Fork me on GitHub

Sharing Generic Class Libraries in SystemVerilog Makes Coding Fun Again

Keisuke Shimizu
ClueLogic, LLC

Look familiar?

```
`define min(x,y) ((x)<(y)?(x):(y))
```

```
function bit random_bool( int unsigned true_percentage );
  return $urandom_range( 99 ) < true_percentage;
endfunction
```

```
task watch_event( event e, time timeout );
  fork // thread firewall
    begin
      fork
        @e ;
        #timeout $error( "timeout" );
      join_any
      disable fork;
    end
  join
endtask
```

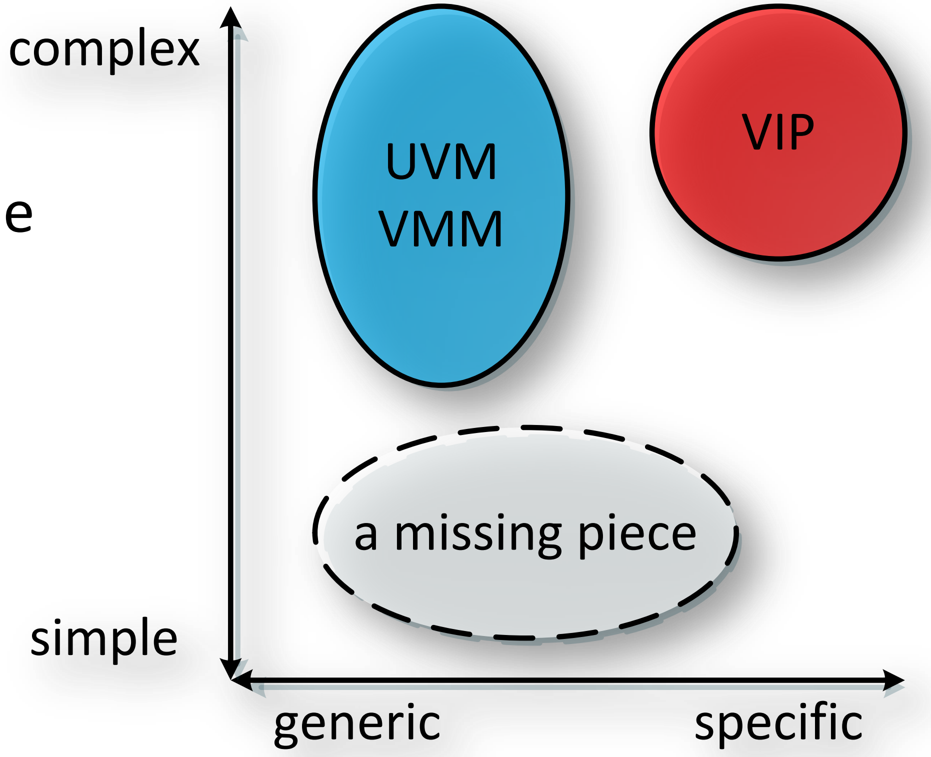
… but no open library is available

Why are these functions rarely shared?

- Most people lack the time to develop them
  - Creating robust functions requires thorough verification
  - Defining consistent and configurable functions is not trivial
  - Writing API documents is a burden
- Some functions are relatively easy to develop
- Sharing them might be difficult, both technically and legally

What does a shared library offer you?

- Sharing a library allows you to:
  - save time
  - write more readable code
  - focus on project-specific logic
  - avoid common mistakes
- A shared library can evolve and be fixed by peers



How did we create this library?

- Investigated nine verification projects from various domains:
  - Array processor
  - Image processor
  - SoC interconnects
  - Mobile peripherals
- Surveyed other programming languages
  - C++, Java, Python, Perl, Ruby, JavaScript

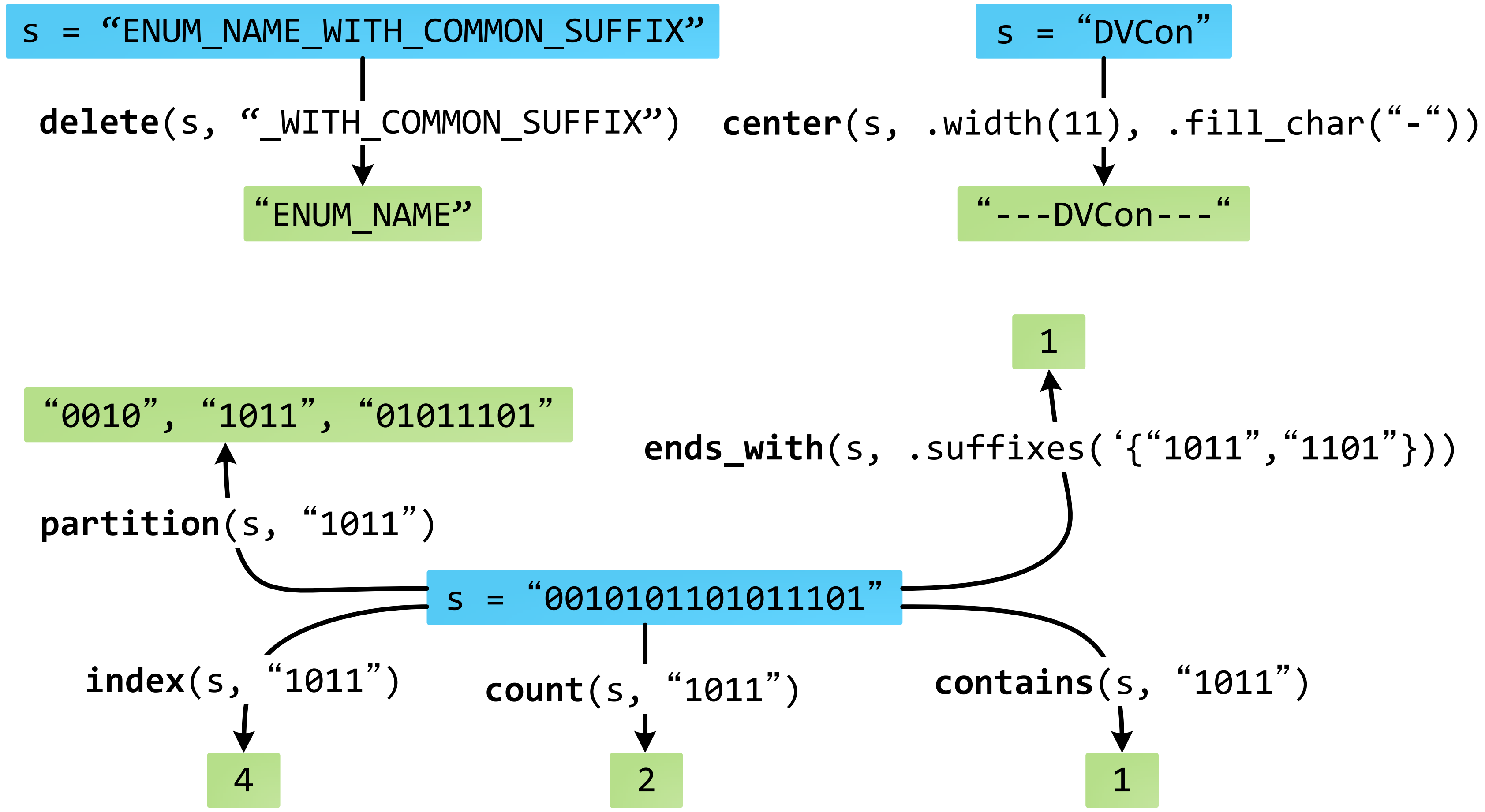
What functions does the library provide?

- Text Processing
- Containers
- Strategy
- Verification-specific
- Domain-specific

… about 360 methodology-independent low-level functions

TEXT PROCESSING

What can you do with the text-processing library?



```
s = "ENUM_NAME_WITH_COMMON_SUFFIX"
delete(s, "_WITH_COMMON_SUFFIX")
"ENUM_NAME"

s = "DVCon"
center(s, .width(11), .fill_char("-"))
"---DVCon---"

s = "0010101101011101"
partition(s, "1011")
"0010", "1011", "01011101"

ends_with(s, .suffixes('{"1011","1101"}))
1

index(s, "1011")
4

count(s, "1011")
2

contains(s, "1011")
1
```

What functions are in the text-processing class?

Python

```
capitalize
center
change
chomp
colorize
contains_str
delete
insert
join_str
lc_first
ljust
lstrip
```

Java Script

```
replace
reverse
rjust
rstrip
slice
slice_len
strip
swap_case
title_case
trim
uc_first
untabify
```

Java

```
contains
ends_with
is_alpha
is_digit
is_lower
is_printable
is_single_bit_type
is_space
is_upper
only
starts_with
```

Ruby

Perl

C++

```
chop
count
find_any
hash
index
partition
rfind_any
rindex
rpartition
rsplit
split
split_lines
```

How we implemented the text-processing class

- All text functions are static

```
s = text::capitalize( “capitalize me” );
```

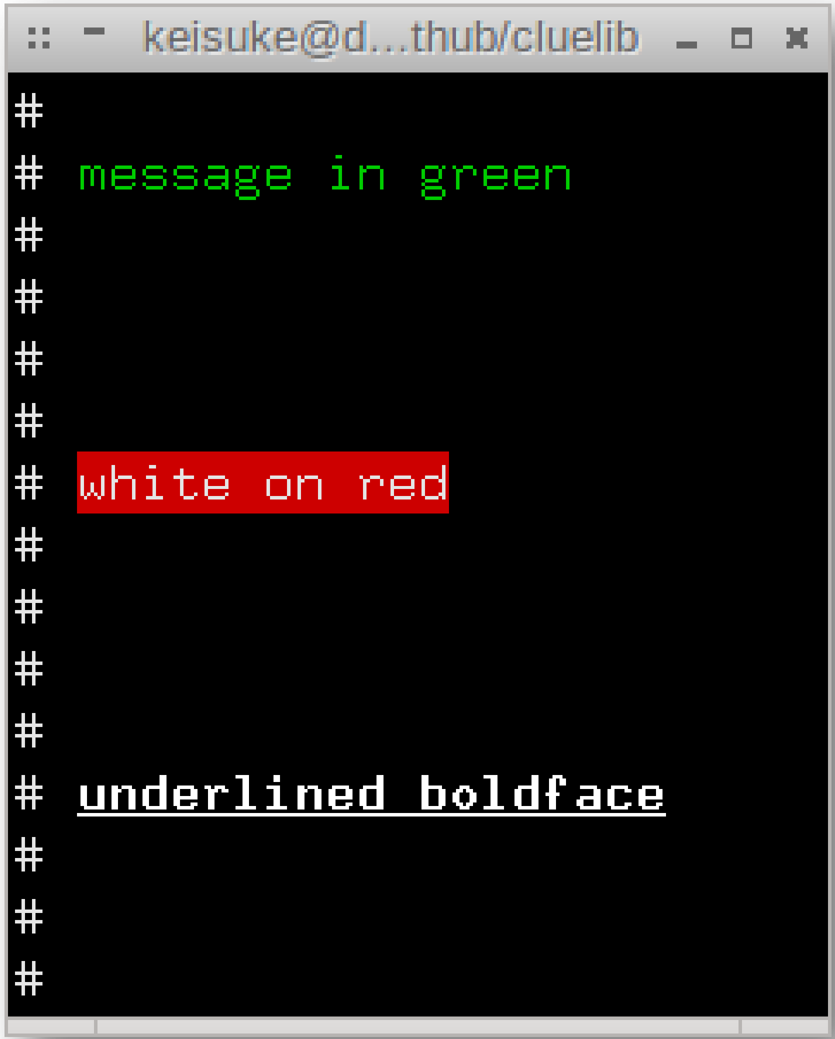
- Two implementations
  - SystemVerilog only
  - SystemVerilog with C++

My favorite function: colorize

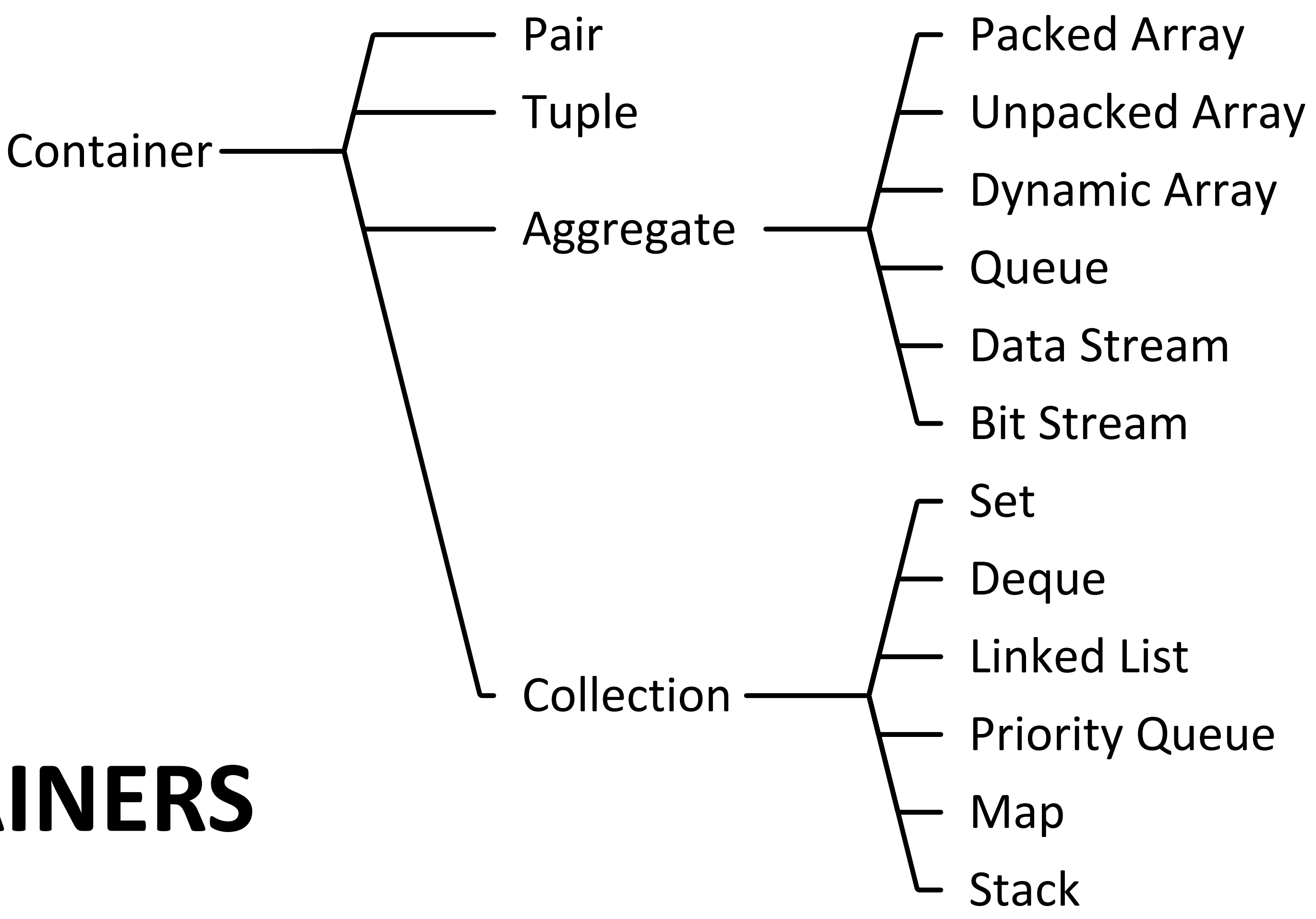
```
$display( text::colorize(
 “message in green”,
 FG_GREEN) );
```

```
$display( text::colorize(
 “white on red”,
 FG_WHITE, BG_RED) );
```

```
$display( text::colorize(
 “underlined boldface”,
 .underline(1), .bold(1)) );
```



```
#
# message in green
#
#
#
#
# white on red
#
#
#
#
# underlined boldface
#
#
#
```


Container
Pair
Tuple
Aggregate
Collection
Packed Array
Unpacked Array
Dynamic Array
Queue
Data Stream
Bit Stream
Set
Deque
Linked List
Priority Queue
Map
Stack

CONTAINERS

Container: A data structure that collects other objects

- Pair (🟥,🟨)

```
pair#(int, string) p;
p = new( 123, “message” );
return p; // as a single unit
```

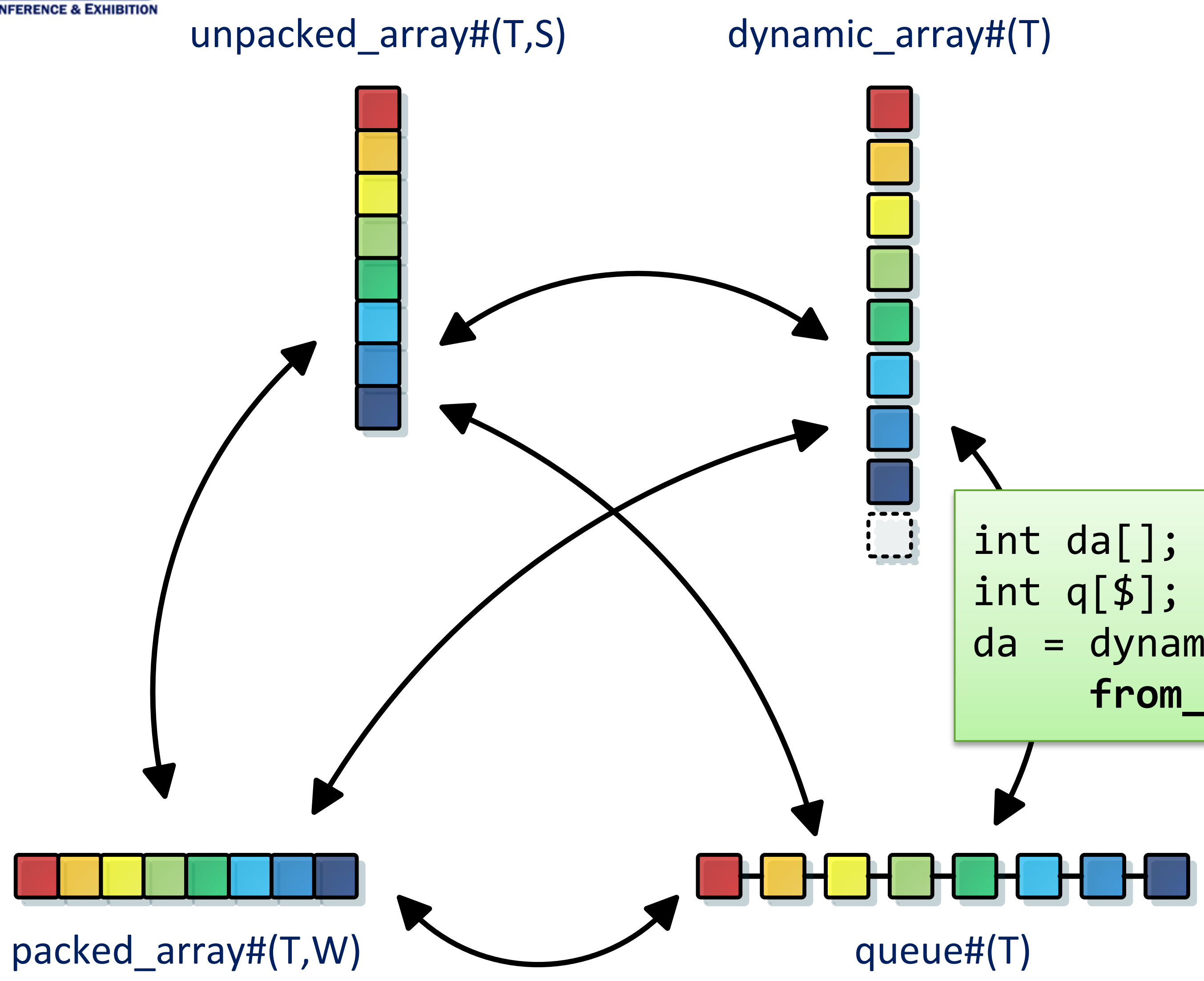
- Tuple (🟥,🟧,🟨,🟩,🟩,🟦,🟦,🟦)
- Aggregates
- Collections

All containers are parameterized classes.

Aggregate classes

- Provides utility functions for:
  - Packed Array
  - Unpacked Array
  - Dynamic Array
  - Queue
  - Data Stream
  - Bit Stream

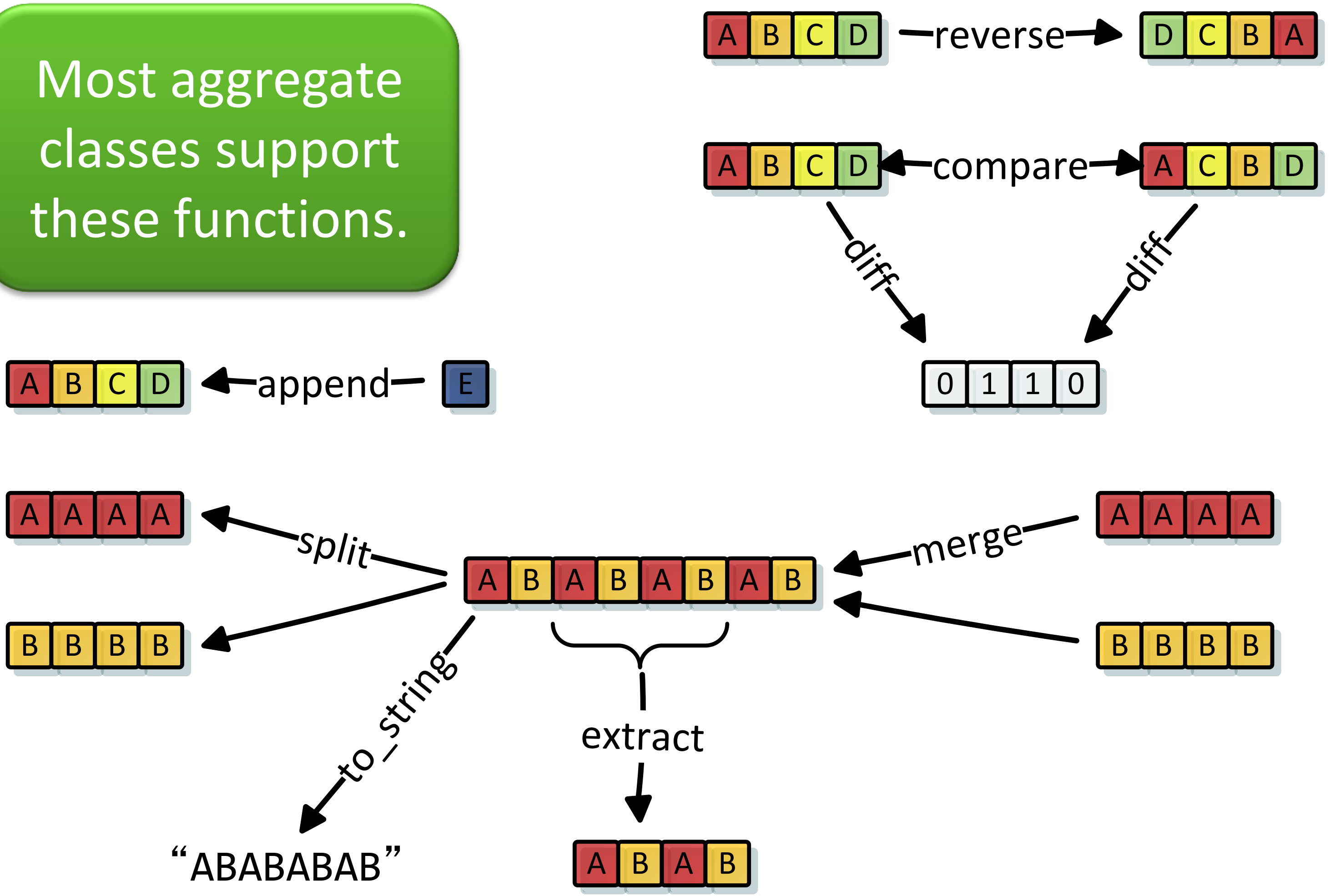
Array converters



```
int da[];
int q[$];
da = dynamic_array#(int)::
     from_queue(q);
```

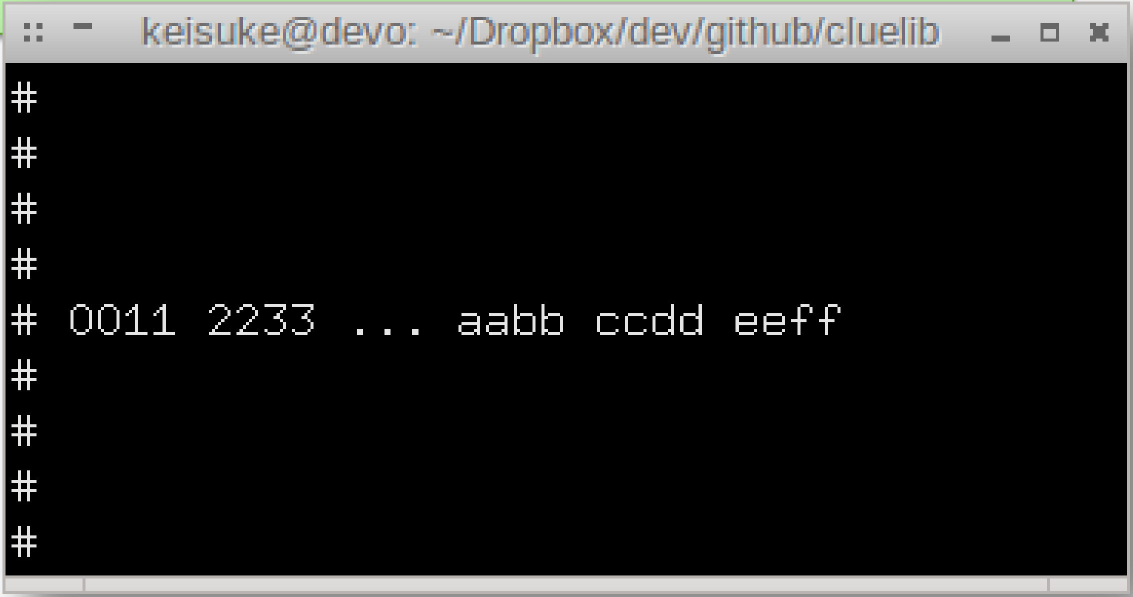
What can you do with the array functions?

Most aggregate classes support these functions.



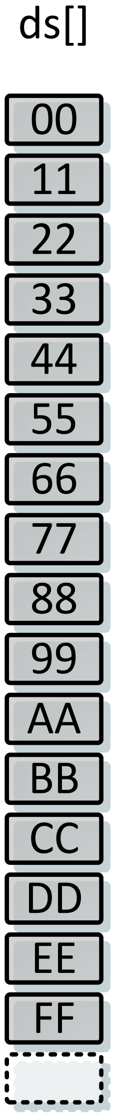
My favorite functions: to_string

```
bit[7:0] ds[]; // data stream
ds = new[16](
   '{ 'h00, 'h11, 'h22, 'h33, 'h44, 'h55, 'h66, 'h77,
      'h88, 'h99, 'hAA, 'hBB, 'hCC, 'hDD, 'hEE, 'hFF });
$display( data_stream#(bit,8)::to_string(
  ds, .group(2), .num_head(4), .num_tail(6)));
```



```
keisuke@devo: ~/Dropbox/dev/github/cluelib
#
#
#
#
# 0011 2233 ... aabb ccdd eeff
#
#
#
#
```

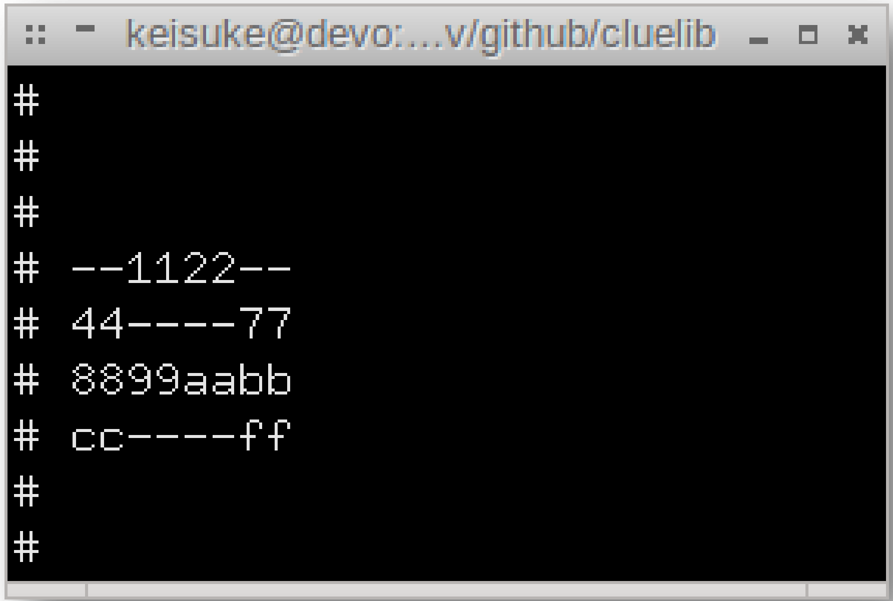
... can display a part of the data stream



And to_string_with_en

```
bit[7:0] ds[]; // same data stream as before
bit de[];      // data enables
de = new[16]( '{ 0, 1, 1, 0, 1, 0, 0, 1,
                 1, 1, 1, 1, 1, 0, 0, 1 });
$display( data_stream#(bit,8)::to_string_with_en(
  ds, de, .group(4), .group_separator("\n")));
```

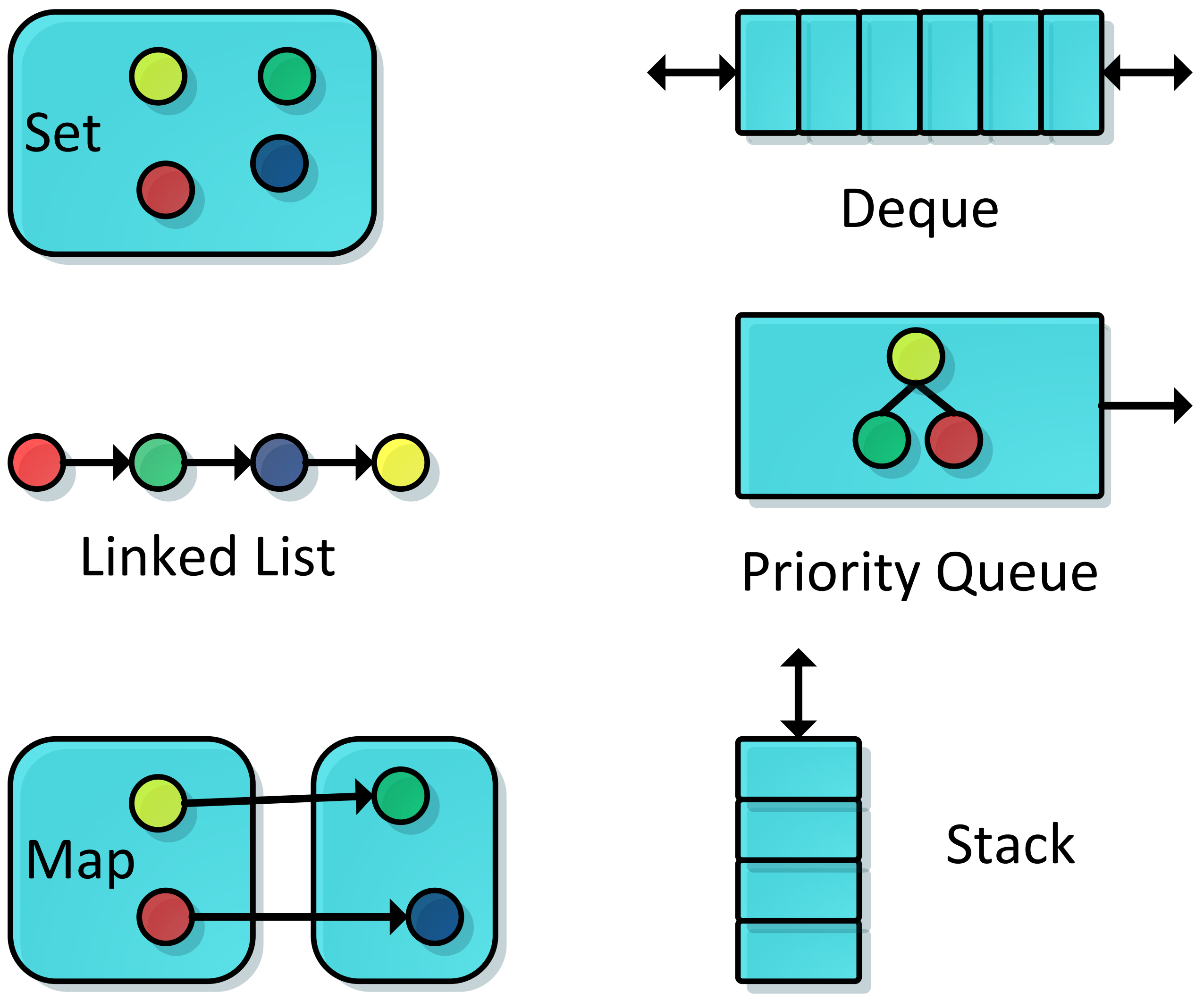
ds[]	de[]
00	0
11	1
22	1
33	0
44	1
55	0
66	0
77	1
88	1
99	1
AA	1
BB	1
CC	1
DD	0
EE	0
FF	1



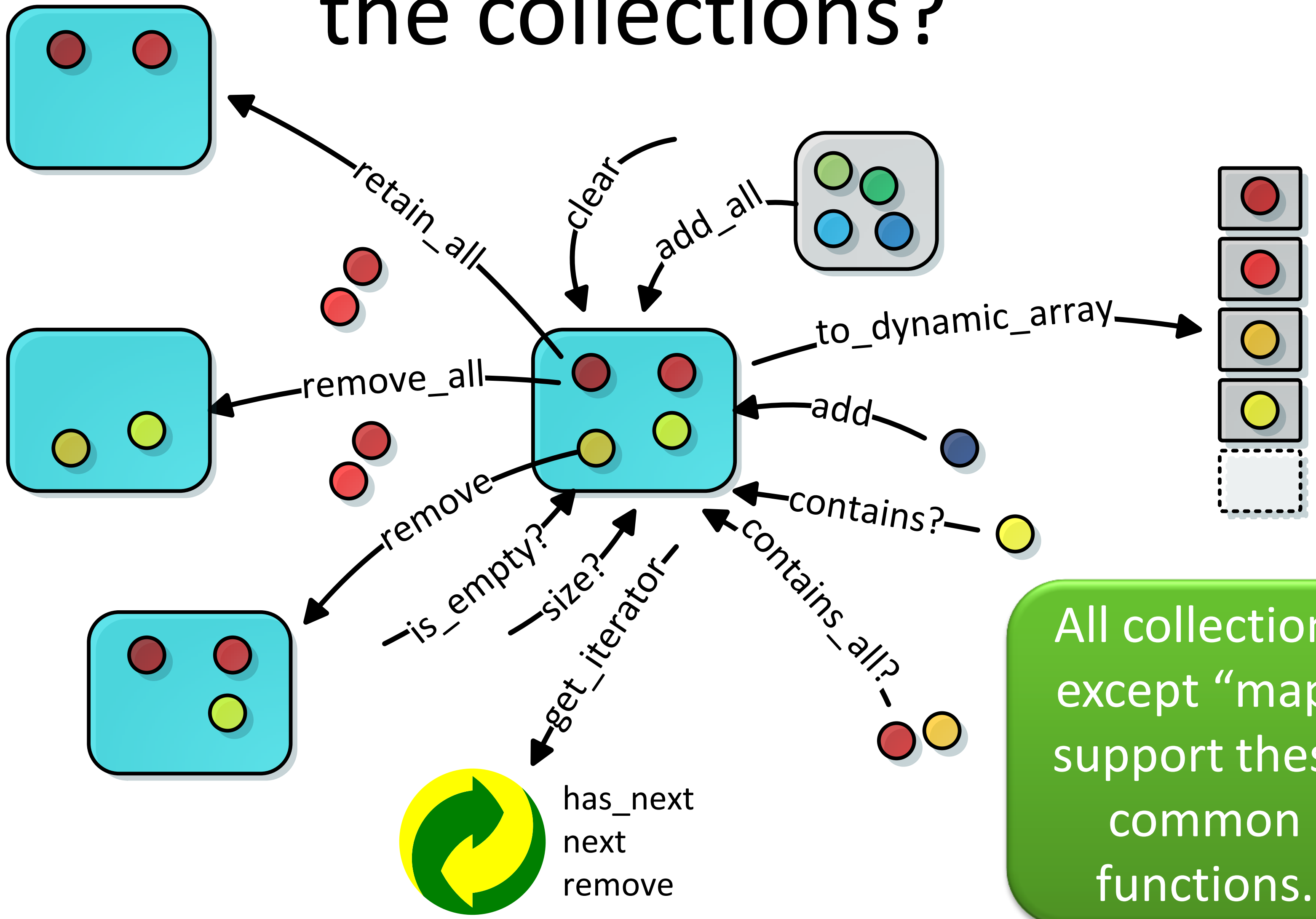
```
#
#
#
# --1122--
# 44----77
# 8899aabb
# cc----ff
#
#
```

… displays enabled data only

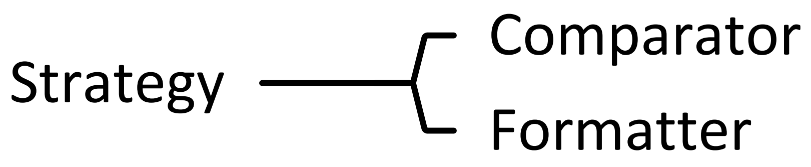
Collections



What can you do with the collections?



All collections except "map" support these common functions.

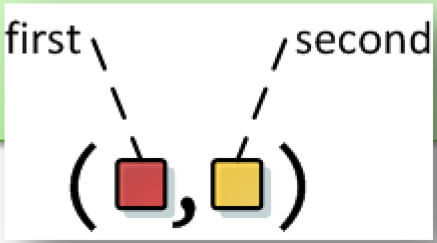


STRATEGY

Strategy: A family of interchangeable algorithms

- Comparators
  - Example: `pair_comparator`

```
class pair_comparator#(type T=pair) extends comparator#(T);
  virtual function bit eq( T x, T y );
    return x.first == y.first && x.second == y.second;
  endfunction
// ... other functions
endclass
```

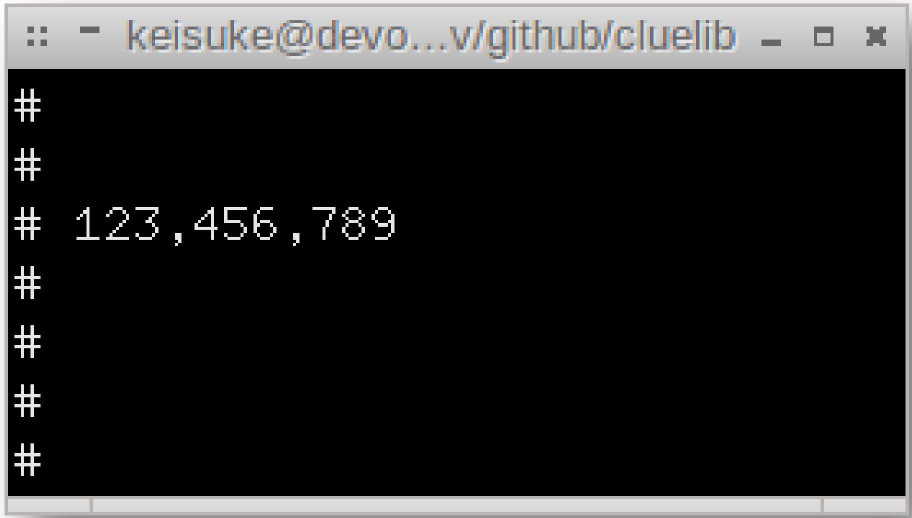


- Formatters
  - Converts an object to a string format

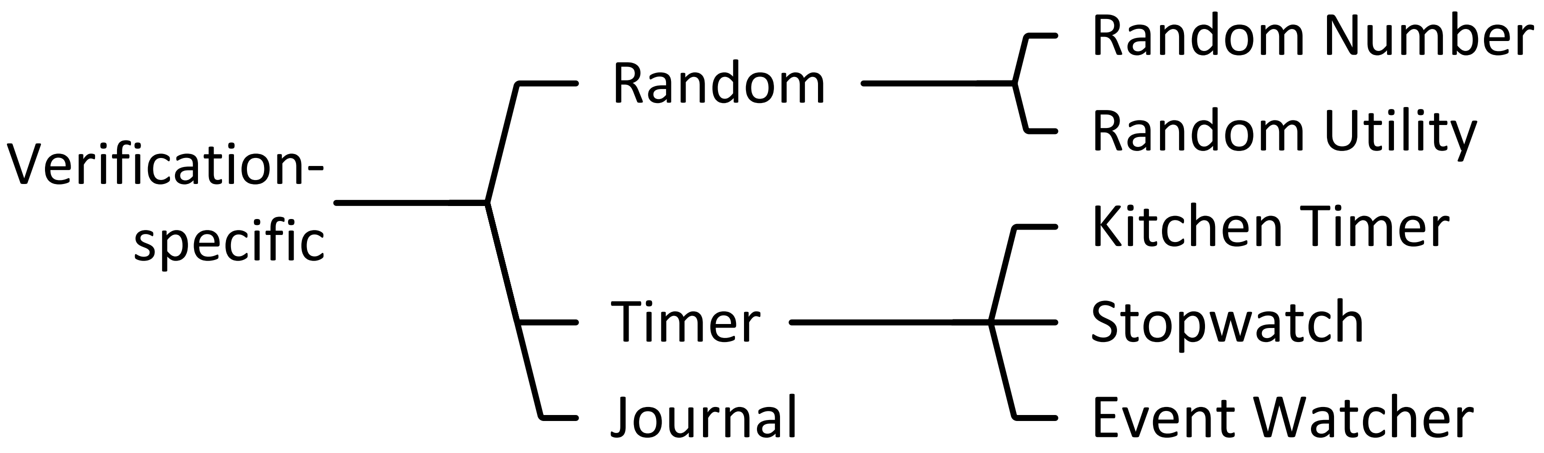
My favorite class: **comma_formatter#(T)**

```
comma_formatter#(longint) com_fmtr =
comma_formatter#(longint)::get_instance();

$display( com_fmtr.to_string( 123456789 ) );
```

... inserts commas as thousands separators



```
#
#
# 123,456,789
#
#
#
#
```



VERIFICATION-SPECIFIC

What can you do with the random classes?

```
random_4_bin_num n = new;

            // min  max  weight
n.db = ‘{ ‘{ 100, 200, 1 },
          ‘{ 300, 400, 2 },
          ‘{ 500, 600, 3 },
          ‘{ 700, 800, 4 } };
assert( n.randomize() );
$display( n.val );
```

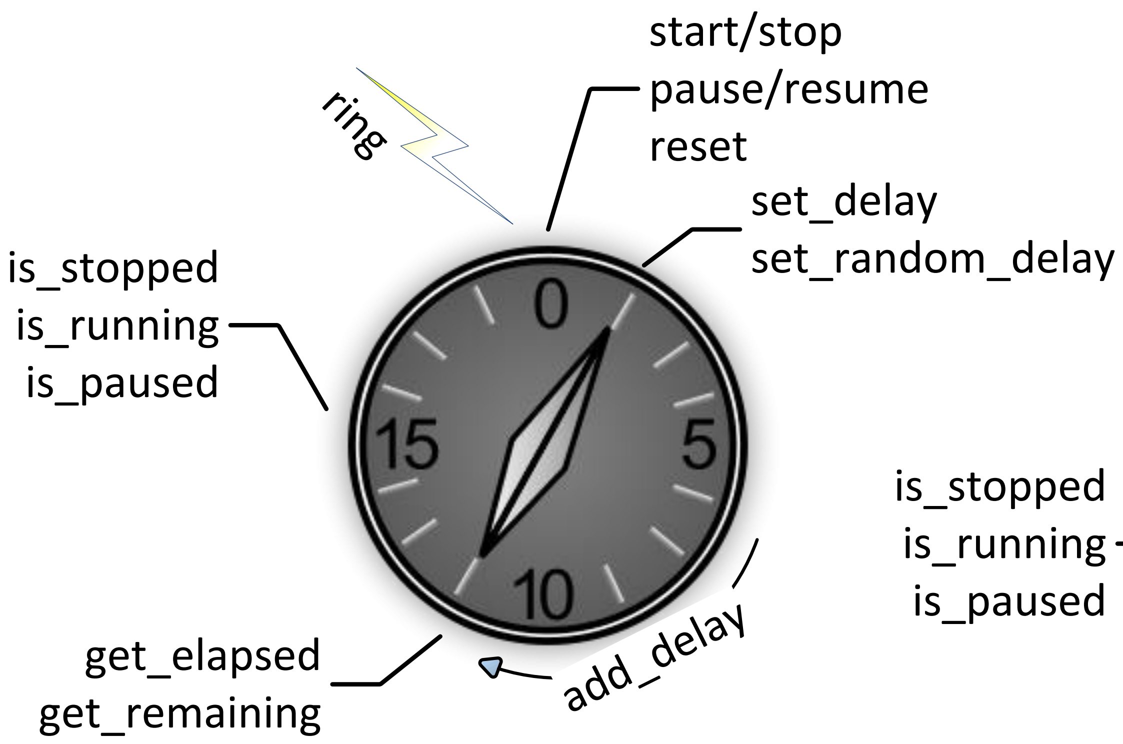
Distribution Bin

min	max	weight
100	200	1
300	400	2
500	600	3
700	800	4

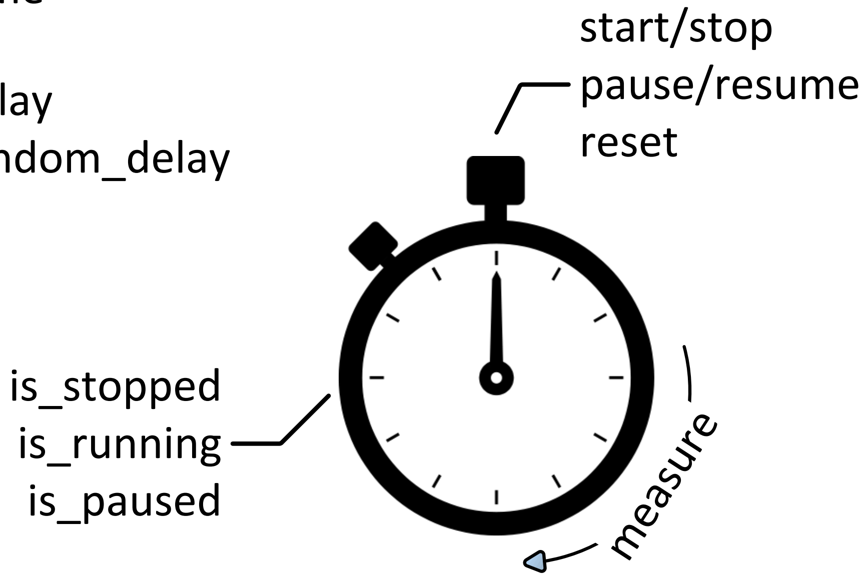
Classes with 2, 4, 8, 16, and 32 distribution bins are defined.

```
if ( random_util::random_bool(70) ) begin // 70% true
```

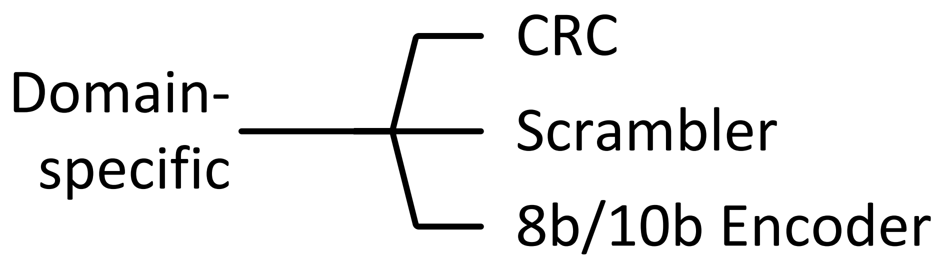
What can do you with the timers?



Kitchen Timer



Stopwatch



DOMAIN-SPECIFIC

Domain-specific

- Offers utility functions to a specific target domain, which are generic enough to reuse
  - CRC
    - 42 commonly used CRC functions
  - Scramble
    - 18 commonly used scramblers
  - 8b/10b Encoder
  - (Color converters)
  - (Matrix functions)

Package and license

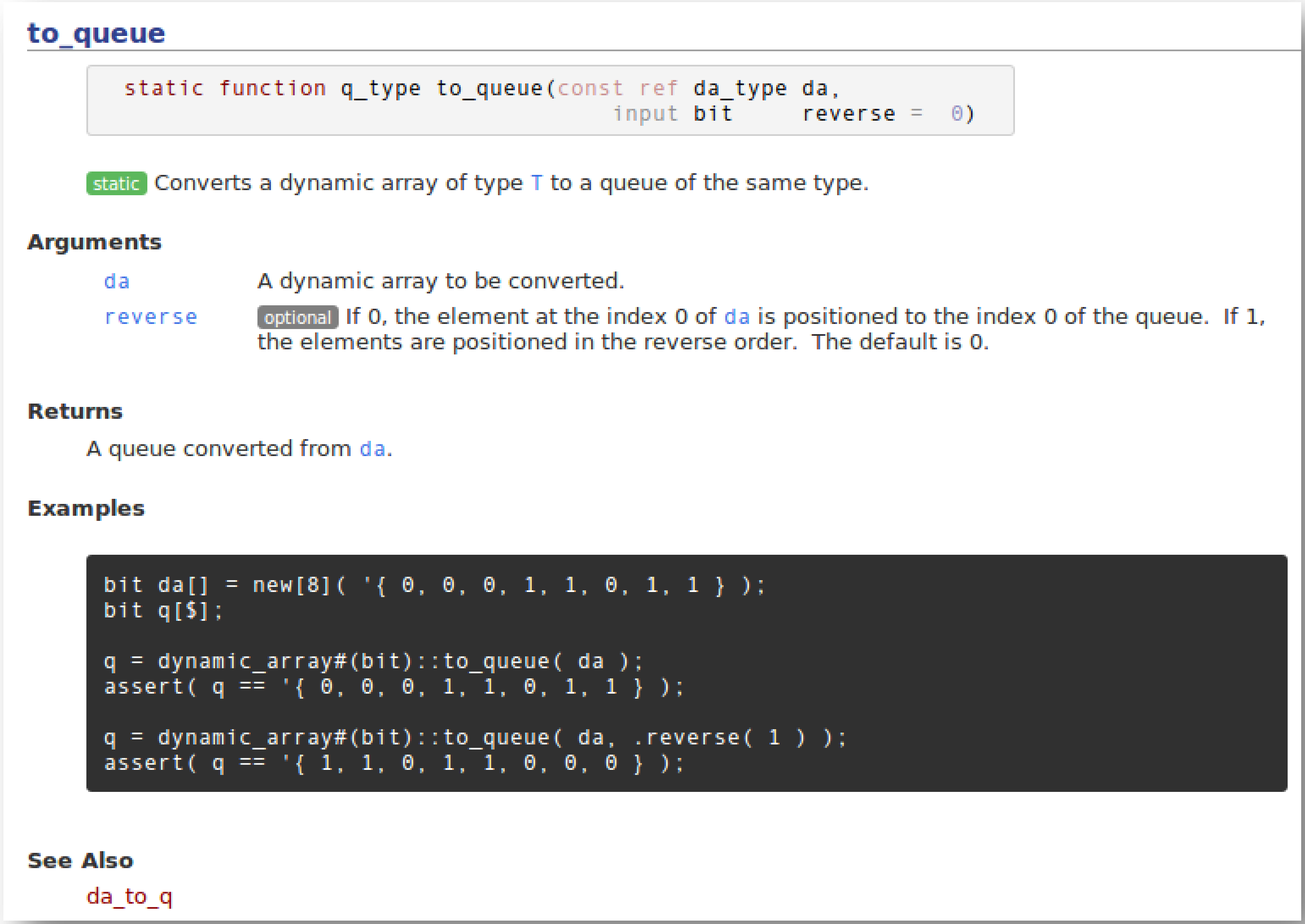
- Using the package

```
import cl::*; // wildcard import
int i = choice#(int)::min(j,k);
```

```
// using the scope resolution operator
int i = cl::choice#(int)::min(j,k);
```

- MIT License

API document



to_queue

```
static function q_type to_queue(const ref da_type da,
                                input bit     reverse =  0)
```

static Converts a dynamic array of type T to a queue of the same type.

Arguments

da	A dynamic array to be converted.
reverse	optional If 0, the element at the index 0 of da is positioned to the index 0 of the queue. If 1, the elements are positioned in the reverse order. The default is 0.

Returns

A queue converted from da.

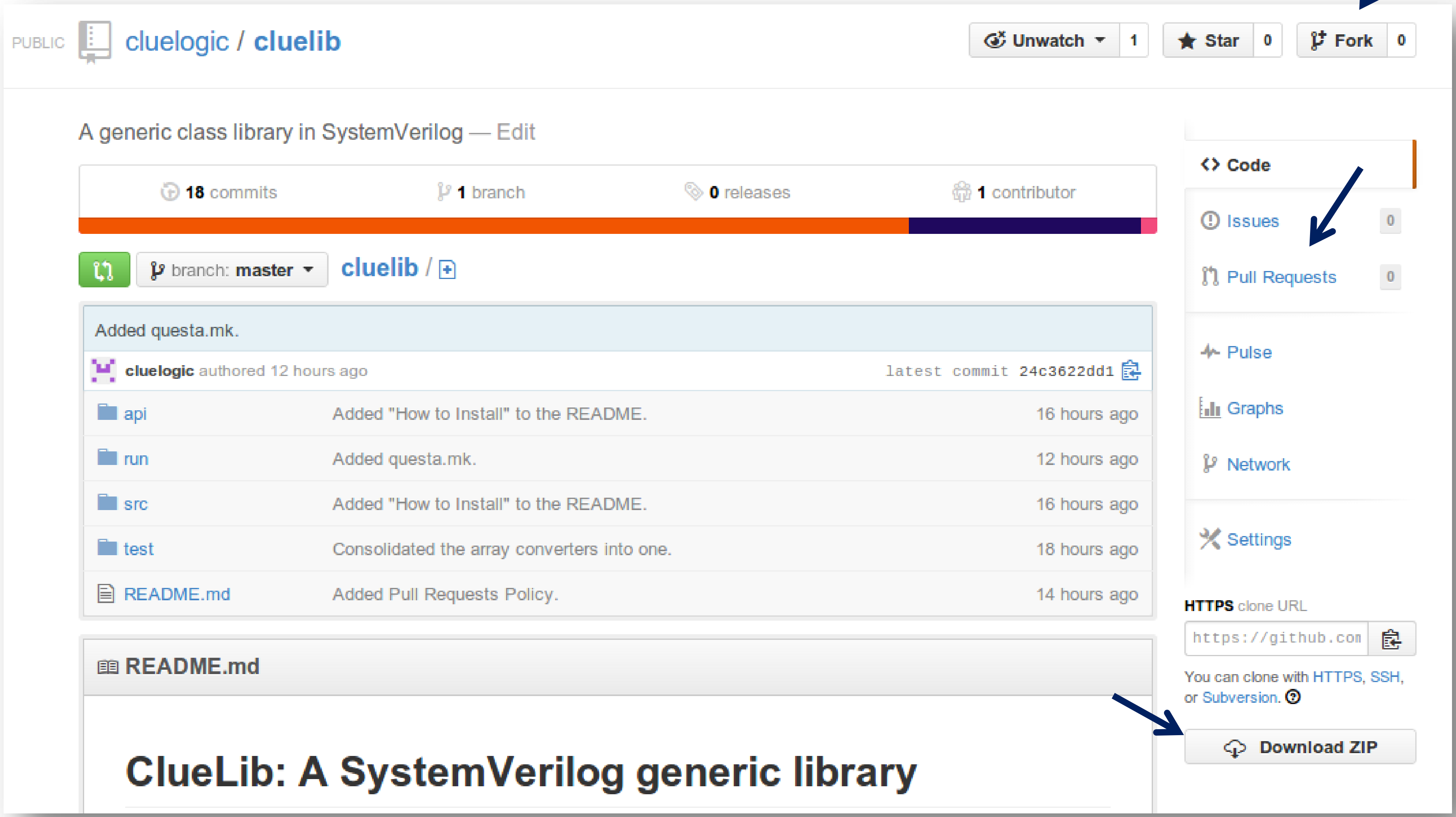
Examples

```
bit da[] = new[8]( '{ 0, 0, 0, 1, 1, 0, 1, 1 } );
bit q[$];

q = dynamic_array#(bit)::to_queue( da );
assert( q == '{ 0, 0, 0, 1, 1, 0, 1, 1 } );

q = dynamic_array#(bit)::to_queue( da, .reverse( 1 ) );
assert( q == '{ 1, 1, 0, 1, 1, 0, 0, 0 } );
```

See Also

da_to_q

Social coding with GitHub



PUBLIC cluelogic / cluelib

Unwatch 1 | Star 0 | Fork 0

A generic class library in SystemVerilog — Edit

18 commits | 1 branch | 0 releases | 1 contributor

branch: master | cluelib / +

Added questa.mk.

cluelogic authored 12 hours ago — latest commit 24c3622dd1

api	Added "How to Install" to the README.	16 hours ago
run	Added questa.mk.	12 hours ago
src	Added "How to Install" to the README.	16 hours ago
test	Consolidated the array converters into one.	18 hours ago
README.md	Added Pull Requests Policy.	14 hours ago

README.md

ClueLib: A SystemVerilog generic library

You can clone with HTTPS, SSH, or Subversion.

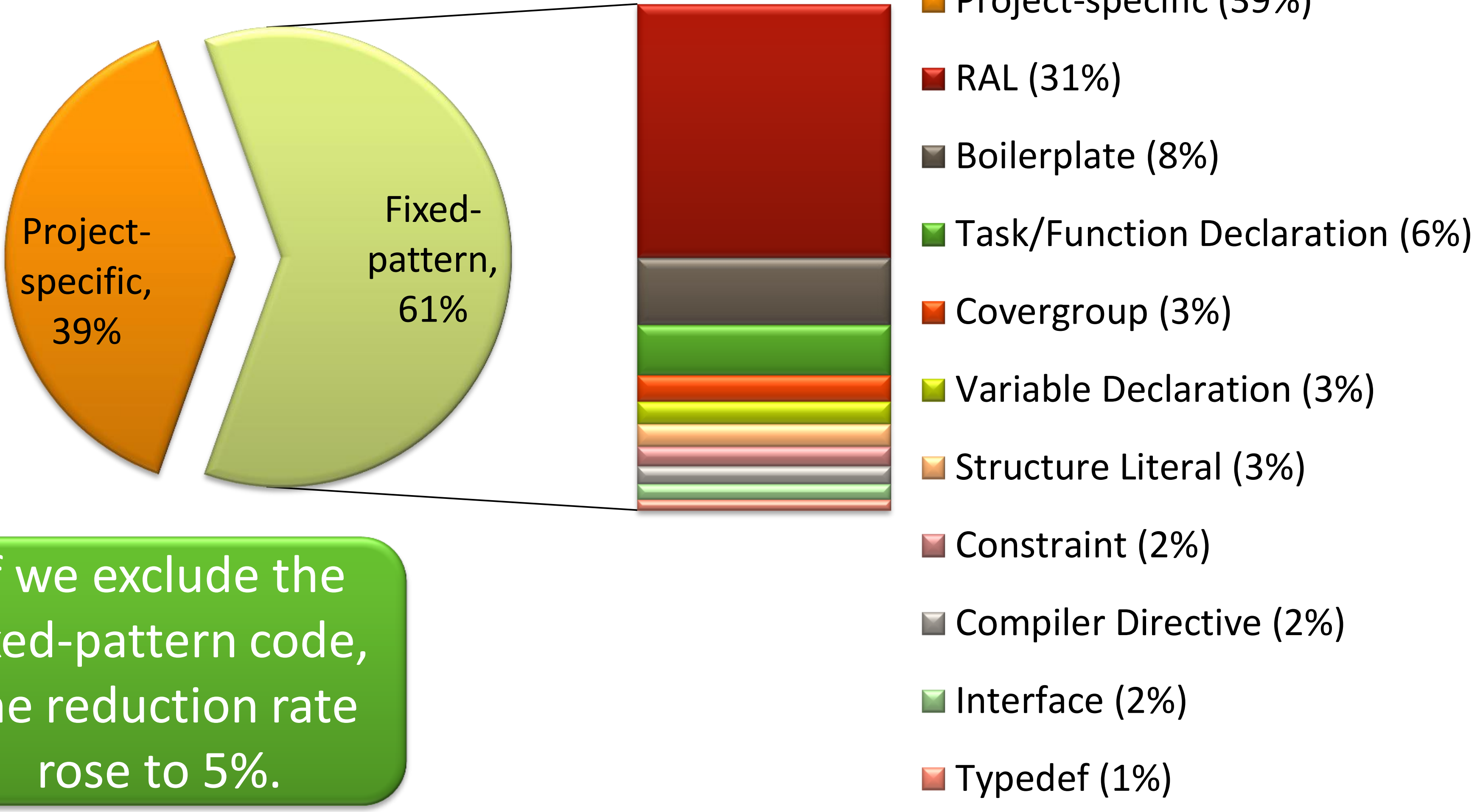
CASE STUDY

How much code could be replaced by the library?

- Investigated nine projects
  - A total of 489,875 lines of class-based code (no module-based code or comments)
- About 2% of code could be replaced by the library

This result fell short of our expectations.

What did the other 98% of the code consist of?



If we exclude the fixed-pattern code, the reduction rate rose to 5%.

Are there any limitations of the library?

- Some functions are probably infeasible to develop in SystemVerilog
  - regular-expressions, image processing, parsing JSON, etc.

What are the disadvantages of an open-source project?

- "Most free software projects fail" [Fogel, '05]
  - New sets of complexities, such as deploying a web site, documentation, and managing contributors
- Verification libraries are usually developed using an employer's resources
  - The copyright of such libraries belongs to the employer

To avoid potential lawsuits, we will not accept any "pull requests" from individual contributors.

What next?

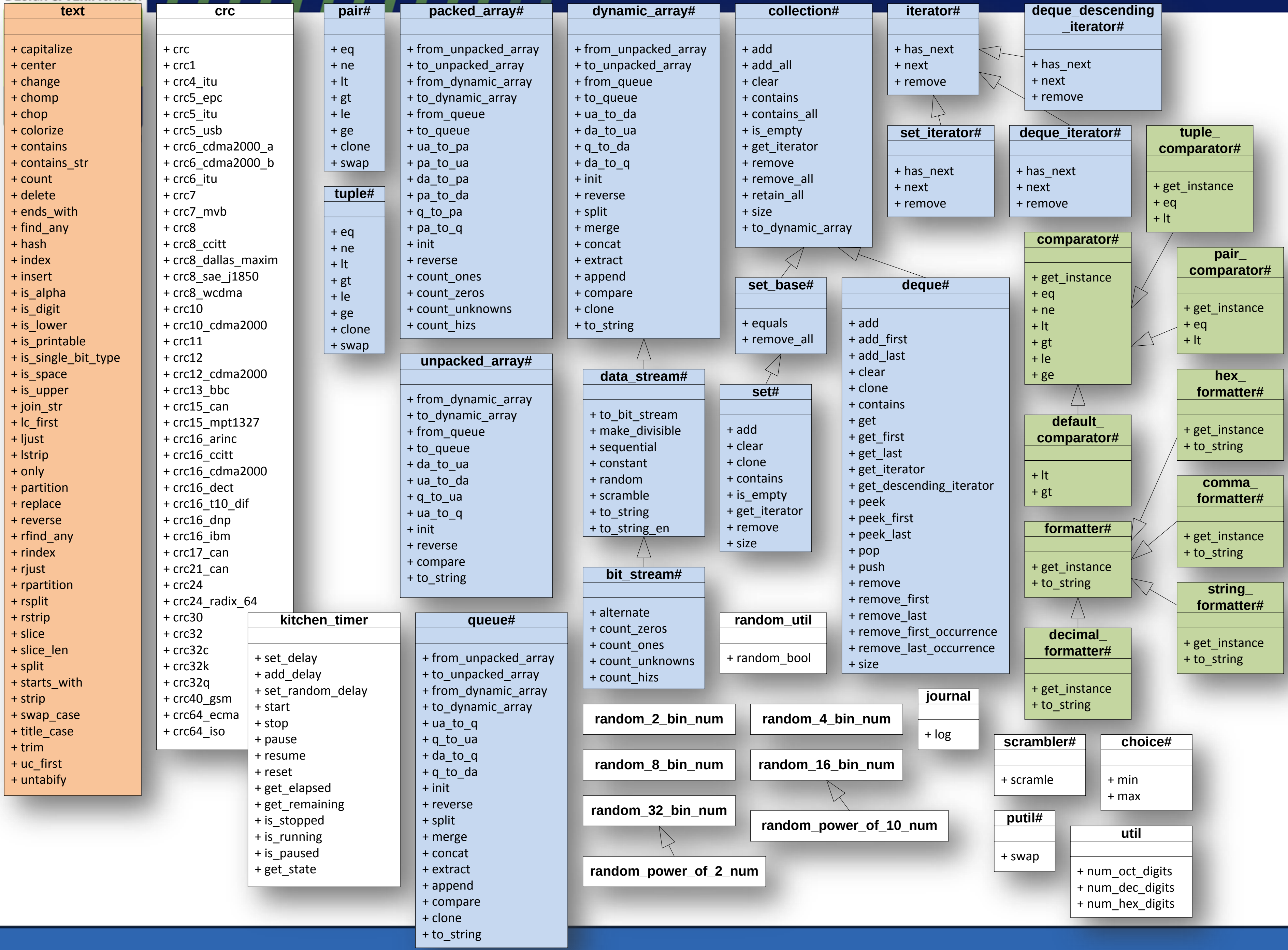
- UVM-dependent library
  - Extension of `uvm_tlm_generic_payload`
  - `uvm_event` waiter
  - `report_phase` that collects the simulation statistics

Is coding in SystemVerilog fun again?

- Absolutely!
- “batteries included”
  - 360+ functions
- Fork me on GitHub
  - https://github.com/cluelogic/cluelib

BACKUP SLIDES

DESIGN & VERIFICATION



SystemVerilog only

```
static function int text::index( string s,
                                 string sub,
                                 int start_pos = 0,
                                 int end_pos = -1 );
  int blen = sub.len();
  // ...
  for ( int i = start_pos; i <= end_pos – blen – 1; i++ )
    if ( s.substr( i, i + blen – 1 ) == sub )
      return i;
  // ...
endfunction
```

… uses a “for” loop to find a match

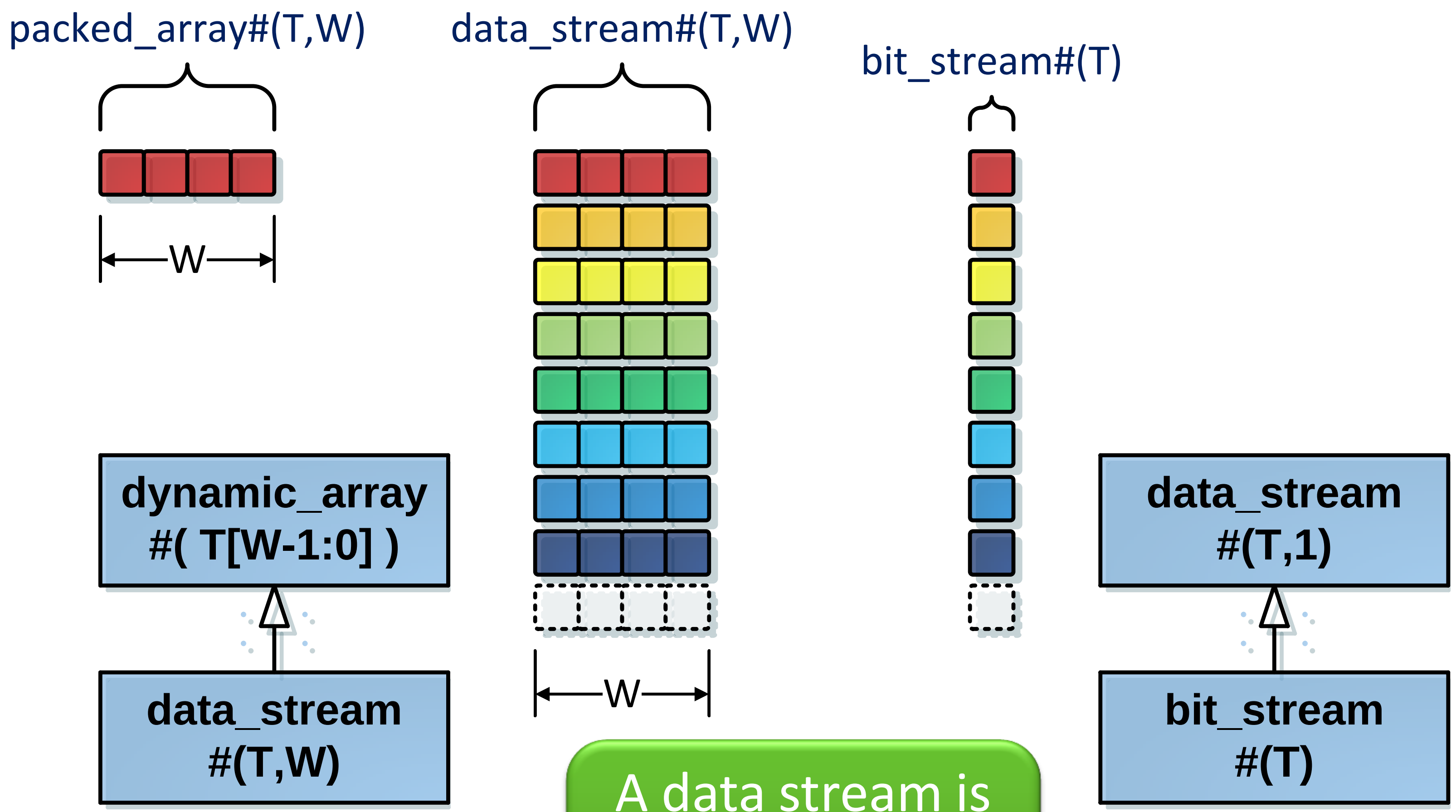
SystemVerilog with C++

```
import “DPI-C” function int c_find( string, string, int );
static function int text::index( string s,
                                 string sub,
                                 int start_pos = 0,
                                 int end_pos = -1 );
  // ...
  i = c_find( s, sub, start_pos );
  if ( i >= 0 && i + sub.len() – 1 <= end_pos ) return i;
  // ...
endfunction
```

```
extern “C” {
  int c_find( const char* s,
              const char* sub,
              const int start_pos ) {
    std::string ss( s );
    return ss.find( sub, start_pos );
  }
}
```

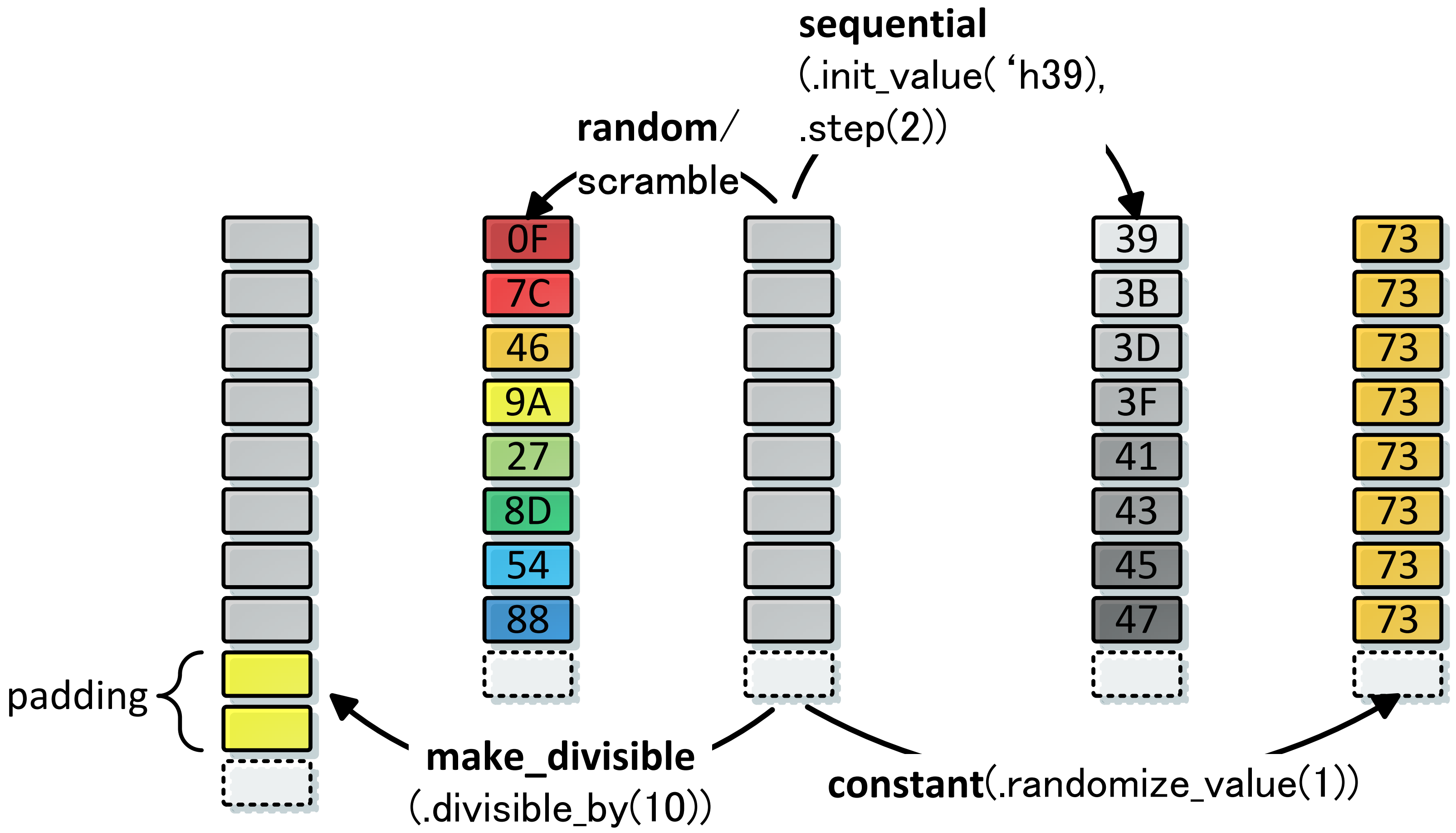
… delegates the search to the “find” function of C++

Data stream and bit stream



A data stream is a dynamic array of packed arrays.

What can you do with the data stream?


sequential
(.init_value(‘h39),
.step(2))
random/
scramble
0F
7C
46
9A
27
8D
54
88
39
3B
3D
3F
41
43
45
47
73
padding
make_divisible
(.divisible_by(10))
constant(.randomize_value(1))