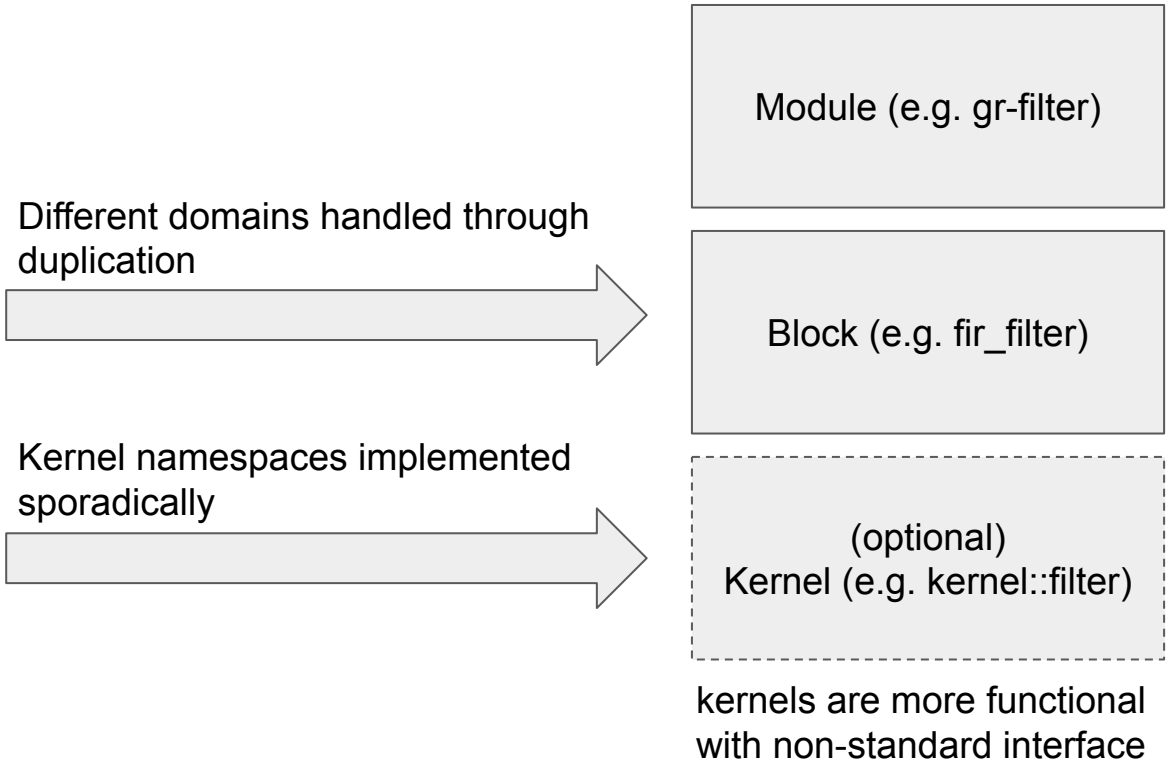
GR 4.0 Block API

Review and open design questions

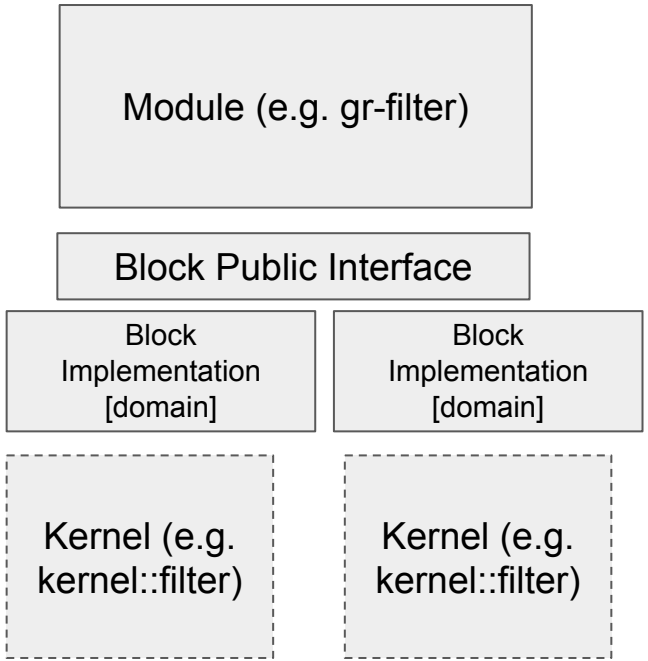
Goals

- Accelerate developers of signal processing code to get to "Insert Code Here" more easily
- Reduce duplication of code
- Swap domain implementations at run time from a single block
- Signal processing library apart from full GR installation (scipy.signal or Matlab toolbox-like) - something like our existing kernel namespaces
- Single public header that hides implementation
- *Simplify.*

Current GR Module Hierarchy



Proposed GR 4.0 Block Hierarchy



kernels are more functional with non-standard interface

- Move block-like things up into block
- Use pimpl-ish pattern to allow for multiple block implementations
- Kernel lib is more matlab-like, less generic

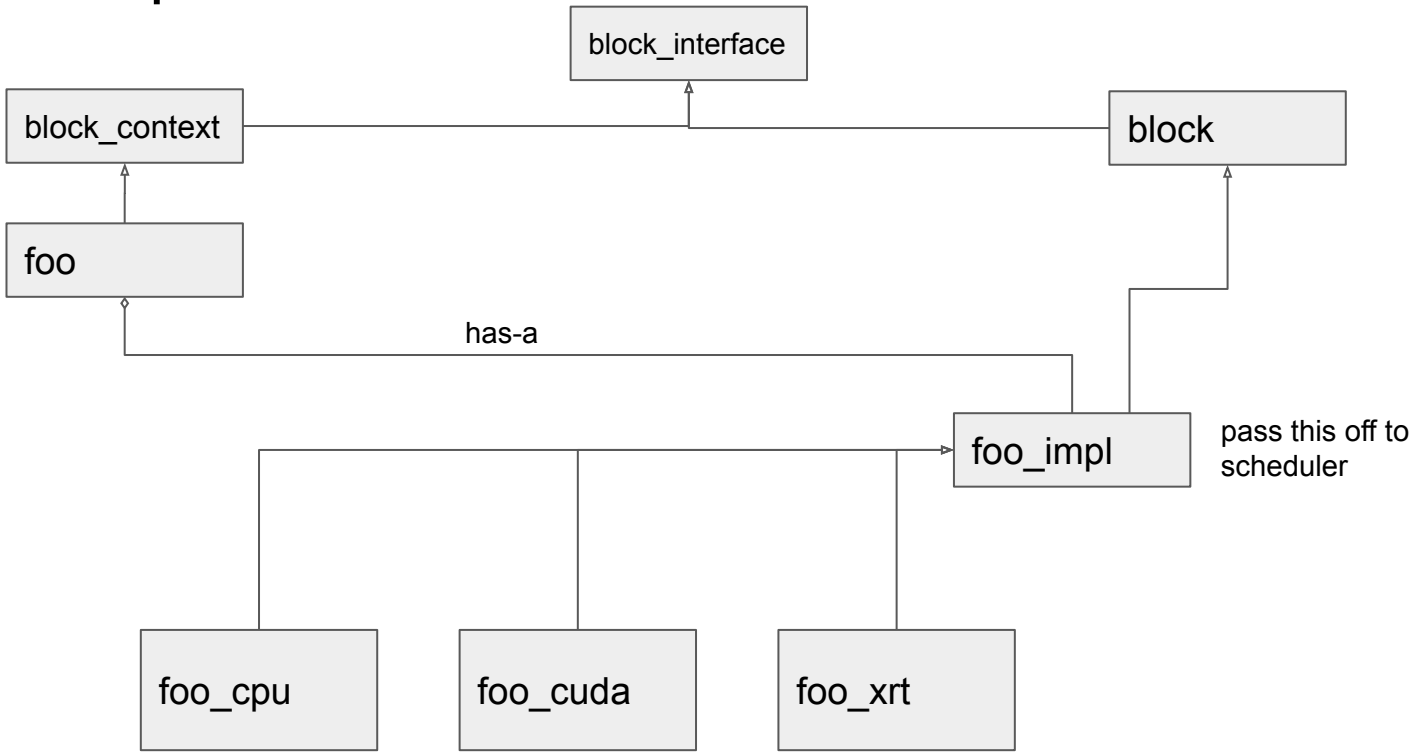
Top level API

```
auto add_cpu = blocks::add::make(2)

-- or --

auto add_cuda = blocks::add::make(2)->cuda()

-- or --

auto add_blk = blocks::add::make(2)
add_blk->set_impl(add_blk::supported_impl::CUDA)

fg->connect(null_source,0,add,0)
fg->connect(null_source,1,add,1)
fg->connect(add_blk,0,null_sink,0)
```

Design Pattern

- hide the implementation
- provide for multiple implementations
- transition between them?
- prevent a ton of files / boilerplate
  - macros, tools to generate code :/ ??
- Strategy design pattern?

Multiple implementations



Code Organization

Goal:

- Make it easier to work on all the moving pieces of a block
- Any way to bring all block files together in a dir or some organization?
- Code Generation 😬
  - Have the user specify:
    - Ports
    - Parameters (with flags)
    - Work function (per domain)

Major Block API differences

Goal is to ***simplify***

- No forecast - return early from work with code / info
- No dynamic nports
- No history
- work() takes in/out structs - signal to caller what took place
    - things that were previously flagged in block, now returned as info

Want to nail down API to get be able to start propagating blocks

Block API (block.h)

GR 3.9	newsched
history	☒
sample_delay	? - (used for tag propagation)
forecast	☒
fixed_rate	?
general_work	✔ work()
start/stop	✔
output_multiple	✔
alignment	?
consume/produce	☒ - handled with work_{input,output} structs
relative_rate	✔
nitems_{read,written}	☒ - handled with work_{input,output} structs
tag_propagation_policy	✔
{min,max}_output_items	port?
{min,max}_output_buffer	port / edge ?
pc_* (perf_counters)	?
rpc	? - handle consistently
processor_affinity / thread priority	☒ - scheduler specific
system port	?
notify_msg_neighbors	☒ - scheduler specific
clear_finished	☒ - scheduler specific

Block API (basic_block.h)

GR 3.9	newsched
unique_id	✔️ id()
symbolic_id	☒
alias/alias_pmt	✔️ alias()
name	✔️ name()
symbol_name	☒
identifier	☒
{input,output}_signature	replaced by ***port*** classes
message_port_*	constructed as ***ports*** - add_port() method to maintain list
_post	port
empty_[handled_]p	port
nmsgs	port
insert_tail	port
delete_head_nowait	port
get_iterator	port
erase_msg	port
has_msg_port	port
get_msg_map	port
check_topology	☒ - no dynamic nports - ports are either mandatory or optional
color	currently hacked in - needs to be part of flowgraph