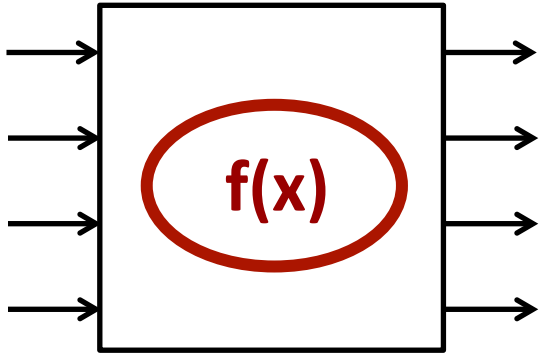
```
def sorter_network( input_list ):
  return sorted( input_list )
```

[3, 1, 2, 0] ---> f(x) ---> [0, 1, 2, 3]

```
class SorterNetworkFL( Model ):
  def __init__( s, nbits, nports ):

    s.in_  = InPort [nports]( nbits )
    s.out  = OutPort[nports]( nbits )


    @s.tick_fl
    def logic():
      for i, v in enumerate( sorted( s.in_ ) ):
        s.out[i].next = v
```



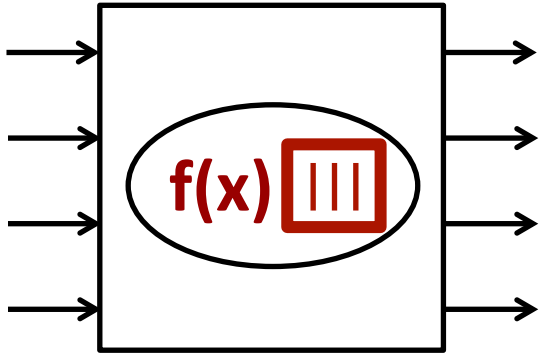
```
def sorter_network( input_list ):
  return sorted( input_list )
```

[3, 1, 2, 0] ---> f(x) ---> [0, 1, 2, 3]

```
class SorterNetworkCL( Model ):
  def __init__( s, nbits, nports ):

    s.in_  = InPort [nports]( nbits )
    s.out  = OutPort[nports]( nbits )


    @s.tick_cl
    def logic():
      # behavioral logic + timing delays
```



```
def sorter_network( input_list ):
  return sorted( input_list )
```

[3, 1, 2, 0] ---> f(x) ---> [0, 1, 2, 3]

```
class SorterNetworkRTL( Model ):
  def __init__( s, nbits, nports ):

    s.in_  = InPort [nports]( nbits )
    s.out  = OutPort[nports]( nbits )


    @s.tick_rtl
    def seq_logic():
      # sequential logic

    @s.combinational
    def comb_logic():
      # combinational logic
```