Appendix A. The Mathematical Derivation of Structural Reparameterization Process

To clearly demonstrate the equivalence between the training-time dual-path architecture and its inference-time reparameterized structure, we provide a step-by-step mathematical derivation of structural reparameterization process.
For notational clarity, we define the following terms:

- Input feature map: $f \in \mathbb{R}^{C\times H\times W}$.
- Convolution kernel: $K^{(s)}_{c_o,c_i}$, where s denotes the kernel size, and C_i, C_o denote input and output channels, respectively.
- Convolution operator: $*$ denotes valid convolution; zero-padding, if needed, is explicitly written in the kernel.
- Element-wise multiplication: $\odot$.
- Sigmoid activation: $\sigma(\cdot)$.

A.1. Dual-Path Formula in Training Phase

The Transformer branch employs a window-based multi-head self-attention mechanism, which can be formally expressed as:

$$f_{\text{att}} = \text{W-MSA}(f) \in \mathbb{R}^{C\times H\times W} \tag{5}$$

In parallel, the convolutional branch applies a standard 3×3 convolution operation to the input feature map:

$$g = \text{Conv}_{3\times 3}(f) \in \mathbb{R}^{C\times H\times W} \tag{6}$$

The outputs of the two branches are fused through element-wise multiplication, followed by a 1×1 convolution:

$$f_{\text{out}} = \text{Conv}^{(1)}_{1\times 1}\Big(\text{Conv}_A(f_{\text{att}}) \odot \sigma\big(\text{Conv}_B(g)\big)\Big) \tag{7}$$

Both Conv_A and Conv_B consist solely of 1×1 convolutions.

A.2. Converting W-MSA's Linear Projection into Convolution

The linear projections used for (Q, K, V) in W-MSA can be viewed as 1×1 convolutions. This equivalence allows the W-MSA operation to be expressed as:

$$\text{W-MSA}(f) = K_{\text{msa}} * f, \quad K_{\text{msa}} \in \mathbb{R}^{C\times C\times 1\times 1}. \tag{8}$$

Therefore, the output can be written as:

$$f_{\text{att}} = K_{\text{msa}} * f \tag{9}$$

A.3. Merging Two 1×1 Convolutions into Single Convolution

For convenience, we define:

$$K_A = \mathrm{kernel}(\mathrm{Conv}_A), \quad K_B = \mathrm{kernel}(\mathrm{Conv}_B), \quad K_{3\times3} = \mathrm{kernel}(\mathrm{Conv}_{3\times3}) \tag{10}$$

These kernels have the following dimensions:

$$K_A, K_B \in \mathbb{R}^{C\times C\times1\times1}, \quad K_{3\times3} \in \mathbb{R}^{C\times C\times3\times3} \tag{11}$$

Substituting Eq. 9 into Eq. 7, we obtain:

$$\begin{aligned} f_{\mathrm{out}} &= \mathrm{Conv}^{(1)}_{1\times1}\Big((K_A * K_{\mathrm{msa}} * f) \odot \sigma\big(K_B * (K_{3\times3} * f)\big)\Big) \\ &= \mathrm{Conv}^{(1)}_{1\times1}\Big(\big[(K_A * K_{\mathrm{msa}}) * f\big] \odot \sigma\big[(K_B * K_{3\times3}) * f\big]\Big) \end{aligned}$$

A.4. Transforming Element-wise Multiplication into 1×1 Convolution

Element-wise multiplication $\odot$ is equivalent to a depthwise convolution with 1×1 kernel and group size of 1. The corresponding weight tensor is constructed as follows:

$$K^{(c,c)}_{\mathrm{mult}} = \big[(K_A * K_{\mathrm{msa}})\big]^{(c)} \cdot \sigma\big[(K_B * K_{3\times3})\big]^{(c)} \tag{12}$$

Consequently, the overall transformation in Eq. 12 can be expressed as a single convolution operation:

$$f_{\mathrm{out}} = K_{\mathrm{mult}} * f \tag{13}$$

where the fused kernel K_{mult} is obtained via:

$$K_{\mathrm{mult}} = \mathrm{Conv}^{(1)}_{1\times1}\Big((K_A * K_{\mathrm{msa}}) \cdot \sigma(K_B * K_{3\times3})\Big) \in \mathbb{R}^{C\times C\times1\times1} \tag{14}$$

A.5. Decomposing the 1×1 Convolution into Cascaded Convolutions

Although the kernel K_{mult} in Eq. 13 consists solely of 1×1 convolutions, hardware accelerators during inference often favor 3×3 convolutions to better leverage algorithmic optimizations such as Winograd or Fast Fourier Transform (FFT). By exploiting the additivity and homogeneity properties of convolution operations, the 1×1 convolution kernel can be equivalently decomposed into a sequence of nested convolutions:

$$K_{\mathrm{mult}} = K^{(1)}_1 \; * \; K^{(2)}_3 \; * \; K^{(3)}_3 \; * \; K^{(4)}_1 \tag{15}$$

where

- $K^{(1)}_1$ is 1×1 convolution that reduces input channels for dimensionality compression,
- $K^{(2)}_3$ and $K^{(3)}_3$ are 3×3 standard convolutions.
- $K^{(4)}_1$ is 1×1 convolution that expands channels back to dimension C.

These kernels are obtained via numerical decomposition (e.g., Singular Value Decomposition) after training and are loaded directly during inference without further decomposition. The inference process can thus be expressed as:

$$f_{\text{out}} = \mathrm{Conv}^{(4)}_{1\times1}\Big(\mathrm{Conv}^{(3)}_{3\times3}\big(\mathrm{Conv}^{(2)}_{3\times3}(\mathrm{Conv}^{(1)}_{1\times1}(f))\big)\Big) \tag{16}$$

The transformations from Eq. 12 to Eq. 13 and finally to Eq. 16 are strictly equivalent at each step. This demonstrates the functional equivalence between the original dual-branch structure and the reparameterized convolutional sequence. Furthermore, the use of nested convolutions reduces computational complexity from $O(N^2)$ to $O(N)$.

Appendix B. Additional Ablation Study Results and Model Complexity Breakdown

Moreover, we provide BD-MS-SSIM and BD-Rate comparison on both datasets in Table 3. Superior compression performance is indicated by positive BD-MS-SSIM and negative BD-Rate, reflecting higher quality preservation and improved coding efficiency, respectively. To evaluate the lightweight nature of the proposed framework and demonstrate the necessity of structural reparameterization, a component-level complexity breakdown is presented in Table 4.

Table 3: BD-Rate and BD-MS-SSIM comparison of architectural variants on InStereo2K and Cityscapes dataset.

Architectural Variants	InStereo2K		Cityscapes	
	BD-Rate	BD-MS-SSIM	BD-Rate	BD-MS-SSIM
Backbone	0	0	0	0
Backbone + RSB (Reparam)	-4.60	0.13 (+0.26%)	-9.40	0.08 (+0.32%)
Backbone + RSB (Dual-Path)	-9.41	0.27 (+0.85%)	-14.35	0.64 (+1.16%)
Backbone + CFEMs	-12.69	0.33 (+1.73%)	-26.96	0.87 (+1.45%)
RSTSIC	**-14.55**	**0.56 (+2.58%)**	**-34.48**	**1.40 (+1.60%)**

Table 4: Component-level complexity breakdown of the proposed framework.

Component	Parameters (M)	FLOPs (G)
RSB	0.20	25.15
CFEM	0.49	2.11
Joint Decoder	1.60	33.81
Independent Encoder	0.42	1.11
RSTSIC	2.73	40.79