

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



Components and Pools

 Session 07

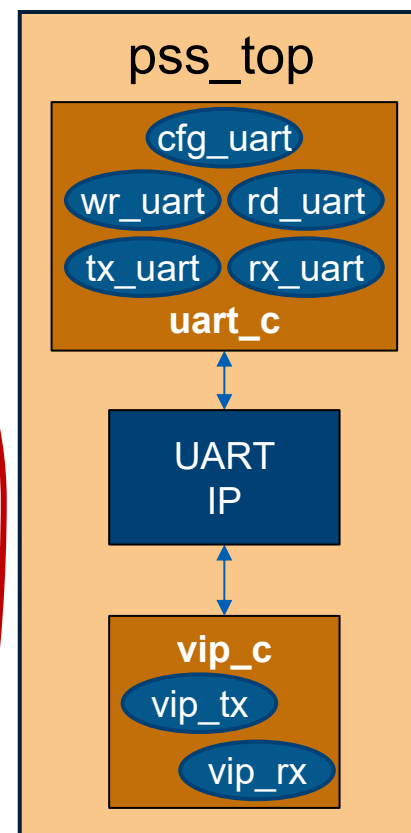
What You Will Learn

- ▶ Organizing PSS actions using components
- ▶ Managing PSS data flow using pools
- ▶ Binding actions to pools
- ▶ Specifying the execution context of an action

::: Organizing Actions: Components

- ▶ Components typically represent functional units of the system:
 - ▶ HW cores / IPs
 - ▶ SW libraries (e.g., Graphical library)
 - ▶ Testbench agents / transactors
- ▶ Structured as an instance tree
- ▶ Top-level component named **pss_top** by default
- ▶ Components serve as *namespaces*

```
component uart_c {  
  action cfg_uart_a {...}  
  action wr_uart_a {...}  
  action rd_uart_a {...}  
  action tx_uart_a {...}  
  action rx_uart_a {...}  
}  
  
component vip_c {  
  action vip_tx_a {...}  
  action vip_rx_a {...}  
}  
  
component pss_top {  
  vip_c vip;  
  uart_c uart;  
  ..  
  action entry_a {  
    activity {  
      do uart_c::wr_uart_a;  
    }  
  }  
}
```



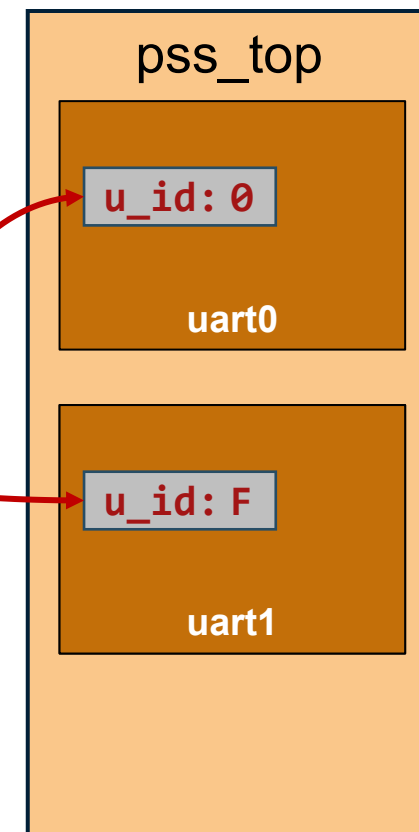
Component Data Fields

- ▶ Components may have *non-random* data fields
 - ▶ Data fields can be initialized at declaration
 - ▶ Fields can also be initialized via **exec init_down/init_up** blocks
 - ▶ **init_down**: top-down
 - ▶ **init_up**: bottom-up
 - ▶ **exec init** blocks override declaration initialization

```
component uart_c {  
    bit[4] u_id = 'hF;  
}
```

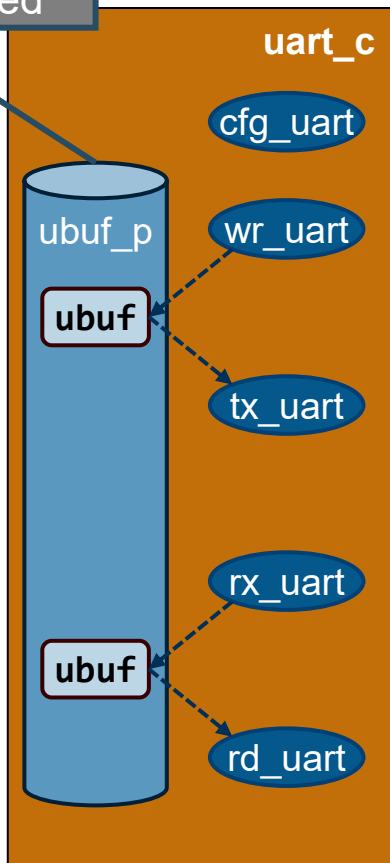
```
component pss_top {  
    uart_c uart0, uart1;
```

```
    exec init_up {  
        uart0.u_id = 0;  
        uart1.u_id = 1;  
    }  
}
```



:::: Data Flow Object Binding & Pools

Buffer/stream pools hold as many objects as needed



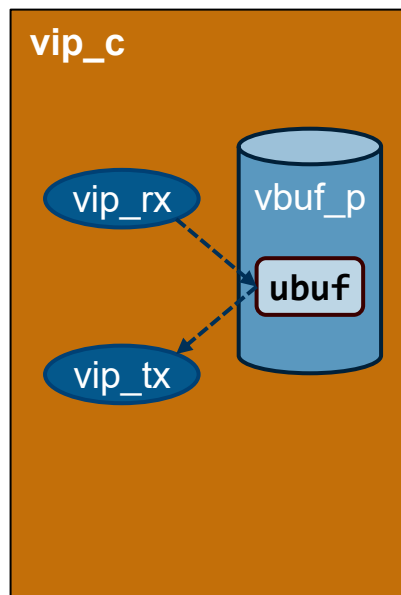
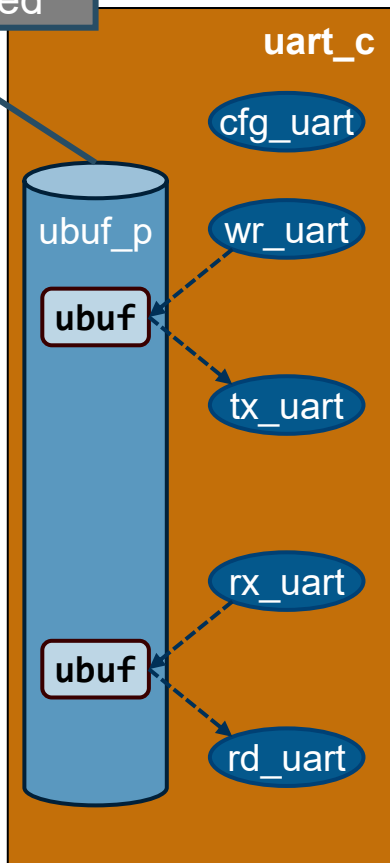
```
component uart_c {  
  action cfg_uart_a {...}  
  action wr_uart_a {output ubuf_b obuf;}  
  action tx_uart_a {input ubuf_b ibuf;}  
}  
  
action rd_uart_a {input ubuf_b ibuf;}  
action rx_uart_a {output ubuf_b obuf;}  
}
```

```
pool ubuf_b ubuf_p;  
bind ubuf_p *;
```

Default binding for all compatible action fields

Data Flow Object Binding & Pools

Buffer/stream pools hold as many objects as needed



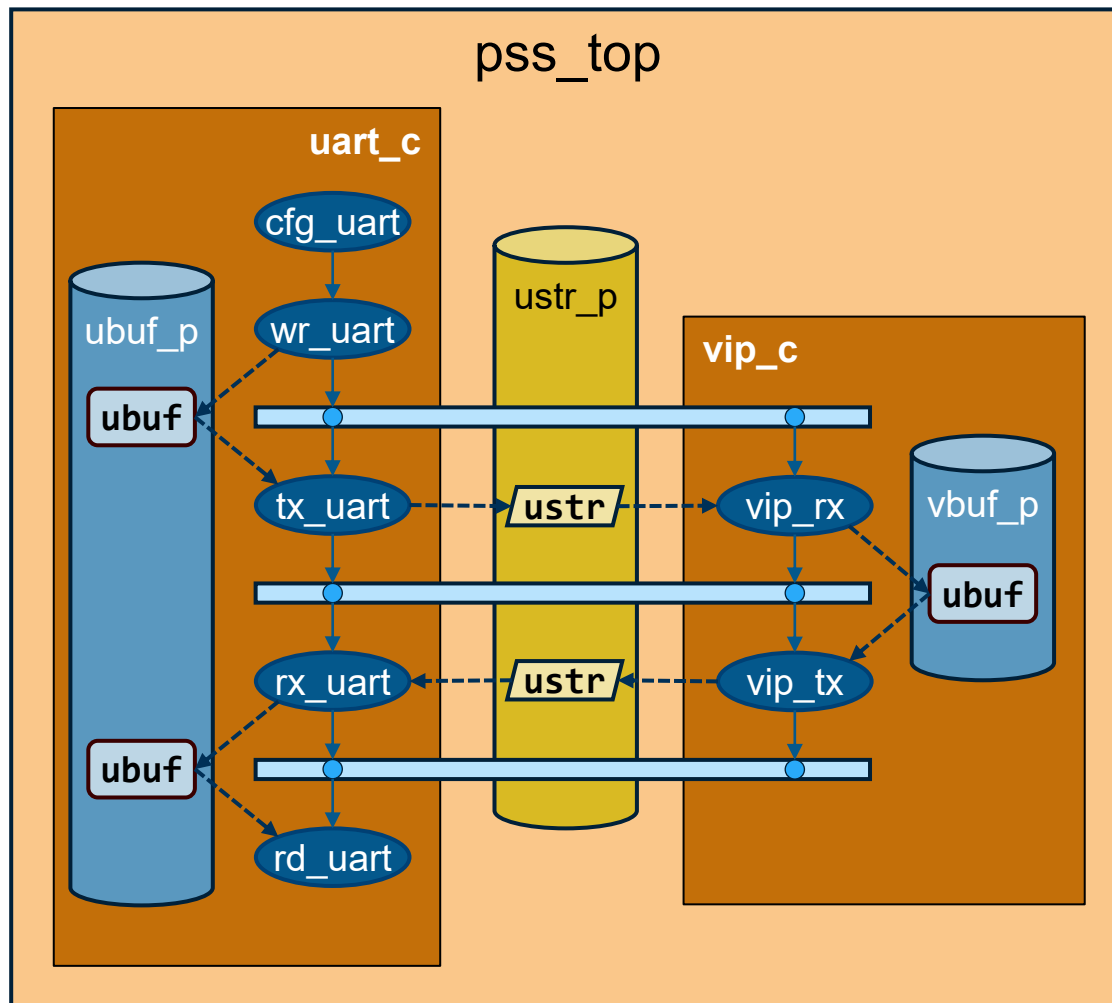
```
component uart_c {  
  action cfg_uart_a {...}  
  action wr_uart_a {output ubuf_b obuf;}  
  action tx_uart_a {input ubuf_b ibuf;}  
}  
  
action rd_uart_a {input ubuf_b ibuf;}  
action rx_uart_a {output ubuf_b obuf;}  
}
```

```
pool ubuf_b ubuf_p;  
bind ubuf_p *;  
}
```

```
component vip_c {  
  action vip_rx_a {output ubuf_b obuf;}  
  action vip_tx_a {input ubuf_b ibuf;}  
}
```

```
pool ubuf_b vbuf_p;  
bind vbuf_p *;  
}
```

Data Flow Object Binding & Pools



```
component uart_c {
  action cfg_uart_a {...}
  action wr_uart_a {output ubuf_b obuf;}
  action tx_uart_a {input ubuf_b ibuf;
                    output ustr_s ostr;}

  action rd_uart_a {input ubuf_b ibuf;}
  action rx_uart_a {output ubuf_b obuf;
                    input ustr_s istr;}
}
```

```
pool ubuf_b ubuf_p;
bind ubuf_p *;
```

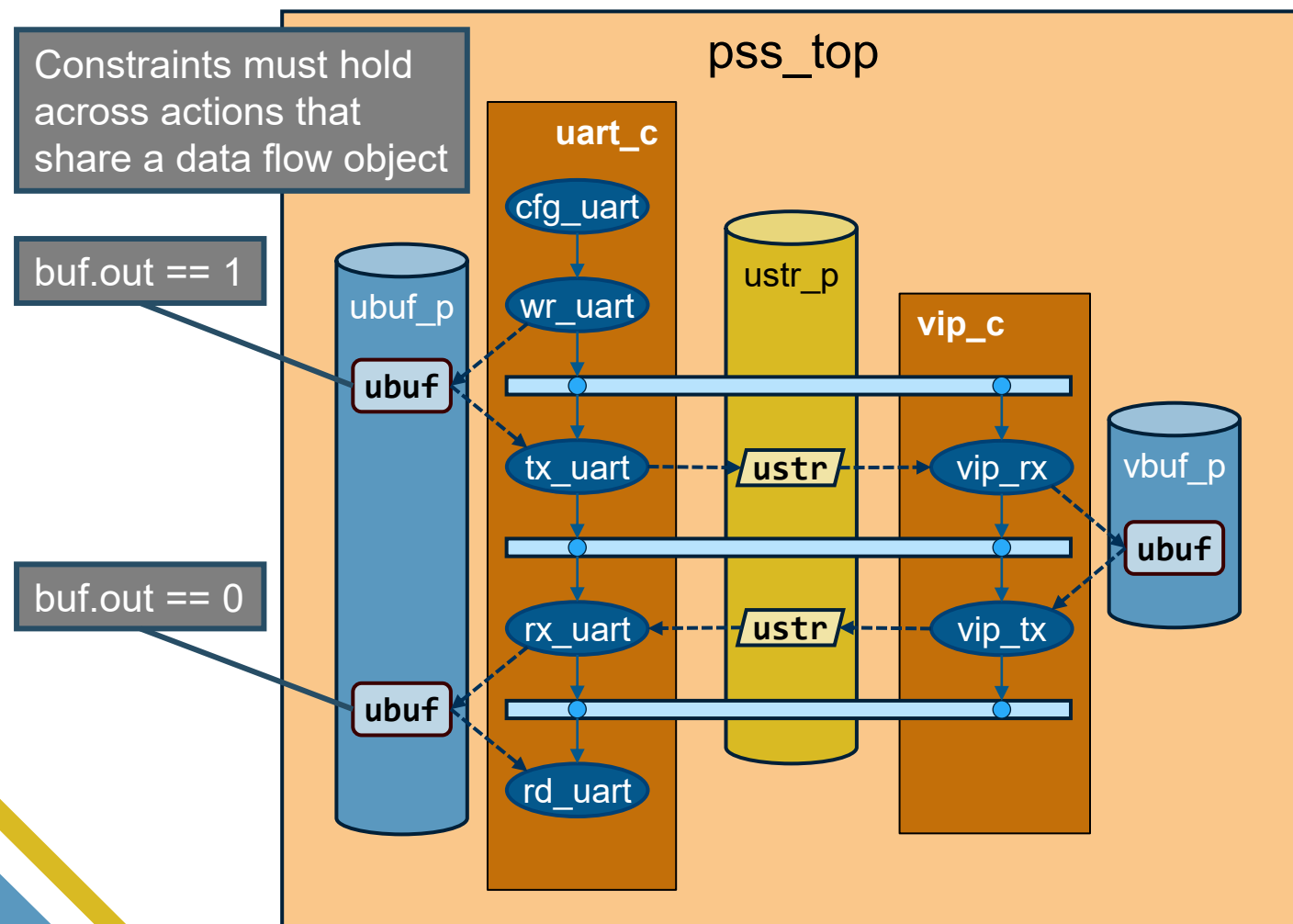
```
component vip_c {
  action vip_rx_a {output ubuf_b obuf;
                   input ustr_s istr;}
  action vip_tx_a {input ubuf_b ibuf;
                   output ustr_s ostr;}
}
```

```
pool ubuf_b vbuf_p;
bind vbuf_p *;
```

```
component pss_top {
  uart_c uart;
  vip_c vip;
}
```

```
pool ustr_s ustr_p;
bind ustr_p *;
```

Flow Object Binding & Constraints



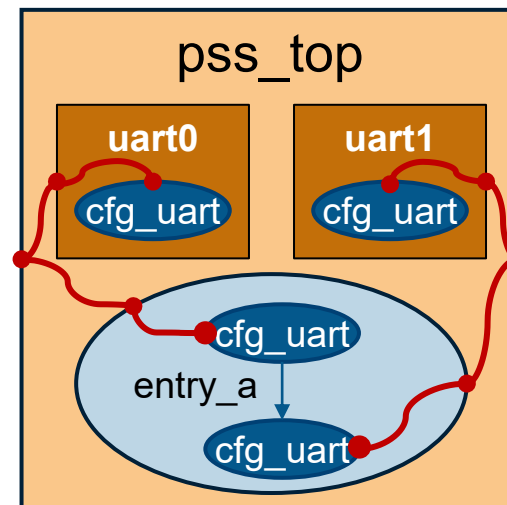
```
component uart_c {  
  action cfg_uart_a {...}  
  action wr_uart_a {output ubuf_b obuf;  
    constraint obuf.out == 1;  
  }  
  action tx_uart_a {input ubuf_b ibuf;  
    output ustr_s ostr;  
    constraint ibuf.out == 1;  
  }  
  action rd_uart_a {input ubuf_b ibuf;  
    constraint ibuf.out == 0;  
  }  
  action rx_uart_a {output ubuf_b obuf;  
    input ustr_s istr;  
    constraint obuf.out == 0;  
  }  
}
```

... The comp Attribute

- ▶ Every action is traversed in a *component context*
- ▶ Pointed to by the built-in **comp** field
- ▶ **this** points to the current (parent) action

```
component pss_top {  
  import uart data_pkg::*;  
  uart_c uart0, uart1;  
  
  pool ucfg_st ucfg_p;  
  bind ucfg_p *;  
  
  action entry_a {  
    activity {  
      do uart c:cfg_uart a with {comp == this.comp.uart0;};  
      do uart_c::cfg_uart_a with {comp == this.comp.uart1;};  
    }  
  }  
}
```

Component type is used as *namespace* for anonymous action traversals





Thank You

