

Learning the Portable Stimulus Standard

Tom Fitzpatrick

CEO Big Fish EDA Consulting

On behalf of the Accellera Systems Initiative

and the Accellera Portable Stimulus Working Group



Test Realization

 Session 12

What You Will Learn

- ▶ Introducing **exec body** blocks
- ▶ Target templates
- ▶ Foreign procedures
- ▶ Procedural **exec** blocks
- ▶ Exec **header** and **declaration** blocks

::: Realization: The Rubber Meets the Road



The Abstract Model must be retargeted to different implementation platforms

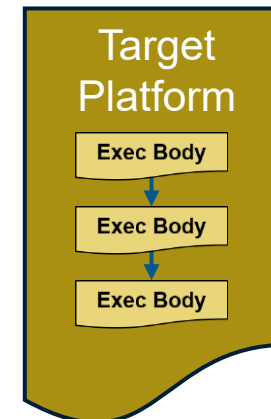
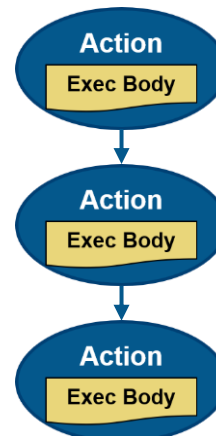
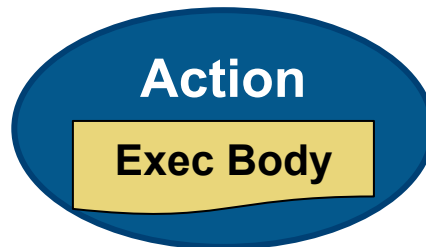


Atomic Actions → target code

- Target code modeled in *exec body* blocks

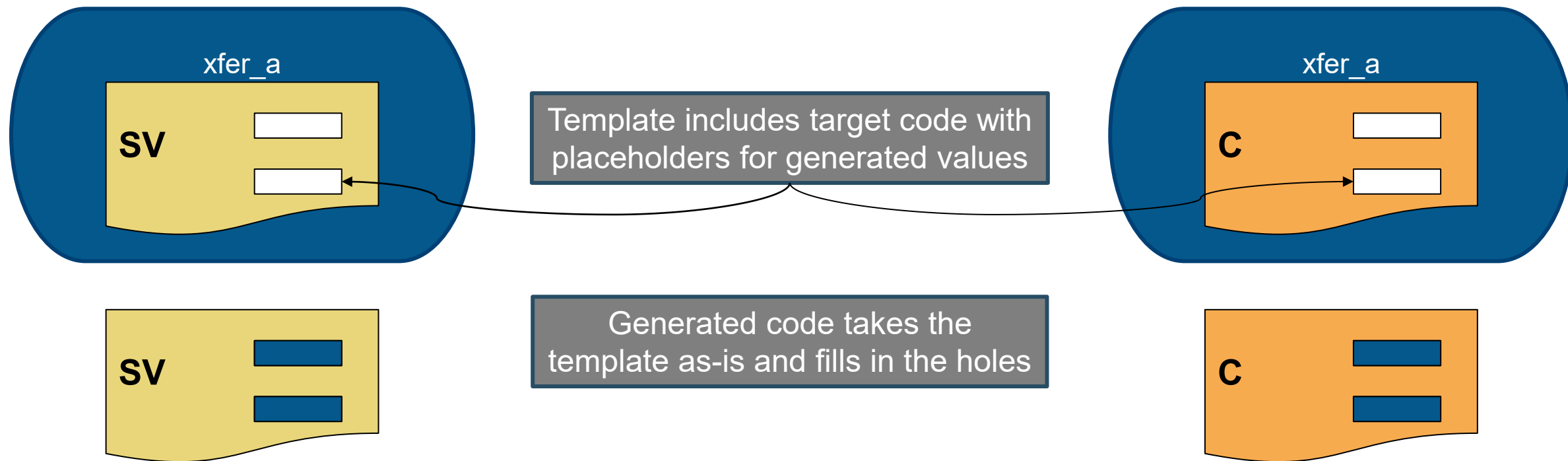


Generator assembles target code according to *Activity* schedule



Exec Body Blocks Define Target Code

Target Templates Define 1:1 Mapping



One exec body for each language-specific implementation

::: PSS Realization: Target Templates



String literal is not recognized nor syntax-highlighted by a simple editor

Model values passed in via `{{moustache}}` notation

Need a separate exec block for each target implementation

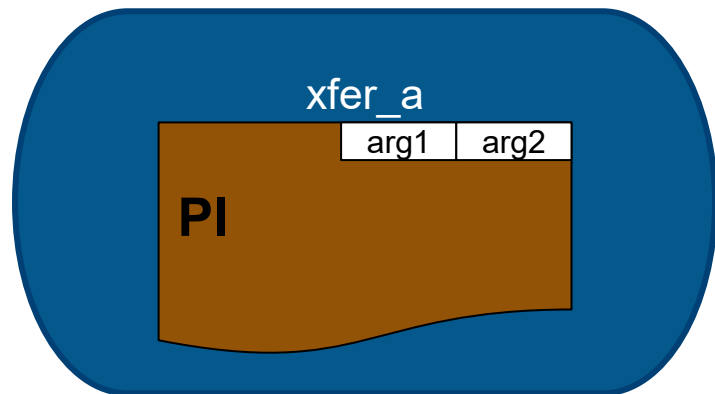
```
extend action dma_c::xfer_a {  
  exec body C = ""  
    dma_program({{channel.instance_id}},{{src.size}},  
                {{src.addr}},{{dst.addr}});  
    dma_activate({{channel.instance_id}});  
    while (!dma_xfer_done({{channel.instance_id}}))  
      yield();  
    "";  
}
```

```
extend action dma_c::xfer_a {  
  exec body SV = ""  
    dma_program({{channel.instance_id}},{{src.size}},  
                {{src.addr}},{{dst.addr}});  
    dma_activate({{channel.instance_id}});  
    while (!dma_xfer_done({{channel.instance_id}}))  
      #0;  
    "";  
}
```

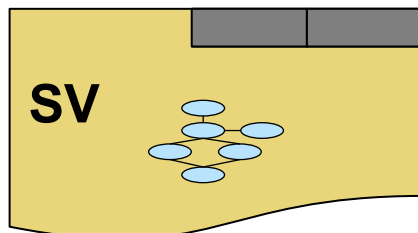
One exec body for each language-specific implementation

Exec Body Blocks Define Target Code

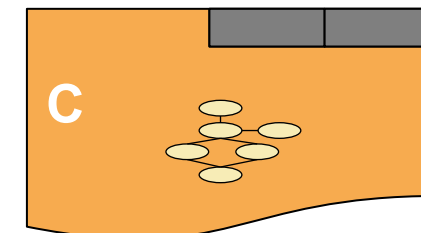
Procedural Interface Isolates Exec from Target Language



Procedural Interface defines functions that map to functions(/tasks) in the target language.
Values passed as arguments



Still have to define possibly complex algorithms in each target language



One exec body for each action

::: PSS Realization: Foreign Procedures



Method implementation determined by which package is imported

```
extend action dma_c::dma_xfer {  
  exec body {  
    do_dma(channel.instance_id,src.size,  
            src.addr,dst.addr);  
  }  
}
```

Still need to implement the algorithm multiple times

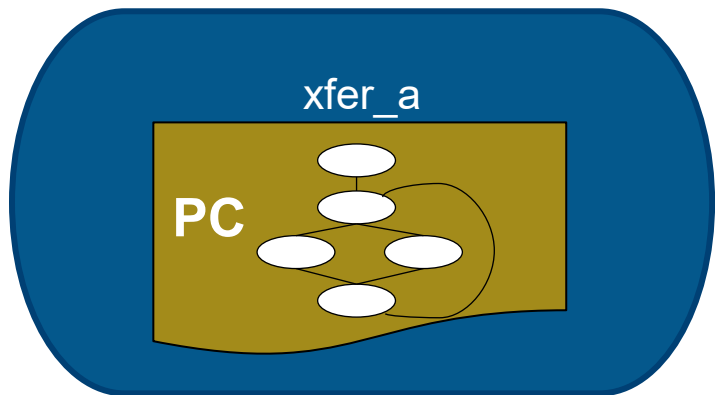
```
package dma_sv_func_pkg {  
  function void do_dma(int channel, int size,  
                       int src_addr, int dst_addr);  
  import target SV function do_dma;  
}  
  
task automatic do_dma(int channel, int size,  
                      int src_addr, int dst_addr);  
  dma_program(channel, size, src_addr, dst_addr);  
  dma_activate(channel);  
  while (!dma_xfer_done(channel))  
    #0;  
endtask
```

```
package dma_c_func_pkg {  
  function void do_dma(int channel, int size,  
                       int src_addr, int dst_addr);  
  import target C function do_dma;  
}  
  
void do_dma(int channel, int size,  
            int src_addr, int dst_addr) {  
  dma_program(channel,size,src_addr,dst_addr);  
  dma_activate(channel);  
  while (!dma_xfer_done(channel))  
    yield();  
}
```

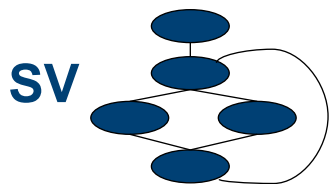
One exec body for each action

Exec Body Blocks Define Target Code

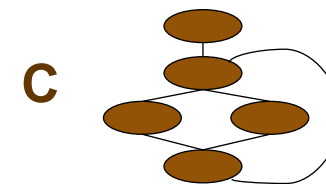
Procedural Constructs Move Complex Flow Control to PSS



Algorithm is specified in exec block
Imported methods called accordingly



Complex control flow generated from PSS
Language-specific code is much simpler



Procedural Constructs provide maximum reuse by
simplifying the migration between languages

::: PSS Realization: Procedural Execs



Code is checked for syntax, type correctness, and all static semantic rules

```
extend action dma_c::dma_xfer {  
  exec body {  
    dma_program(channel.instance_id,src.size,  
                src.addr,dst.addr,INCR);  
    dma_activate(channel.instance_id);  
    while (!dma_xfer_done(channel.instance_id))  
      yield;  
  }  
}
```

Each procedural (sub)method must still be implemented multiple times

```
package dma_sv_funcs_pkg {  
  function void dma_program(int channel, int size,  
                             int src_addr, int dst_addr);  
  import target SV function dma_program;  
  function void dma_activate(int channel);  
  import target SV function dma_activate;  
  function bit dma_xfer_done(int channel);  
  import target SV function dma_xfer_done;  
}
```

```
package dma_c_funcs_pkg {  
  function void dma_program(int channel, int size,  
                             int src_addr, int dst_addr);  
  import target C function dma_program;  
  function void dma_activate(int channel);  
  import target C function dma_activate;  
  function bit dma_xfer_done(int channel);  
  import target C function dma_xfer_done;  
}
```

Procedural Constructs provide maximum reuse by simplifying the migration between languages

::: PSS Realization: Procedural Execs



Use different files to
differentiate package contents

```
import dma_funcs_pkg::*;
extend action dma_c::dma_xfer {
  exec body {
    dma_program(channel.instance_id, src.size,
                src.addr, dst.addr, INCR);
    dma_activate(channel.instance_id);
    while (!dma_xfer_done(channel.instance_id))
      yield;
  }
}
```

dma_sv_funcs_pkg.pss

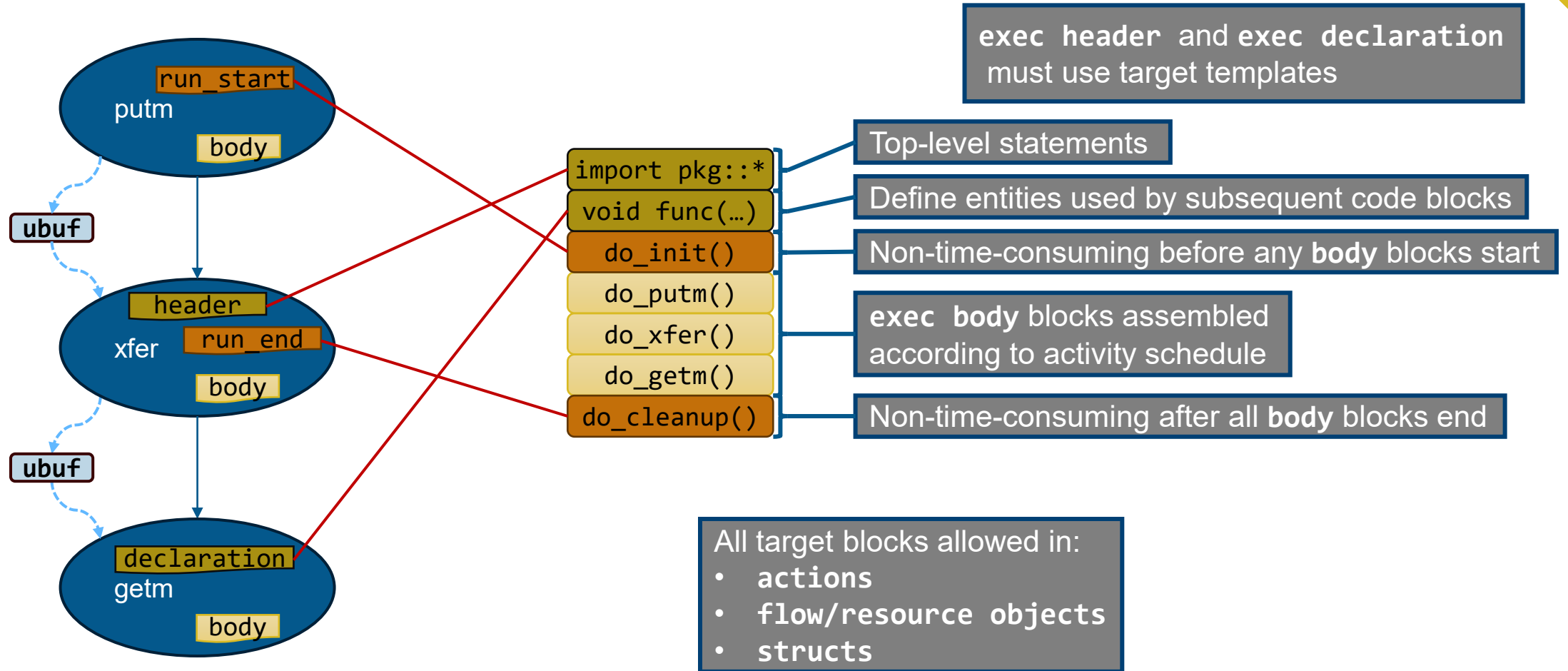
```
package dma_funcs_pkg {
  function void dma_program(int channel, int size,
                           int src_addr, int dst_addr);
  import target SV function dma_program;
  function void dma_activate(int channel);
  import target SV function dma_activate;
  function bit dma_xfer_done(int channel);
  import target SV function dma_xfer_done;
}
```

dma_c_funcs_pkg.pss

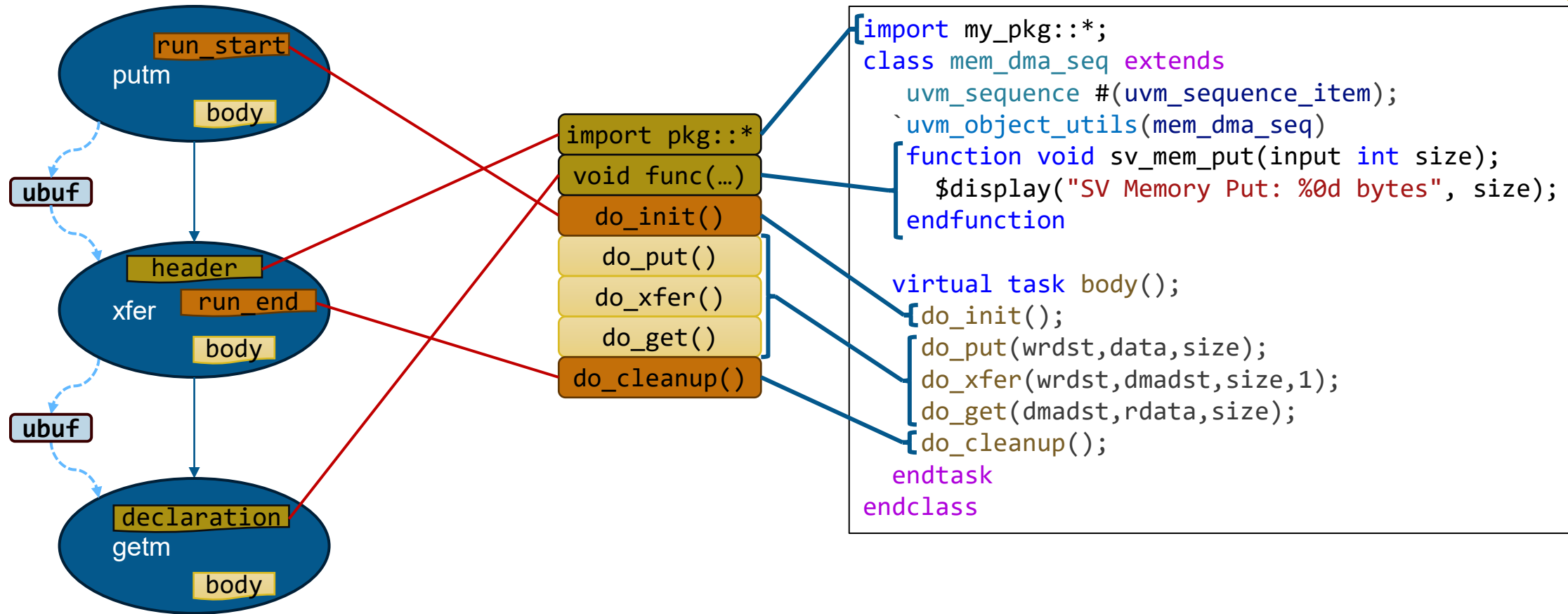
```
package dma_funcs_pkg {
  function void dma_program(int channel, int size,
                           int src_addr, int dst_addr);
  import target C function dma_program;
  function void dma_activate(int channel);
  import target C function dma_activate;
  function bit dma_xfer_done(int channel);
  import target C function dma_xfer_done;
}
```

Procedural Constructs provide maximum reuse by
simplifying the migration between languages

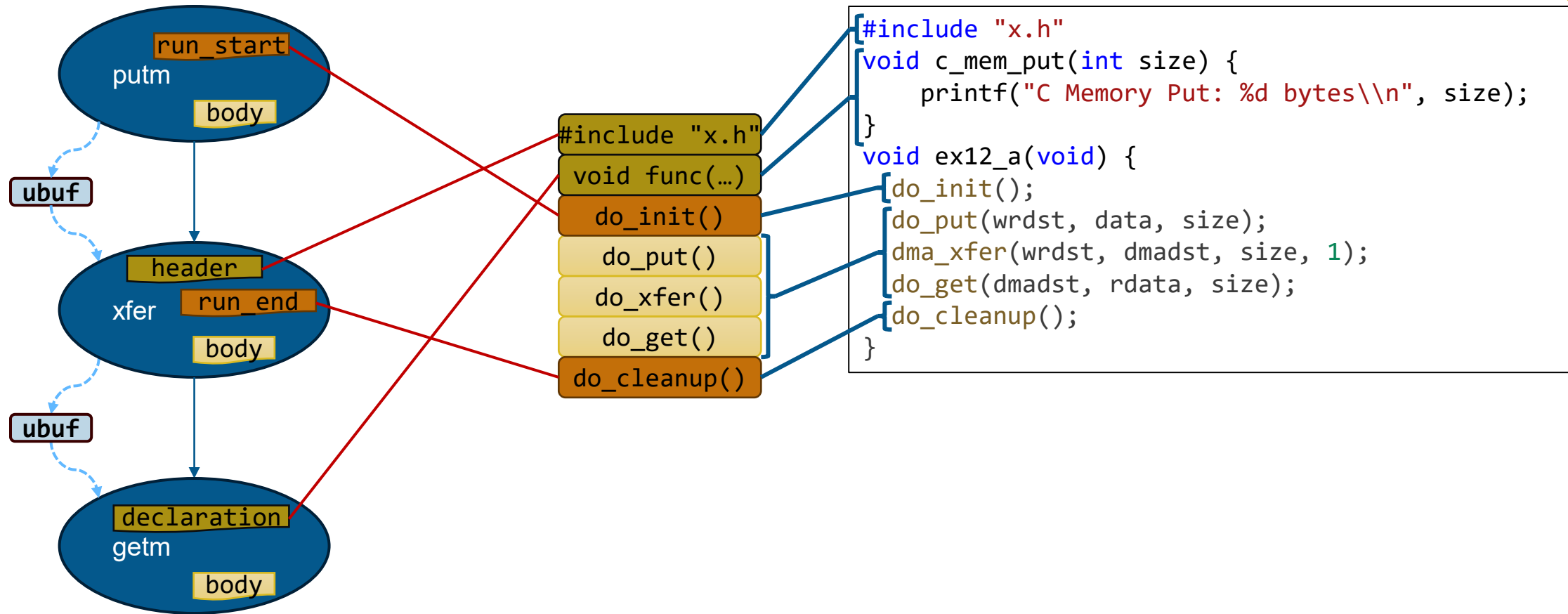
::: Mapping exec Blocks to Target



::: Mapping exec Blocks to Target: SV



::: Mapping exec Blocks to Target: C





Thank You

