for the moment that the system was synchronized and locked on a certain position of the 66-bit range. When a SEE occurs, the synchronizing position might change, so the transmission of 66-bit blocks is no longer synchronized. The system can detect this desynchronization by checking the header of each 66-bit block. When a wrong header appears at the previously locked position, the system will reset HeaderValidCount to zero, flag the transmitting data as invalid, and turn on slip mode to find a new lock position.

This recovery scheme evaluates headers at different positions by bit slip. In slip mode, the DDR SERDES will perform slipping behavior that shifts the data stream back-and-forth within a range of 8 bits, but mostly in a single direction. After 8 slips, the SERDES snaps back to the original position. Then, the slip control block will disable SERDES slip and enable gearbox slip. The gearbox slip is achieved by freezing the shifter. Since the gearbox combinational shifter shifts 2 positions every output cycle, freezing the shifter by one output cycle is equivalent to moving the header position by 2 bits. After one gearbox slip, the next 8 slips are handled by the deserializer before another gearbox slip.

When a valid header is found, the system stops slipping and starts counting valid headers. When HeaderValidCount saturates at 32, the system will flag the transmitted data as valid and send out the descrambled data. However, once an invalid header position is found, the system will slip at most once for every 17x66b blocks, presumably to allow the incoming data to stabilize.

V. YARR HEADER SEEKERS (HSN)

The YARR Header Seekers are our newly developed synchronization recovery scheme, which improves recovery speed from bit drops and bit adds within the 64b/66b line encoding. The HSn recovery schemes aim to leverage parallelism and rapid evaluation to increase the number of indices (referred to here as positions) evaluated per 66-bit block.

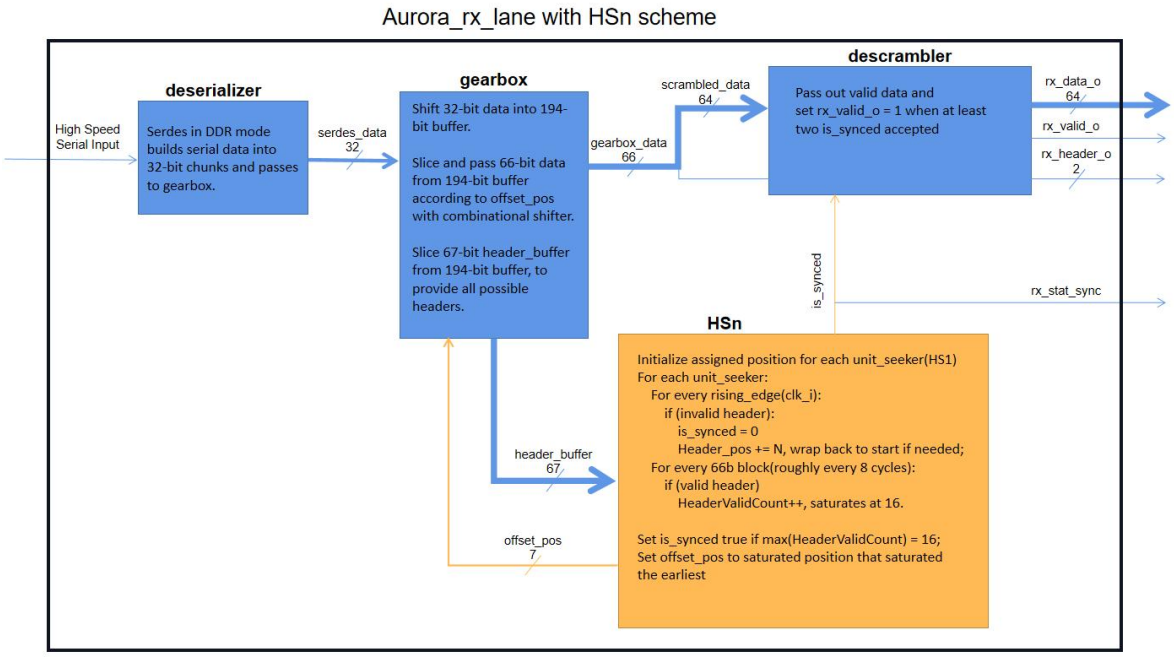

Figure 2: Block diagram of HSn recovery scheme.

Fig. 2 illustrates the mechanism of the HSn recovery scheme, with N parallel seekers, each handling roughly 1/Nth of the possible header positions. Similar to the original scheme, the system detects desynchronization by checking the header of 66-bit blocks. If the header at the examined position violates the protocol, the seeker switches to the next assigned position and resets the valid header counter to 0. Otherwise, the seeker will increment the valid header counter by 1 for each block with a valid header. For this HSn scheme there are three major innovations compared to the original scheme: fail-fast, fast search, and parallelism.

A. Fail-fast

When the original scheme searches for a new lock position, for each position it includes a 16-header "dead time" where it ignores invalid headers. This means that as it scans candidate positions, it takes at least 17 blocks before it can recognize an incorrect position. Instead of waiting for 16x66b blocks, the HSn scheme updates the search position immediately when the seeker finds an invalid header.

B. Fast search

In addition to fail-fast, the HSn scheme further improves search efficiency by evaluating headers at a higher rate. The original scheme checks the validity of one header for every valid 66b block, which arrives roughly every 8 internal clock cycles. This is necessary because the header checker looks at the header on the output of the gearbox, which can only "slip" roughly every 8 clock cycles.

Our HSn system instead "sniffs" the data inside the gearbox, allowing each seeker to check a potential header position every clock cycle. On an incorrect header, it can move quickly to a new candidate position. Note that for a position that has a legal header (either because it is the correct position, or the 50% of the time that scrambled data will happen to have a transition), it must wait for a new 66b block, which appears at the next 8 clock boundary.

When 16 consecutive 66b blocks with valid headers are detected, the seeker will send out the current position offset and a signal indicating that that offset is valid. We choose 16 because statistically, the chance of randomly getting 16 consecutive headers at a non-header position is 2^(-16), or about 1 in 65k, which is sufficiently unlikely [10].

Fig. 3 is the flowchart of the seeker in the HSn scheme with n = 1 (only one seeker).

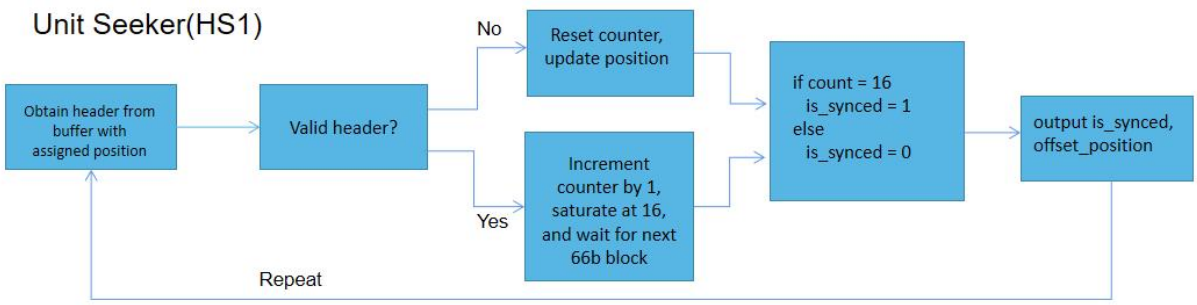

Figure 3:Flowchart of the unit seeker (HS1).

With just the two new mechanisms mentioned above, our new scheme (n = 1) can reduce the block loss by a factor of 44x, while increasing the size of the Aurora receiver to 1.89x.

the misalignment, the longer it would take the recovery system to realign. The exception is at the far right end; because the Xilinx bitslip mechanism in the DDR SERDES sometimes uses slips in the opposite direction, we sometimes get lucky for cases with only a few bits

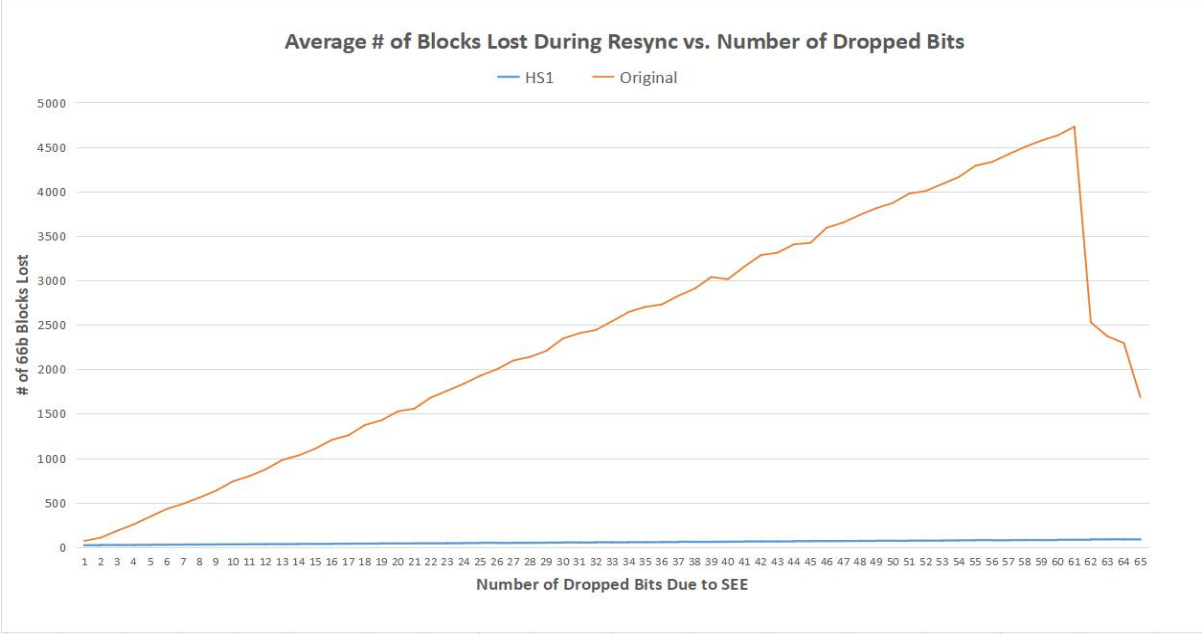

Figure 4: Resynchronization performance of the 64b/66b intrinsic scheme compared to the variant with n = 1 (unit seeker).

Fig. 4 shows the stark difference between the performance of the existing recovery system and our unit seeker variant (n=1). The orange plot demonstrates the near-linear response of the existing recovery system to the misalignment distance. Recall that the 64b/66b recovery system operates on individual “slips” which drop one bit at a time from the stream until the packet boundary is found. It naturally follows that the greater

inserted.

C. *Parallelism*

Beyond just the improvements of HS1, we also use parallelism by employing N seekers, each handling roughly one of the N possible header positions. This allows us to evaluate multiple stream positions simultaneously, which can reduce the time to find the correct header location by up to a factor of n. If n is 66 (the number of bits in each block), the seeker can very quickly find the correct header position because every possible position is evaluated simultaneously. Resynchronization would only be limited by the time it takes to detect that a SEE occurred and to verify that all positions but one are incorrect. However, setting n to 66 has the drawback of requiring significant hardware resources. Instead, by assigning a few seekers to multiple positions each, and rotating between those positions quickly, the benefits of parallelism can be realized without as significant a resource cost [10].

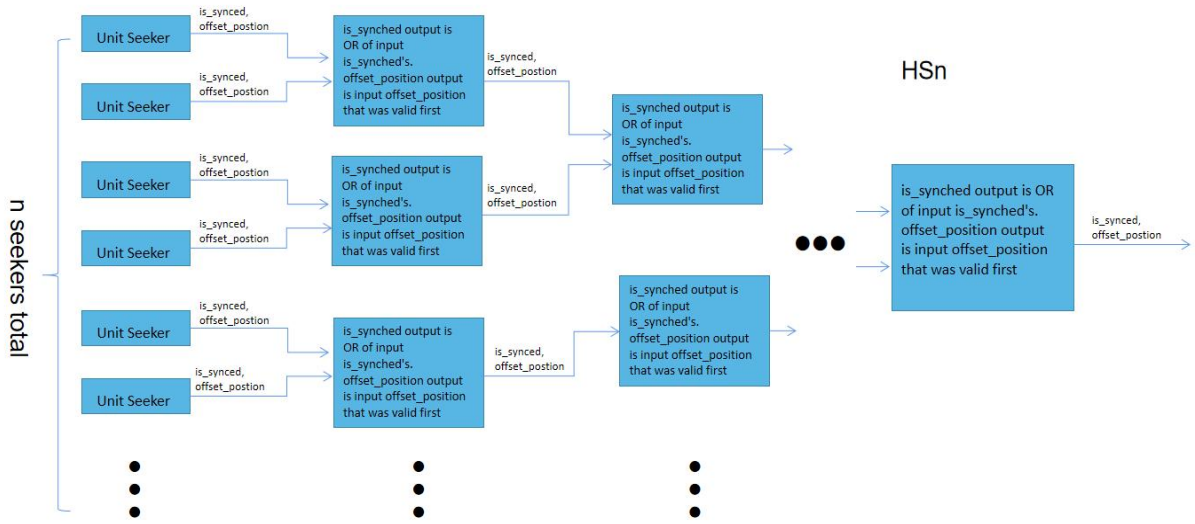

Figure 5:Block Diagram of the HSn recovery scheme.

Note that it is possible to have multiple locked positions (positions with 16 consecutive legal transitions) at the same time. When we are first trying to lock onto a position, there is only a ~1/65k chance of a given head seeker being locked onto an incorrect position. However, during operation, each head seeker at a non-header position will believe it has found a locked position every 65k blocks, or many times a second. To deal with this, we follow two rules: (1) when any head seeker is already locked onto a position, it remains the winner until it sees an invalid header; (2) if multiple head seekers lock at the same time, we choose one arbitrarily.

Fig. 5 illustrates the mechanism of the HSn module.

As the number of head seekers increases the speed of recovery increases, but so do the resource costs. Therefore, we created a parameterized version of the synchronization scheme. By setting the parameter n of the top module, the HSn system generates a binary-tree structure with n seekers. The i-th head seeker searches positions i, i+n, i+2n …

VI. RESULTS

The performance of the existing YARR recovery scheme [13], and our proposed header seeker recovery schemes, were evaluated on their recovery time and resource utilization. Recovery time was measured by the number of blocks lost after a desynchronization and was evaluated for all possible misalignments. To evaluate this recovery response, the stream was systematically misaligned so that the packet boundary was shifted away from its expected position. The boundary is expected every 66 bits, and by dropping a few bits of the preceding block, the subsequent block's header bits will be effectively shifted out of position and into the preceding blocks' data field. Utilizing the Modelsim simulation tool, the stream was misaligned

for values of 1 through 65, and each misalignment was tested 66 times. Taking the average of the 66 tests and graphing them shows the resynchronization response to any possible stream misalignment [10]. The recovery times of typical variants are shown in Fig. 6.

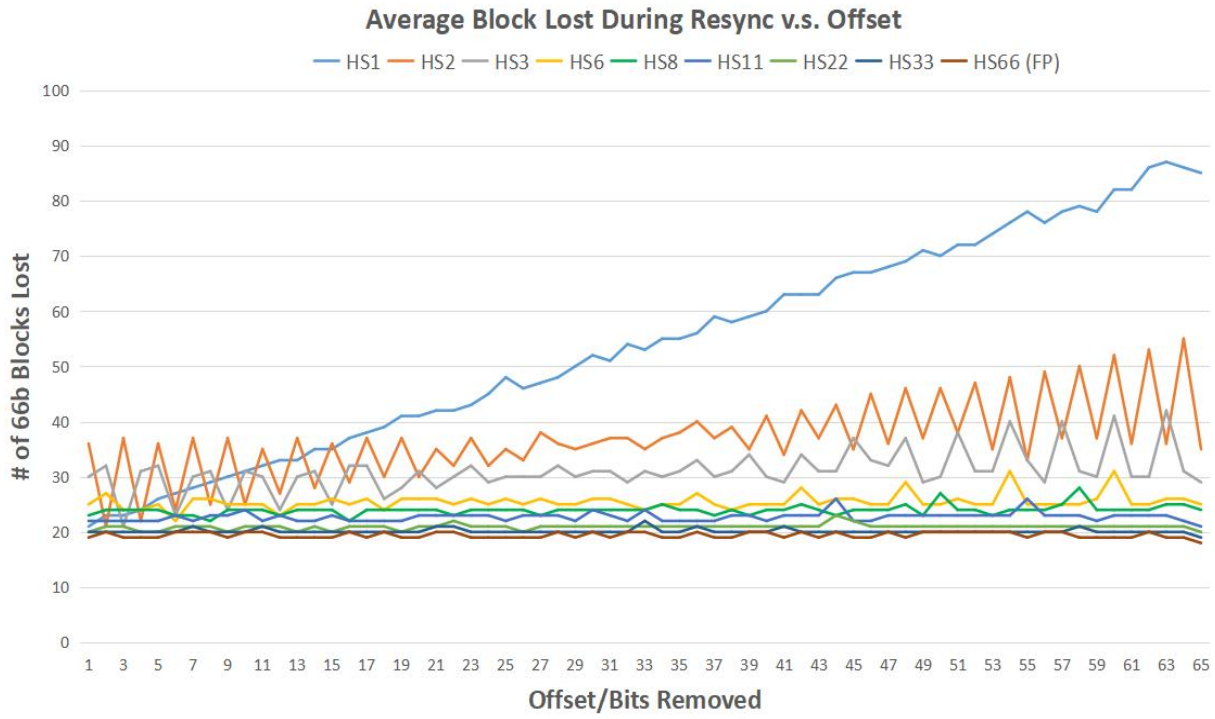

Figure 6: Resynchronization performance of the YARR header seeker variants developed for the LHC.

As seen in Fig. 6, the greater the parallelism, the faster the recovery, though HS11 is nearly as fast as the fully parallel version. The HS66 variant (referred to as FP for Fully Parallel) had the least average blocks lost during recovery, whereas HS1 had the highest average blocks lost. Also, the average blocks lost for HS1 grow nearly linearly as offset increases. This is because the seeker evaluates position starting at offset 0 (the current locked position), and the evaluated position shifts 1 bit for each fail. Thus, before reaching offset 65, the seeker needs to fail 65 evaluations for positions 0 to 64.

The linear behavior caused by the position updating rule can also be seen in the performance of other HSn variants. For example, consider the jagged line for HS2. HS2 has two seekers, one for the odd positions, the other for the even ones. Thus, if the number of bits dropped is a multiple of 2, then the same seeker previously locked at the old correct header position will be the one needed to find the new position, and thus lock time is linear in the number of bits removed. However, if the number of bits dropped is not a multiple of 2, the other seeker will need to find it. That seeker was NOT originally locked, and thus has been freely scanning. Therefore, the number of bits removed does not affect the time to lock, and the number of blocks lost is flat. Similar effects can be seen in each of the performance lines, where the HSn has the delay of every nth value exhibiting a linear growth in delay, and all other points are roughly flat.

Resource utilization was measured as the total number of LUTs and FFs utilized in a Xilinx Series 7 device. To compare the cost between competing recovery schemes, we measure the overall FF and LUT utilizations of an Aurora single RX lane from YARR with each recovery scheme. The target device for the hardware was the Xilinx Virtex 7 VC709, which is a candidate for the YARR DAQ hardware utilized by the LHC. Note that the Xilinx VC709 includes 433.2K LUTs and 866.4K FFs, meaning even our most resource-intensive design (HS33) consumes only 0.44% LUTs and 0.19% FFs on the chip [14].

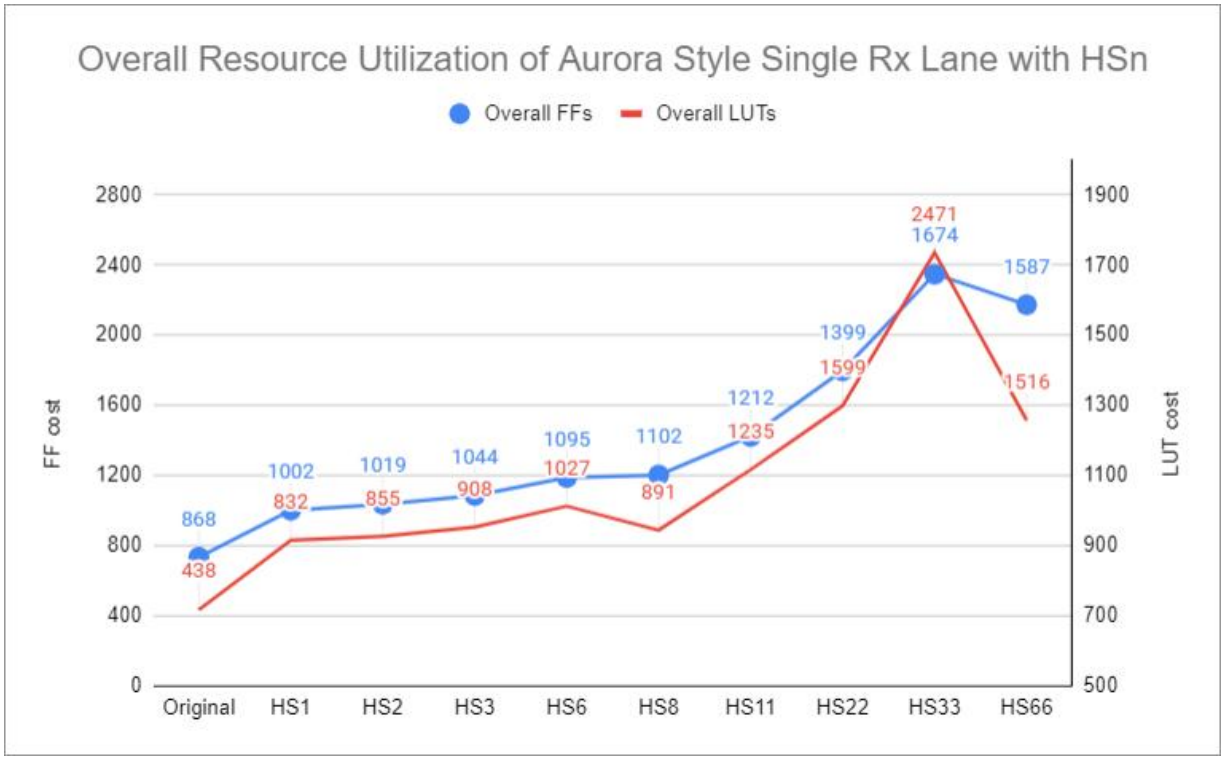

Figure 7: Overall resource utilization of Aurora style single rx lane with HSn variants and original scheme.

VII. CONCLUSION

In this paper we have presented a mechanism to perform rapid resynchronization of communications within the 64b/66b protocol. Existing mechanisms, when confronted with bit addition/deletion due to SEEs or other causes, can take a long time to resynchronize, and thus drop a significant amount of data. This is primarily due to a sequential search for the new synchronization point, and a conservative but slow testing of each potential alignment position. To address these issues, we contribute three new techniques: multiple head-seekers to search for a new alignment in parallel, high-speed searching by sniffing the incoming data, and a fail-fast design approach to rapidly assess each position. Compared with the original scheme, the HS1 scheme reduced the average data loss to 2.28% of the original scheme with 89.95% more LUTs and 15.44% more FFs. Although the best number of head-seekers depends on the relative importance of resource usage and expected data loss, a promising datapoint is the HS8 (parameter n = 8) system. This HS8 system provides an expected reduction in average data loss to 1.01% of original scheme, while only increasing the hardware resource costs of the transceiver by 103.42% LUTs and 26.96% FFs. This cost represents 0.23% LUTs and 0.13% FFs of the target FPGA.

REFERENCES

[1] A. X. Widmer and P. A. Franaszek, "A DC-balanced, partitioned-block, 8B/10B transmission code," *IBM Journal of Research and Development*, vol. 27, no. 5, pp. 440-451, Sept. 1983, doi: 10.1147/rd.275.0440.

[2] R. Walker and R. Dugan, "64b/66b low-overhead coding proposal for serial links," *IEEE 802.3 High Speed Study Group*, Jan. 2000.

[3] Xilinx. (2022, May. 11). *Aurora 8B/10B v11.1 LogiCORE IP Product Guide.* [Online]. Available: https://docs.xilinx.com/r/en-US/pg046-aurora-8b10b/Aurora-8B/10B-v11.1-LogiCORE-IP-Product-Guide

[4] Xilinx. (2022, Oct. 19). *Aurora 64B/66B v11.1 LogiCORE IP Product Guide*. [Online]. Available: https://docs.xilinx.com/r/en-US/pg074-aurora-64b66b/Aurora-64B/66B-v12.0-LogiCORE-IP-Product-Guide.

[5] D. White, "Considerations surrounding single event effects in FPGAs, ASICs, and processors," *Xilinx White Paper*, 2012. [Online]. Available: https://xilinx.eetrend.com/files-eetrend-xilinx/download/201110/2144-4013-wp402seeconsiderations.pdf

[6] J. Lalic *et al.,* "Single event effects on the RD53B pixel chip digital logic and on-chip CDR," *Journal of Instrumentation,* vol. 17, no.05, 2022, C05001.

[7] ATLAS collaboration. “Technical design report for the ATLAS inner tracker strip detector”, CERN, Rep. CERN-LHCC-2017-005; ATLAS-TDR-025, Apr. 2017. [Online]. Available: https://cds.cern.ch/record/2257755?ln=en.

[8] J. Lalic and M. Menouni, “SEE summary,” 2022, [Slide Deck]. Available: https://indico.cern.ch/event/1145919/contributions/4809744/attachments/2437591/4175048/TestSummary.pdf

[9] “High Luminosty LHC Project”, CERN. [Online]. Available: https://hilumilhc.web.cern.ch

[10] A. Martynyuk, “Rapid synchronization recovery from single event effects in the large hadron collider,” M.S. thesis, Dept. Electrical & Computer Eng., Univ. of Washington., Seattle, WA, USA, 2022.

[11] M. Menouni *et al.,* "Single event effects testing of the RD53B chip." *Journal of Physics: Conference Series*, vol. 2374, no. 1, p. 012084, IOP Publishing, 2022.

[12] M. Garcia-Sciveres, J. Christiansen, and F. Loddo. “RD53B manual,” CERN, Rep. CERN-RD53-PUB-19-002, 2019.

[13] LS. Kurilenko, “FPGA development of an emulator framework and a high speed I/O core for the ITk Pixel upgrade,” M.S. thesis, Dept. Electrical & Computer Eng., Univ. of Washington., Seattle, WA, USA, 2018. [Online]. Available: https://cds.cern.ch/record/2631488

[14] Xilinx, Sep. 8, 2020. “7 Series FPGAs Data Sheet: Overview (DS180),” Xilinx, [Online]. Available: https://docs.xilinx.com/v/u/en-US/ds180_7Series_Overview