Fenwick range-update-range-query on rings

Here's the problem: You have an array A, of elements of a ring S, on which you want to do efficient range updates and range queries (henceforth RURQ).

For our purposes, range updates are always of the form "Add some $s \in S$ to every element in the subarray $A[L, R]$, denoted $U(L, R, s)$" and range queries are always of the form "Return the total cumulative sum of the subarray $A[L, R]$."

For the case where the ring S is the integers $\mathbb{Z}$, this problem is most frequently solved with a lazy-propagation segment tree. However I've just found out that all you need is a fenwick tree.

A fenwick tree is an incredibly lightweight and simple data structure that does range updates and point queries (henceforth RUPQ) on arrays of integers in $\mathcal{O}(\log n)$ time. That is, the queries only ask for the values of individual array elements.

Fenwick trees rely on the "inclusion-exclusion" property, expressed in the formula below. This intuitive formula implies you only need to care about prefix sums, and is satisfied by any abelian group.

$$\sum_{i=L}^{R} A[i] = \left(\sum_{i=0}^{R} A[i]\right) - \left(\sum_{i=0}^{L-1} A[i]\right).$$

Hence, fenwick trees readily do RUPQ on any abelian group (and by extension, any ring.)

Fenwick trees are also capable of doing point updates and range queries (henceforth PURQ), simply by doing RUPQ on the finite difference array D, where $D[0] := A[0]$ and $D[i] := A[i] - A[i-1]$. That this works is actually a restatement of the fundamental theorem of discrete calculus, and I won't go into it.

Most commonly, the RURQ fenwick implementation comes about through ugly case analysis, and the fact that it works is unremarkable. There is, however, a very beautiful mathsy path by which to come to the solution. The following is by no means a proof, just the concept.

The idea.

For the time being, lets focus on a specific example, where $S = \mathbb{Z}$ and our array is $A = (0, 0, 0, 0, 0, 0, 0, 0)$. These assumptions don't have to be made, but the idea is much clearer through them.

Consider a range update of $U(L = 2, R = 5, g = 2)$.
The resulting array is now $A = (0, 0, 2, 2, 2, 2, 0, 0)$. If we want to do range queries on A, we should consider the cumulative/prefix sum array given by $\Sigma = (0, 0, 2, 4, 6, 8, 8, 8)$. The inclusion-exclusion property ensures that we only need two elements from Σ to evaluate a range query on A.

Σ has three regions of behaviour: initially it is level at 0, then it climbs in increments of 2 until it plateaus as a value of 8. This is best described by the function ξ defined below, where $\Sigma[i] = \xi(i)$.

$$\xi(x) = \begin{cases} 0 & x \in \{0, 1\} \\ 2x - 2 & x \in \{2, 3, 4, 5\} \\ 8 & x \in \{6, 7\} \end{cases}$$

Now consider the group $P_2(\mathbb{Z})$ of linear polynomials $ax + b$ where $a, b \in \mathbb{Z}$. This group is very naturally isomorphic to the additive group within $\mathbb{Z}^2$. Our new perspective will be that there is a one-to-one correspondence between elements of Σ in $\mathbb{Z}$, and elements of $\Xi := (0, 0, 2x - 2, 2x - 2, 2x - 2, 2x - 2, 8, 8)$ in $P_2(\mathbb{Z})$. This correspondence is given by $\Sigma[i] = \Xi[i](i)$, where $\Xi[i](x)$ denotes the evaluation of the function $\Xi[i]$ at $x \in \mathbb{Z}$.

The key idea is this: Ξ is clearly composed of two range updates:
$U(L = 2, R = 5, g = 2x - x)$ and $U(L = 6, R = 7, g = 8)$. In fact, any range update $U(L, R, g)$ in A will induce two range updates in Ξ, namely $U(L, R, gx - L)$ and $U(R + 1, N - 1, 0x + a(R - L + 1))$.

But the brilliant thing is that, even though range updates in A correspond to range updates in Ξ, *range queries* in A correspond to *point queries* in Ξ, because of the relation between Ξ and Σ.

So RURQ in $\mathbb{Z}$ relaxes to RUPQ in $P_2(\mathbb{Z})$!

Now to step back and generalise this.

For an array A of a ring S, range queries $Q(L, R)$ can be answered by taking $\Sigma[R] - \Sigma[L-1]$, where Σ is the prefix sum array of A in S. $\Sigma[i]$ may be obtained from $\Xi[i](i)$, where $\Xi[i]$ is an array in $P_2(S)$ (which is very naturally isomorphic to the additive group within S^2), and $\Xi[i](i)$ is the evaluation at $i \in \mathbb{Z}$ of the linear polynomial $\Xi[i]$ with coefficients in S. $\Xi[R]$ and $\Xi[L]$ are obtained as point queries on a RUPQ fenwick tree. When a range update $U(L, R, s)$ is applied to A, an induced pair of range updates $U(L, R, (s)x + (-L1_S))$ and $U(R+1, N-1, 0_S + (R-L+1)s)$ is applied to Ξ as a range update on a RUPQ fenwick tree.

Every update/query then takes $\mathcal{O}(m_S \log n)$, where $\mathcal{O}(m_S)$ is the time complexity of addition/multiplication in S.