	Block	Operation	Input Shape (H×W×C)	Output Shape (H×W×C) (XXS / XS / S)	Hardware Mapping
Overall	Input	Image Input	–	256×256×3	DMA load → buffer
	Stem	Conv 3×3, stride=2	256×256×3	128×128×16	Conv kernel
	Stage 1	MV2 Block ×1	128×128×16	128×128×(16 / 32 / 32)	Uses **MV2** master table
	Stage 2	MV2 Block, stride=2	128×128×(16 / 32 / 32)	64×64×(24 / 48 / 64)	Uses **MV2** master table
	Stage 2	MV2 Block ×2	64×64×(24 / 48 / 64)	64×64×(24 / 48 / 64)	Uses **MV2** master table
	Stage 3	MV2 Block, stride=2	64×64×(24 / 48 / 64)	32×32×(48 / 64 / 96)	Uses **MV2** master table
	Stage 3	MobileViT Block (L=2)	32×32×(48 / 64 / 96)	32×32×(48 / 64 / 96) (d=64/96/144)	Uses **MVB** master table
	Stage 4	MV2 Block, stride=2	32×32×(48 / 64 / 96)	16×16×(64 / 80 / 128)	Uses **MV2** master table
	Stage 4	MobileViT Block (L=4)	16×16×(64 / 80 / 128)	16×16×(64 / 80 / 128) (d=80/120/192)	Uses **MVB** master table
	Stage 5	MV2 Block, stride=2	16×16×(64 / 80 / 128)	8×8×(80 / 96 / 160)	Uses **MV2** master table
	Stage 5	MobileViT Block (L=3)	8×8×(80 / 96 / 160)	8×8×(80 / 96 / 160) (d=96/144/240)	Uses **MVB** master table
	Head	Conv 1×1	8×8×(80 / 96 / 160)	8×8×(320 / 384 / 640)	Conv kernel
	Head	Global Pool	8×8×(320 / 384 / 640)	1×1×(320 / 384 / 640)	Pool unit
	Classifier	Linear	1×1×(320 / 384 / 640)	1×1×1000	Matmul kernel

	Step	Operation	Input → Output	Output	Hardware Mapping
MV2	1	Expansion Conv 1×1	C	4C (XXS: 2C)	Conv kernel
	2	Batch Normalization	4C	4C	Batch Normalization Unit
	3	Nonlinearity	4C	4C	Activation unit (ReLU6 / Swish)
	4	Depthwise Conv 3×3	4C	4C (stride=1 or 2)	Conv kernel (channel-wise mode)
	5	Batch Normalization	4C	4C	Batch Normalization Unit
	6	Nonlinearity	4C	4C	Activation unit (ReLU6 / Swish)
	7	Projection Conv 1×1	4C	Cout	Conv kernel
	8	Batch Normalization	Cout	Cout	Batch Normalization Unit
	9	Nonlinearity	Cout → Cout	Cout	Activation unit (ReLU6 / Swish)
	10 (optional)	Residual Add	If stride=1 & same dims	Add unit	Add unit

	Step	Operation	Input	Output	Hardware Mapping
MVB	1	Local Conv 3×3	H × W ×C	H × W ×C	Conv kernel
	2	Pointwise Conv 1×1	H × W ×C	H × W × d	Conv kernel
	3	Unfold (Patchify)	H × W × d	Patches × d	DMA + buffer rearrange
	4	Transformer Layer ×L	Patches × d	Patches × d	See **Transformer** Table for hardware mapping
	5	Fold (Reconstruct)	Patches × d	H × W × d	DMA + buffer rearrange
	6	Pointwise Conv 1×1	H × W × d	H × W ×C	Conv kernel
	7	Concatenation	H× W ×C	H × W × 2C	Concat unit
	8	Conv 3x3	H × W × 2C	H × W × C	Conv kernal

	Step	Operation	Input Shape	Output Shape	Hardware Mapping / Accelerator View
Transformer	1	LayerNorm (LN1)	Tokens × d	Tokens × d	Element-wise ops: mean/variance reduction (adder tree + divider) + scale/shift (mult/add)
	2	Linear (Q projection)	Tokens × d	Tokens × d_k	Matmul kernel (weights = [d × d_k])
	3	Linear (K projection)	Tokens × d	Tokens × d_k	Matmul kernel
	4	Linear (V projection)	Tokens × d	Tokens × d_v	Matmul kernel
	5	Q × K^T (Attention scores)	Tokens × d_k **X** d_k × Tokens	Tokens × Tokens	Matmul kernel (systolic array), heavy compute O(L^2· d_k)
	6	Softmax	Tokens × Tokens	Tokens × Tokens	Exp LUT/polynomial + row reduction + reciprocal multiply
	7	A × V (apply attention)	Tokens × Tokens	Tokens × d_v	Matmul kernel
	8	Linear (output proj)	Tokens × d_v	Tokens × d	Matmul kernel
	9	Residual Add (skip)	Tokens × d	Tokens × d	Elementwise add unit
	10	LayerNorm (LN2)	Tokens × d	Tokens × d	Same as LN1
	11	Linear (FFN expansion)	Tokens × d	Tokens × 4d	Matmul kernel (larger width)
	12	Activation (GELU/ReLU)	Tokens × 4d	Tokens × 4d	Nonlinear unit (approx poly/LUT)
	13	Linear (FFN projection)	Tokens × 4d	Tokens × d	Matmul kernel
	14	Residual Add (skip)	Tokens × d	Tokens × d	Elementwise add unit