rocSPARSELt Data Types

Data Structures

`rocsparselt_handle`

The structure holds the rocSPARSELt library context (device properties, system information, etc.).
The handle must be initialized and destroyed
with rocsparselt_init() and rocsparselt_destroy() functions respectively.

`rocsparselt_mat_descr`

The structure captures the shape and characteristics of a matrix.
It is initialized with rocsparselt_dense_descr_init() or rocsparselt_structured_descr_init() functions and destroyed with rocsparselt_mat_descr_destroy().

`rocsparselt_matmul_descr`

The structure holds the description of the matrix multiplication operation.
It is initialized with rocsparselt_matmul_descr_init() function.

`rocsparserlt_matmul_alge_selection`

The structure holds the description of the matrix multiplication algorithm.
It is initialized with rocsparselt_matmul_alg_selection_init() function.

`rocsparselt_matmul_plan`

The structure holds the matrix multiplication execution plan, namely all the information necessary to execute the `rocsparselt_matmul()` operation.
It is initialized and destroyed
with rocsparselt_matmul_plan_init() and rocsparselt_matmul_plan_destroy() functions respectively.

Enumerators

`rocsparselt_sparsity`

The enumerator specifies the sparsity ratio of the structured matrix as

$$sparisty\ ratio = \frac{nnz}{num\_rows * num\_cols}$$

Value	Description
`rocsparselt_sparsity_50_percent`	50% Sparsity Ratio: - **2:4** for `half`, `bfloat16`, `int`

The sparsity property is used in the rocsparselt_structured_descr_init() function.

`rocsparselt_compute_type`

The enumerator specifies the compute precision modes of the matrix

Value	Description
`rocsparselt_compute_f32`	- Default mode for 32-bit floating-point precision - All computations and intermediate storage ensure at least 32-bit precision - Matrix Core will be used whenever possible
`rocsparselt_compute_i32`	- Default mode for 32-bit integer precision - All computations and intermediate storage ensure at least 32-bit integer precision - Matrix Core will be used whenever possible

The compute precision is used in the rocsparselt_matmul_descr_init() function.

`rocsparselt_mat_descr_attribute`

The enumerator specifies the additional attributes of a matrix descriptor

Value	Description
`rocsparselt_mat_num_batches`	Number of matrices in a batch (`int` data type)
`rocsparselt_mat_batch_stride`	Stride between consecutive matrices in a batch expressed in terms of matrix elements (`int64_t` data type)

The algorithm enumerator is used in the rocsparselt_mat_descr_set_attribute() and rocsparselt_mat_descr_get_attribute() functions.

`rocsparselt_matmul_descr_attribute`

The enumerator specifies the additional attributes of a matrix multiplication descriptor

Value	Type	Default Value	Description
`rocsparselt_matmul_activation_relu`	`int` 0: **false**, **true** otherwise	`false`	ReLU activation function
`rocsparselt_matmul_activation_relu_upperbound`	`float`	`inf`	Upper bound of the ReLU activation function
`rocsparselt_matmul_activation_relu_threshold`	`float`	`0.0f`	Lower threshold of the ReLU activation function
`rocsparselt_matmul_activation_gelu`	`int` 0: **false**, **true** otherwise	`false`	GeLU activation function

where the *ReLU* activation function is defined as:

$$\text{ReLU(v)} = \begin{cases} v > threshold, & min(v, upperbound) \\ v \leq threshold, & 0 \end{cases}$$

The algorithm enumerator is used in the rocsparselt_matmul_descr_set_attribute() and rocsparselt_matmul_descr_get_attribute() functions.

`rocsparselt_matmul_alg`

The enumerator specifies the algorithm for matrix-matrix multiplication

Value	Description
`rocsparselt_matmul_alg_default`	Default algorithm

The algorithm enumerator is used in the rocsparselt_matmul_alg_selection_init() function.

`rocsparselt_matmul_alg_attribute`

The enumerator specifies the matrix multiplication algorithm attributes

Value	Description
`rocsparselt_matmul_alg_config_id`	Algorithm ID (set and query)
`rocsparselt_matmul_alg_config_max_id`	Algorithm ID limit (query only)
`rocsparselt_matmul_search_iterations`	Number of iterations (kernel launches per algorithm) for rocsparselt_matmul_search(), default=10

The algorithm attribute enumerator is used in the rocsparselt_matmul_alg_get_attribute() and rocsparselt_matmul_alg_set_attribute() functions.

`rocsparselt_prune_alg`

The enumerator specifies the pruning algorithm to apply to the structured matrix before the compression

Value	Description
`rocsparselt_prune_smfmac_strip`	- `half`, `bfloat16`, `int8`, `float8`, `bfloat8`: Zero-out two values in a 1x4 strip to maximize the *L1-norm* of the resulting strip The strip direction is chosen according to the operation `op` and matrix layout applied to the structured (sparse) matrix

The pruning algorithm is used in the rocsparselt_smfmac_prune() function.

rocsparselt Functions

Library Management Functions

`rocsparselt_init`

```
rocsparselt_status
rocsparselt_init(rocsparselt_handle* handle)
```

The function initializes the rocsparselt library handle (`rocsparselt_handle`) which holds the rocsparselt library context. It allocates light hardware resources on the host, and must be called prior to making any other rocsparselt library calls. Calling any rocsparselt function which uses `rocsparselt_handle` without a previous call of `rocsparselt_init()` will return an error.

The rocsparselt library context is tied to the current ROCm device. To use the library on multiple devices, one rocsparselt handle should be created for each device.

Parameter	Memory	In/Out	Description
`handle`	Host	OUT	rocsparselt library handle

See rocsparselt_status for the description of the return status.

`rocsparselt_destroy`

```
rocsparselt_status
rocsparselt_destroy(const rocsparselt_handle* handle)
```

The function releases hardware resources used by the rocsparselt library. This function is the last call with a particular handle to the rocsparselt library.

Calling any rocsparselt function which uses `rocsparselt_handle` after `rocsparselt_destroy()` will return an error.

Parameter	Memory	In/Out	Description
`handle`	Host	IN	rocsparselt library handle

See rocsparselt_status for the description of the return status.

Matrix Descriptor Functions

`rocsparselt_dense_descr_init`

```
rocsparselt_status
rocsparselt_dense_descr_init(const rocsparselt_handle* handle,
                             rocsparselt_mat_descr*    matDescr,
                             int64_t                   rows,
                             int64_t                   cols,
                             int64_t                   ld,
                             uint32_t                  alignment,
                             rocsparselt_datatype      valueType,
                             rocsparselt_order         order);
```

The function initializes the descriptor of a *dense* matrix.

Parameter	Memory	In/Out	Description	Possible Values
`handle`	Host	IN	rocsparselt library handle	
`matDescr`	Host	OUT	Dense matrix description	
`rows`	Host	IN	Number of rows	
`cols`	Host	IN	Number of columns	
`ld`	Host	IN	Leading dimension	≥ rows if column-major, ≥ cols if row-major
`alignment`	Host	IN	Memory alignment in bytes	Multiple of 16 (not-used)
`valueType`	Host	IN	Data type of the matrix	`rocsparselt_datatype_f16_r`, `rocsparselt_datatype_bf16_r`, `rocsparselt_datatype_i8_r`, `rocsparselt_datatype_f8_r`, `rocsparselt_datatype_bf8_r`
`order`	Host	IN	Memory layout	`rocsparselt_order_column`, `rocsparselt_order_row (not supported)`

Constrains:

- `row`, `col` must ≥ 8
- For matrix B = K x N, `K` must be a multiple of 8

See rocsparselt_status for the description of the return status.

`rocsparselt_structured_descr_init`

```
rocsparselt_status
rocsparselt_structured_descr_init(const rocsparselt_handle* handle,
                                  rocsparselt_mat_descr*    matDescr,
                                  int64_t                   rows,
                                  int64_t                   cols,
                                  int64_t                   ld,
                                  uint32_t                  alignment,
                                  rocsparselt_datatype      valueType,
                                  rocsparselt_order         order,
                                  rocsparselt_sparsity_     sparsity)
```

The function initializes the descriptor of a *structured* matrix.

Parameter	Memory	In/Out	Description	Possible Values
`handle`	Host	IN	rocsparselt library handle	
`matDescr`	Host	OUT	Dense matrix description	
`rows`	Host	IN	Number of rows	
`cols`	Host	IN	Number of columns	
`ld`	Host	IN	Leading dimension	≥ rows if column-major, ≥ cols if row-major
`alignment`	Host	IN	Memory alignment in bytes	Multiple of 16 (not used)
`valueType`	Host	IN	Data type of the matrix	`rocsparselt_datatype_f16_r`, `rocsparselt_datatype_bf16_r`, `rocsparselt_datatype_i8_r`,

Parameter	Memory	In/Out	Description	Possible Values
				`rocsparselt_datatype_f8_r`, `rocsparselt_datatype_bf8_r`
`order`	Host	IN	Memory layout	`rocsparselt_order_column`, `rocsparselt_order_row (not supported)`
`sparsity`	Host	IN	Matrix sparsity ratio	`rocsparselt_sparsity_50_percent`

Constrains:

- `row`, `col` must ≥ 8
- For `op = rocsparselt_operation_non`
  - `col` must be the multiplication of 8
- For `op = rocsparselt_operation_transpose`
  - `row` must be the multiplication of 8

See rocsparselt_status for the description of the return status.

`rocsparselt_mat_descr_destroy`

```
rocsparselt_status
rocsparselt_mat_descr_destroy(const rocsparselt_mat_descr* matDescr)
```

The function releases the resources used by an instance of a matrix descriptor. After this call, the matrix descriptor and the matmul descriptor can no longer be used.

Parameter	Memory	In/Out	Description
`matDescr`	Host	IN	Matrix descriptor

See rocsparselt_status for the description of the return status.

---

rocsparselt_mat_descr_set_attribute

```
rocsparselt_status
rocsparselt_mat_descr_set_attribute(const rocsparselt_handle*        handle,
                                    rocsparselt_mat_descr*           matDescr,
                                    rocsparselt_mat_descr_attribute* matAttribute,
                                    const void*                      data,
                                    size_t                           dataSize)
```

The function sets the value of the specified attribute belonging to matrix descriptor such as number of batches and their stride.

Parameter	Memory	In/Out	Description	Possible Values
handle	Host	IN	rocsparselt library handle	
matmulDescr	Host	OUT	Matrix descriptor	
matAttribute	Host	IN	Attribute to set	rocsparselt_mat_num_batches, rocsparselt_mat_batch_stride
data	Host	IN	Pointer to the value to which the specified attribute will be set	
dataSize	Host	IN	Size in bytes of the attribute value used for verification	

See rocsparselt_status for the description of the return status.

rocsparselt_mat_descr_get_attribute

```
rocsparselt_status
rocsparselt_mat_descr_get_attribute(const rocsparselt_handle        handle,
                                    const rocsparselt_mat_descr     matmulDescr,
                                    rocsparselt_mat_descr_attribute matAttribute,
                                    void*                           data,
                                    size_t                          dataSize)
```

The function gets the value of the specified attribute belonging to matrix descriptor such as number of batches and their stride.

Parameter	Memory	In/Out	Description	Possible Values
`handle`	Host	IN	rocsparselt library handle	
`matmulDescr`	Host	IN	Matrix descriptor	
`matAttribute`	Host	IN	Attribute to retrieve	`rocsparselt_mat_num_batches`, `rocsparselt_mat_batch_stride`
`data`	Host	OUT	Memory address containing the attribute value retrieved by this function	
`dataSize`	Host	IN	Size in bytes of the attribute value used for verification	

See rocsparselt_status for the description of the return status.

Matmul Descriptor Functions

`rocsparselt_matmul_descr_init`

```
rocsparselt_status
rocsparselt_matmul_descr_init(const rocsparselt_handle*         handle,
                              rocsparselt_matmul_descr*         matmulDescr,
                              rocsparselt_operation             opA,
                              rocsparselt_operation             opB,
                              const rocsparselt_mat_descr*      matA,
                              const rocsparselt_mat_descr*      matB,
                              const rocsparselt_mat_descr*      matC,
                              const rocsparselt_mat_descr*      matD,
                              rocsparselt_compute_type          computeType)
```

The function initializes the *matrix multiplication* descriptor.

Parameter	Memory	In/Out	Description	Possible Values
`handle`	Host	IN	rocsparselt library handle	
`matmulDescr`	Host	OUT	Matrix multiplication descriptor	
`opA`	Host	IN	Operation applied to the matrix `A`	`rocsparselt_operation_non`, `rocsparselt_operation_transpose`
`opB`	Host	IN	Operation applied to the matrix `B`	`rocsparselt_operation_non`, `rocsparselt_operation_transpose`
`matA`	Host	IN	Structured matrix descriptor `A`	
`matB`	Host	IN	Dense matrix	

Parameter	Memory	In/Out	Description	Possible Values
			descriptor `B`	
`matC`	Host	IN	Dense matrix descriptor `C`	
`matD`	Host	IN	Dense matrix descriptor `D`	
`computeType`	Host	IN	Compute precision	`rocsparselt_compute_f32`, `rocsparselt_compute_32i`

The structured matrix descriptor can used for `matA` or `matB` but not both.

Data types Supported:

Input	Output	Compute
`rocsparselt_datatype_f16_r`	`rocsparselt_datatype_f16_r`	`rocsparselt_compute_f32`
`rocsparselt_datatype_f16_r`	`rocsparselt_datatype_f16_r`	`rocsparselt_compute_f32`
`rocsparselt_datatype_i8_r`	`rocsparselt_datatype_i8_r`	`rocsparselt_compute_i32`
`rocsparselt_datatype_f8_r`	`rocsparselt_datatype_f8_r`	`rocsparselt_compute_f32`
`rocsparselt_datatype_bf8_r`	`rocsparselt_datatype_bf8_r`	`rocsparselt_compute_f32`

See rocsparselt_status for the description of the return status.

---

rocsparselt_matmul_descr_set_attribute

```
rocsparselt_status
rocsparselt_matmul_descr_set_attribute(const rocsparselt_handle*           handle,
                                       rocsparselt_matmul_descr*           matmulDescr,
                                       rocsparselt_matmul_descr_attribute  matmulAttribute,
                                       const void*                         data,
                                       size_t                              dataSize)
```

The function sets the value of the specified attribute belonging to matrix descriptor such as activation function and bias.

Parameter	Memory	In/Out	Description	
`handle`	Host	IN	rocsparselt library handle	
`matmulDescr`	Host	OUT	Matrix descriptor	
`matmulAttribute`	Host	IN	Attribute to set	`rocsparselt_matmul_activation_relu`, `rocsparselt_matmul_activation_relu_upperbound`, `rocsparselt_matmul_activation_relu_threshold`, `rocsparselt_matmul_activation_gelu`, `rocsparselt_matmul_bias_pointer`, `rocsparselt_matmul_bias_stride`
`data`	Host	IN	Pointer to the value to which the specified attribute will be set	
`dataSize`	Host	IN	Size in bytes of the attribute value used for verification	

See rocsparselt_status for the description of the return status.

rocsparselt_matmul_descr_get_attribute

```
rocsparselt_status
rocsparselt_matmul_descr_get_attribute(const rocsparselt_handle*           handle,
                                       const rocsparselt_matmul_descr*     matmulDescr,
                                       rocsparselt_matmul_descr_attribute matmulAttribute,
                                       void*                               data,
                                       size_t                              dataSize)
```

The function gets the value of the specified attribute belonging to matrix descriptor such as activation function and bias.

Parameter	Memory	In/Out	Description	
handle	Host	IN	rocsparselt library handle	
matmulDescr	Host	IN	Matrix descriptor	
matmulAttribute	Host	IN	Attribute to retrieve	rocsparselt_matmul_activation_relu, rocsparselt_matmul_activation_relu_upperbound, rocsparselt_matmul_activation_relu_threshold, rocsparselt_matmul_activation_gelu, rocsparselt_matmul_bias_pointer, rocsparselt_matmul_bias_stride
data	Host	OUT	Memory address containing the attribute value retrieved by this function	
dataSize	Host	IN	Size in bytes of the attribute value used for verification	

See rocsparselt_status for the description of the return status.

Matmul Algorithm Functions

`rocsparselt_matmul_alg_selection_init`

```
rocsparselt_status
rocsparselt_matmul_alg_selection_init(const rocsparselt_handle*           handle,
                                      rocsparselt_matmul_alg_selection*   algSelection,
                                      const rocsparselt_matmul_descr*     matmulDescr,
                                      rocsparselt_matmul_alg              alg)
```

The function initializes the *algorithm selection* descriptor.

Parameter	Memory	In/Out	Description	Possible Values
`handle`	Host	IN	rocsparselt library handle	
`algSelection`	Host	OUT	Algorithm selection descriptor	
`matmulDescr`	Host	IN	Matrix multiplication descriptor	
`alg`	Host	IN	Algorithm mode	`rocsparselt_matmul_alg_default`

See rocsparselt_status for the description of the return status.

---

rocsparselt_matmul_alg_set_attribute

```
rocsparselt_status
rocsparselt_matmul_alg_set_attribute(const rocsparselt_handle*         handle,
                                     rocsparselt_matmul_alg_selection* algSelection,
                                     rocsparselt_matmul_alg_attribute  attribute,
                                     const void*                       data,
                                     size_t                            dataSize)
```

The function sets the value of the specified attribute belonging to algorithm selection descriptor.

Parameter	Memory	In/Out	Description	Possible Values
handle	Host	IN	rocsparselt library handle	
algSelection	Host	OUT	Algorithm selection descriptor	
attribute	Host	IN	The attribute to set	rocsparselt_matmul_alg_config_id, rocsparselt_matmul_alg_config_max_id, rocsparselt_matmul_search_iterations
data	Host	IN	Pointer to the value to which the specified attribute will be set	
dataSize	Host	IN	Size in bytes of the attribute value used for verification	

See rocsparselt_status for the description of the return status.

---

rocsparselt_matmul_alg_get_attribute

```
rocsparselt_status
rocsparselt_matmul_alg_get_attribute(const rocsparselt_handle*               handle,
                                     const rocsparselt_matmul_alg_selection* algSelection,
                                     rocsparselt_matmul_alg_attribute        attribute,
                                     void*                                   data,
                                     size_t                                  dataSize)
```

The function returns the value of the queried attribute belonging to algorithm selection descriptor.

Parameter	Memory	In/Out	Description	Possible Values
`handle`	Host	IN	rocsparselt library handle	
`algSelection`	Host	IN	Algorithm selection descriptor	
`attribute`	Host	IN	The attribute that will be retrieved by this function	`rocsparselt_matmul_alg_config_id`, `rocsparselt_matmul_alg_config_max_id`, `rocsparselt_matmul_search_iterations`
`data`	Host	OUT	Memory address containing the attribute value retrieved by this function	
`dataSize`	Host	IN	Size in bytes of the attribute value used for verification	

See rocsparselt_status for the description of the return status.

Matmul Functions

rocsparselt_matmul_get_workspace

```
rocsparselt_status
rocsparselt_matmul_get_workspace(const rocsparselt_handle*              handle,
                                 const rocsparselt_matmul_alg_selection* algSelection,
                                 size_t*                                 workspaceSize)
```

The function determines the required workspace size associated to the selected algorithm.

Parameter	Memory	In/Out	Description
handle	Host	IN	rocsparselt library handle
algSelection	Host	IN	Algorithm selection descriptor
workspaceSize	Host	OUT	Workspace size in bytes

See rocsparselt_status for the description of the return status.

---

rocsparselt_matmul_plan_init

```
rocsparselt_status
rocsparselt_matmul_plan_init(const rocsparselt_handle*              handle,
                             rocsparselt_matmul_plan*               plan,
                             const rocsparselt_matmul_descr*        matmulDescr,
                             const rocsparselt_matmul_alg_selection* algSelection,
                             size_t                                 workspaceSize)
```

Parameter	Memory	In/Out	Description
handle	Host	IN	rocsparselt library handle
plan	Host	OUT	Matrix multiplication plan
matmulDescr	Host	IN	Matrix multiplication descriptor
algSelection	Host	IN	Algorithm selection descriptor
workspaceSize	Host	IN	Workspace size in bytes

See rocsparselt_status for the description of the return status.

---

rocsparselt_matmul_plan_destroy

```
rocsparselt_status
rocsparselt_matmul_plan_destroy(const rocsparselt_matmul_plan* plan)
```

The function releases the resources used by an instance of the matrix multiplication plan. This function is the last call with a specific plan instance.

Calling any rocsparselt function which uses `rocsparselt_matmul_plan` after `rocsparselt_matmul_plan_destroy()` will return an error.

Parameter	Memory	In/Out	Description
`plan`	Host	IN	Matrix multiplication plan

See rocsparselt_status for the description of the return status.

---

rocsparselt_matmul

```
rocsparselt_status
rocsparselt_matmul(const rocsparselt_handle*           handle,
                   const rocsparselt_matmul_plan*      plan,
                   const void*                         alpha,
                   const void*                         d_A,
                   const void*                         d_B,
                   const void*                         beta,
                   const void*                         d_C,
                   void*                               d_D,
                   void*                               workspace,
                   hipStream_t*                        streams,
                   int32_t                             numStreams)
```

The function computes the matrix multiplication of matrices `A` and `B` to produce the output matrix `D`, according to the following operation:

$$D = \text{Activation}(\alpha op(A) * op(B) + \beta C + \text{bias})$$

where `A`, `B`, and `C` are input matrices, and α and β are input scalars.

Note: The function currently only supports the case where `D` has the same shape of `C`

Parameter	Memory	In/Out	Description
`handle`	Host	IN	rocsparselt library handle
`plan`	Host	IN	Matrix multiplication plan

Parameter	Memory	In/Out	Description
`alpha`	Host	IN	α scalar used for multiplication (`float` data type)
`d_A`	Device	IN	Pointer to the structured matrix `A`
`d_B`	Device	IN	Pointer to the dense matrix `B`
`beta`	Host	IN	β scalar used for multiplication (`float` data type)
`d_C`	Device	OUT	Pointer to the dense matrix `C`
`d_D`	Device	OUT	Pointer to the dense matrix `D`
`workspace`	Device	IN	Pointer to workspace
`streams`	Host	IN	Pointer to HIP stream array for the computation
`numStreams`	Host	IN	Number of HIP streams in `streams`

Data types Supported:

Input	Output	Compute
`rocsparselt_datatype_f16_r`	`rocsparselt_datatype_f16_r`	`rocsparselt_compute_f32`
`rocsparselt_datatype_f16_r`	`rocsparselt_datatype_f16_r`	`rocsparselt_compute_f32`
`rocsparselt_datatype_i8_r`	`rocsparselt_datatype_i8_r`	`rocsparselt_compute_i32`
`rocsparselt_datatype_f8_r`	`rocsparselt_datatype_f8_r`	`rocsparselt_compute_f32`
`rocsparselt_datatype_bf8_r`	`rocsparselt_datatype_bf8_r`	`rocsparselt_compute_f32`

The *structured matrix* `A` (before the compression) must respect the following constrains depending on the operation applied on it:

- For `op = rocsparselt_operation_non`
  - each row must have at least two zero values every four elements
- For `op = rocsparselt_operation_transpose`
  - each column must have at least two zero values every four elements

The correctness of the pruning result (matrix `A`) can be check with the function rocsparselt_smfmac_prune_check().

Properties

- The routine requires no extra storage
- The routine supports asynchronous execution with respect to `streams[0]`

See rocsparselt_status for the description of the return status.

`rocsparselt_matmul_search`

```
rocsparselt_status
rocsparselt_matmul_search(const rocsparselt_handle*       handle,
                          const rocsparselt_matmul_plan*  plan,
                          const void*                     alpha,
                          const void*                     d_A,
                          const void*                     d_B,
                          const void*                     beta,
                          const void*                     d_C,
                          void*                           d_D,
                          void*                           workspace,
                          hipStream_t*                    streams,
                          int32_t                         numStreams)
```

The function evaluates all available algorithms for the matrix multiplication and automatically updates the `plan` by selecting the fastest one. The functionality is intended to be used for auto-tuning purposes when the same operation is repeated multiple times over different inputs.

The function behavior is the same of rocsparselt_matmul().

- The function is *NOT* asynchronous with respect to `streams[0]` (*blocking call*)
- The number of iterations for the evaluation can be set by using rocsparselt_matmul_alg_set_attribute() with `rocsparselt_matmul_search_iterations`.
- The selected algorithm id can be retrieved by using rocsparselt_matmul_alg_get_attribute() with `rocsparselt_matmul_alg_config_id`.

Helper Functions

`rocsparselt_smfmac_prune`

```
rocsparselt_status
rocsparselt_smfmac_prune(const rocsparselt_handl*e       handle,
                         const rocsparselt_matmul_descr* matmulDescr,
                         const void*                     d_in,
                         void*                           d_out,
                         rocsparselt_prune_alg           pruneAlg,
                         hipStream_t                     stream)
```

The function prunes a dense matrix `d_in` according to the specified algorithm `pruneAlg`.

Parameter	Memory	In/ Out	Description	Possible Values
`handle`	Host	IN	rocsparselt library handle	
`matmulDescr`	Host	IN	Matrix multiplication descriptor	
`d_in`	Device	IN	Pointer to the dense matrix	
`d_out`	Device	OU T	Pointer to the pruned matrix	
`pruneAlg`	Device	IN	Pruning algorithm	`rocsparselt_prune_smfmac_tile`, `rocsparselt_prune_smfmac_strip`
`stream`	Host	IN	HIP stream for the computation	

Properties

- The routine requires no extra storage
- The routine supports asynchronous execution with respect to `stream`

See rocsparselt_status for the description of the return status.

rocsparselt_smfmac_prune_check

```
rocsparselt_status
rocsparselt_smfmac_prune_check(const rocsparselt_handle*       handle,
                               const rocsparselt_matmul_descr* matmulDescr,
                               const void*                     d_in,
                               int*                            d_valid,
                               hipStream_t                     stream)
```

The function checks the correctness of the pruning structure for a given matrix.

Parameter	Memory	In/Out	Description
`handle`	Host	IN	rocsparselt library handle
`matmulDescr`	Host	IN	Matrix multiplication descriptor
`d_in`	Device	IN	Pointer to the matrix to check
`d_valid`	Device	OUT	Validation results (`0` correct, `1` wrong)
`stream`	Host	IN	HIP stream for the computation

See rocsparselt_status for the description of the return status.

---

rocsparselt_smfmac_compressed_size

```
rocsparselt_status
rocsparselt_smfmac_compressed_size(const rocsparselt_handle*      handle,
                                   const rocsparselt_matmul_plan* plan,
                                   size_t*                        compressedSize)
```

The function provides the size of the *compressed* matrix to be allocated before calling rocsparselt_smfmac_compress().

Parameter	Memory	In/Out	Description
`handle`	Host	IN	rocsparselt library handle
`plan`	Host	IN	Matrix plan descriptor
`compressedSize`	Host	OUT	Size in bytes of the compressed matrix

See rocsparselt_status for the description of the return status.

---

rocsparselt_smfmac_compress

```
rocsparselt_status
rocsparselt_smfmac_compress(const rocsparselt_handle*      handle,
                            const rocsparselt_matmul_plan* plan,
                            const void*                    d_dense,
                            void*                          d_compressed,
                            hipStream_t                    stream)
```

The function compresses a dense matrix `d_dense`. The *compressed* matrix is intended to be used as the first operand `A` in the rocsparselt_matmul() function.

Parameter	Memory	In/Out	Description
`handle`	Host	IN	rocsparselt library handle
`plan`	Host	IN	Matrix multiplication plan
`d_dense`	Device	IN	Pointer to the dense matrix
`d_compressed`	Device	OUT	Pointer to the *compressed* matrix
`stream`	Host	IN	HIP stream for the computation

Properties

- The routine requires no extra storage
- The routine supports asynchronous execution with respect to `stream`

See rocsparselt_status for the description of the return status.
