MaxCompiler Cheat Sheet

v2 Kernel Interface

MAXELER Technologies

Primitive Types

KernelType: DFEType
KernelObject: DFEVar
Type creation helper functions:
dfeBool()
dfeFix(int int_bits, int frac_bits, SignMode sign_mode)
dfeFloat(int exponent_bits, int mantissa_bits)
dfeInt(int bits)
dfeUInt(int bits)
Operators: cast, +, −, *, /, <, >, <=, >=, eq, ~, &, |, ^, ? :, @, [], <== (connect operator), (+=, *=, >>= etc.)
Constant creation:
constant.var(boolean v) **or**
constant.var(double v) **or**
constant.var(DFEType type, double v)
Create a source-less instance:
DFEVar x = dfeUInt(32).newInstance(myKernel);

Complex Number Type

KernelType: DFEComplexType
KernelObject: DFEComplex
Constructor: DFEComplexType(DFEType type) e.g.:
DFEComplexType t = new DFEComplexType(dfeFloat(8,24));
Operators: +, +=, −, −=, *, *=, /, /=, <== (connect operator)
Constant creation:
constant.cplx(double real, double imaginary) **or**
constant.cplx(DFEType type, double real, double imaginary)
Create a source-less instance:
DFEComplex x = new DFEComplexType(dfeFloat(8,24)).newInstance(this);

Vector Type

KernelType: DFEVectorType
KernelObject: DFEVector
Constructor: DFEVectorType(KernelType type, int size)
e.g. DFEVectorType t = new DFEVectorType(dfeBool(), 4);
Set an element: vector[i] <== KernelObject o
Get an element: vector[i]
e.g. vectorB[i] <== vectorA[i];
Constant creation:
constant.vector(DFEType type, double... vs)
Create a source-less instance:
DFEVector<DFEVar> x = new DFEVectorType<DFEVar>(dfeUInt(32), 2).newInstance(this)

Inputs and Outputs

Stream inputs:
KernelObject io.input(String name, KernelType type)
KernelObject io.input(String name, KernelType type, DFEVar control_var)
e.g. DFEVar x = io.input("x", dfeUInt(32));
Scalar inputs:
KernelObject io.scalarInput(String name, KernelType type)
Stream outputs:
io.output(String name, KernelObject output, KernelType type)
io.output(String name, KernelObject output, KernelType type, DFEVar control_var)
e.g. io.output("y", x, dfeUInt(32));

Counters

Simple counters:
DFEVar control.count.simpleCounter(int bit_width) **or**
DFEVar control.count.simpleCounter(int bit_width, DFEVar wrap_point) **or**
DFEVar control.count.simpleCounter(int bit_width, int wrap_point)
Chained counters:
CounterChain control.count.makeCounterChain() **or**
CounterChain control.count.makeCounterChain(DFEVar enable)
  DFEVar addCounter(DFEVar max, int increment)
  DFEVar addCounter(long max, int increment)
  DFEVar getCounterWrap(KernelObject counter)
Advanced counters:
Counter control.count.makeCounter(Count.Params params)
Count.Params control.count.makeParams(int bit_width)
  withCountMode(Count.CountMode count_mode)
Count.CountMode {NUMERIC_INCREMENTING, SHIFT_LEFT, SHIFT_RIGHT}
  withEnable(DFEVar enable)
  withInc(long inc)
  withInitValue(long value)
  withMax(DFEVar max) **or**
  withMax(long max)
  withWrapMode(Count.WrapMode wrap_mode)
  withWrapValue(long wrap_value)
Count.WrapMode {COUNT_LT_MAX_THEN_WRAP, MODULO_MAX_OF_COUNT, STOP_AT_MAX}

Stream Manipulation

Fixed offset:
KernelObject stream.offset(KernelObject src, int offset)
Variable offset:
OffsetExpr stream.makeOffsetParam(String name, int min, int max)
KernelObject stream.offset(KernelObject src, Stream.OffsetExpr offset_eq)
Dynamic offset:
KernelObject stream.offset(KernelObject src, DFEVar offset, int min, int max)

Fast Memory (FMem)

Memory<KernelObject> mem.alloc(DFEType type, int depth)

DFEVar Memory.read(DFEVar address)
void Memory.write(DFEVar address, DFEVar data, DFEVar enable)
DFEVar Memory.port(DFEVar address, DFEVar data_in, DFEVar write_enable, RamWriteMode portMode)
Mem.RamWriteMode {READ_FIRST, WRITE_FIRST}

void Memory.mapToCPU(String name)
void Memory.setContents(double[] contents)
void Memory.setContents(Bits[] contents)

KernelObjects and KernelType

All classes implementing KernelObject define:
watch(String name)
connect(KernelObject in_stream) (<== operator)
KernelObject cast(KernelType type)
KernelType getType()
All classes inheriting from KernelType define:
KernelObject newInstance(KernelLib MyKernel)
Bits encodeConstant(Object value)

SLiC CPU Examples

```
#include "MaxFileName.h"
#include <MaxSLiCInterface.h>
```

Basic Static:

```
MaxFileName(n, x, y);
MaxFileName_InterfaceName(n, x, y);
```

Advanced Static:

```
max_file_t *myMaxFile = MaxFileName_init();
max_engine_t *myDFE = max_load(myMaxFile, "local:*");

MaxFileName_actions_t actions;
actions.param_x = n;
actions.instream_x = x;
actions.outstream_y = y;

MaxFileName_run(myDFE, &actions);

max_unload(myDFE);
```

Advanced Dynamic:

```
max_file_t *myMaxF = MaxFileName_init();
max_engine_t *myDFE = max_load(myMaxF, "local:*");

max_actions_t *actions = max_actions_init(myMaxF, "default");
max_set_param_uint64t(actions, "N", n);
max_queue_input(actions, "x", x, nBytes);
max_queue_output(actions, "y", y, nBytes);

max_run(myDFE, actions);

max_unload(myDFE);
```