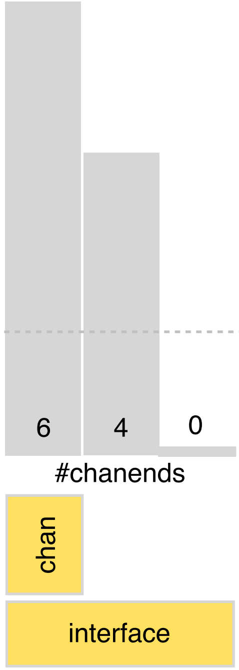
XC code, see http://www.teigfam.net/oyvind/home/technology/141-xc-is-c-plus-x/

XC is C plus X, The combined code: 6 to zero channels by Øyvind Teig (6Jul2018)



```
#if defined TEST_CHAN_AND_COMBINE_TEST

    #include <platform.h>
    #include <stdio.h>
    #include <timer.h> // XS1_TIMER_HZ etc

    #define DEBUG_PRINT_TEST 0
    #if (DEBUG_PRINT_TEST == 1)
        // Uses 1 timer and one chanend (not counted below)
        #define debug_print(fmt, ...) do \
            { if(DEBUG_PRINT_TEST) printf(fmt, __VA_ARGS__); } while (0)
    #else
        #define debug_print(fmt, ...)
    #endif

     [[combinable]]
     void button (chanend c_out) {
        timer t;
        int s;
        t :> s;
        while (1) {
            select {
                case t when timerafter(s) :> void: {
                    c_out <: (s/XS1_TIMER_KHZ); // ms
                    s += XS1_TIMER_HZ;
                    break;
                }
            }
        }
    }

    [[combinable]]
    void handle (chanend c_but[3]) {
        int val;
        while (1) {
            select {
                case c_but[int i] :> val: {
                    debug_print ("handle: from %d val %u\n", i, val);
                    break;
                }
            }
        }
    }

    #define DO_PLACED 1 // 1-4 works
```

chan

```
    int main (void) {
        chan c_but[3]; // Using 6 chanends always
        par {
            #if (DO_PLACED == 1) // Works, also with interface. Uses 4 cores, 4 timers, 6 chanends
                on tile[0].core[0]: handle (c_but);
                par {
                    on tile[0].core[2]: button (c_but[0]);
                    on tile[0].core[3]: button (c_but[1]);
                    on tile[0].core[4]: button (c_but[2]);
                }
            #elif (DO_PLACED == 2) // Works, also with interface. Uses 2 cores, 2 timers, 6 chanends
                on tile[0].core[0]: handle (c_but);
                par {
                    on tile[0].core[1]: button (c_but[0]);
                    on tile[0].core[1]: button (c_but[1]);
                    on tile[0].core[1]: button (c_but[2]);
                }
            #elif (DO_PLACED == 3) // Works, also with interface. Uses 4 cores, 4 timers, 6 chanends
                handle (c_but);
                par {
                    button (c_but[0]);
                    button (c_but[1]);
                    button (c_but[2]);
                }
            #elif (DO_PLACED == 4) // Works, also with interface. Uses 2 cores, 2 timers, 6 chanends
                handle (c_but);
                [[combine]]
                par {
                    button (c_but[0]);
                    button (c_but[1]);
                    button (c_but[2]);
                }
            #elif (DO_PLACED == 5) // Errs, WORKS with interface
                on tile[0].core[0]: handle (c_but);
                                         // ^~~~~ note: other end is used here
                par {
                    on tile[0].core[0]: button (c_but[0]);
                    // ^~~~~ error: `c_but' used between two combined tasks
                    on tile[0].core[1]: button (c_but[1]);
                    on tile[0].core[1]: button (c_but[2]);
                }
            #elif (DO_PLACED == 6) // Errs, WORKS with interface
                 [[combine]]
                 par {
                     handle (c_but);
                     // ^~~~~ note: other end is used here
                     button (c_but[0]);
                          // ^~~~~ error: `c_but' used between two combined tasks
                     button (c_but[1]);
                          // ^~~~~ error: `c_but' used between two combined tasks
                     button (c_but[2]);
                          // ^~~~~ error: `c_but' used between two combined tasks
                 }
            #elif (DO_PLACED == 7) // Errs as with interface
                on tile[0].core[0]: handle (c_but);
                [[combine]]
                par {
                // ^~~~~ error: cannot apply [[combine]] to multi-tile par
                    on tile[0].core[2]: button (c_but[0]);
                    on tile[0].core[3]: button (c_but[1]);
                    on tile[0].core[4]: button (c_but[2]);
                }
            #elif (DO_PLACED == 8)  // Errs as with interface
                on tile[0].core[0]: handle (c_but);
                [[combine]]
                par {
                // ^~~~~ error: cannot apply [[combine]] to multi-tile par
                    button (c_but[0]);
                    // ^~~~~ error: components of multi-tile par must have `on' specifier or call a service
                    button (c_but[1]);
                    // ^~~~~ error: components of multi-tile par must have `on' specifier or call a service
                    button (c_but[2]);
                    // ^~~~~ error: components of multi-tile par must have `on' specifier or call a service
                }
            #else
                // warning: unused variable `c_but' [-Wunused-variable]
            #endif
        }
        return 0;
    }
```

```
#elif defined TEST_INTERFACE_AND_COMBINE_TEST

    #include <platform.h>
    #include <stdio.h>
    #include <timer.h> // XS1_TIMER_HZ etc

    #define DEBUG_PRINT_TEST 1
    #if (DEBUG_PRINT_TEST == 1)
        // Uses 1 timer and one chanend (not counted below)
        #define debug_print(fmt, ...) do \
            { if(DEBUG_PRINT_TEST) printf(fmt, __VA_ARGS__); } while (0)
    #else
        #define debug_print(fmt, ...)
    #endif

    interface ifa {
        void but (int x);
    };

    [[combinable]]
    void button (client interface ifa i_but) {
       timer t;
       int s;
       t :> s;
       while (1) {
           select {
               case t when timerafter(s) :> void: {
                   i_but.but(s/XS1_TIMER_KHZ); // ms
                   s += XS1_TIMER_HZ;
                   break;
               }
           }
       }
    }

    [[combinable]]
    void handle (server interface ifa i_but[3]) {
        while (1) {
            select {
                case i_but[int i].but (int val) : {
                    debug_print ("handle: from %d val %u\n", i, val);
                    break;
                }
            }
        }
    }

    #define DO_PLACED 6 // 1-6 works
```

interface

```
    int main (void) {
        interface ifa i_but[3]; // 6 to zero chanends
        par {
            #if (DO_PLACED == 1) // Works, also with chan. Uses 4 cores, 4 timers, 6 chanends
                on tile[0].core[0]: handle (i_but);
                par {
                    on tile[0].core[2]: button (i_but[0]);
                    on tile[0].core[3]: button (i_but[1]);
                    on tile[0].core[4]: button (i_but[2]);
                }
            #elif (DO_PLACED == 2) // Works, also with chan. Uses 2 cores, 2 timers, 4 chanends
                on tile[0].core[0]: handle (i_but);
                par {
                    on tile[0].core[1]: button (i_but[0]);
                    on tile[0].core[1]: button (i_but[1]);
                    on tile[0].core[1]: button (i_but[2]);
                }
            #elif (DO_PLACED == 3) // Works, also with chan. Uses 4 cores, 4 timers, 6 chanends
                handle (i_but);
                par {
                    button (i_but[0]);
                    button (i_but[1]);
                    button (i_but[2]);
                }
            #elif (DO_PLACED == 4) // Works, also with chan. Uses 2 cores, 2 timers, 4 chanends
                handle (i_but);
                [[combine]]
                par {
                    button (i_but[0]);
                    button (i_but[1]);
                    button (i_but[2]);
                }
            #elif (DO_PLACED == 5) // Works, NOT with chan. Uses 2 cores, 2 timers, 4 chanends
                on tile[0].core[0]: handle (i_but);
                par {
                    on tile[0].core[0]: button (i_but[0]);
                    on tile[0].core[1]: button (i_but[1]);
                    on tile[0].core[1]: button (i_but[2]);
                }

            #elif (DO_PLACED == 6) // Works, NOT with chan. Uses 1 core, 1 timer, 0 chanends
                 [[combine]]
                 par {
                     handle (i_but);
                     button (i_but[0]);
                     button (i_but[1]);
                     button (i_but[2]);
                 }

            #elif (DO_PLACED == 7) // Errs as with chan
                on tile[0].core[0]: handle (i_but);
                [[combine]]
                par {
                // ^~~~~ error: cannot apply [[combine]] to multi-tile par
                    on tile[0].core[2]: button (i_but[0]);
                    on tile[0].core[3]: button (i_but[1]);
                    on tile[0].core[4]: button (i_but[2]);
                }
            #elif (DO_PLACED == 8)  // Errs as with chan
                on tile[0].core[0]: handle (i_but);
                [[combine]]
                par {
                // ^~~~~ error: cannot apply [[combine]] to multi-tile par
                    button (i_but[0]);
                    // ^~~~~ error: components of multi-tile par must have `on' specifier or call a service
                    button (i_but[1]);
                    // ^~~~~ error: components of multi-tile par must have `on' specifier or call a service
                    button (i_but[2]);
                    // ^~~~~ error: components of multi-tile par must have `on' specifier or call a service
                }
            #else
                    // warning: unused variable `i_but' [-Wunused-variable]
            #endif
        }
        return 0;
    }
```