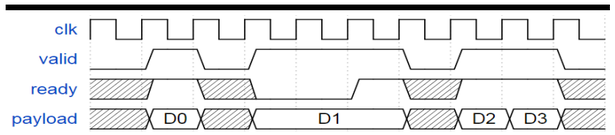
SpinalHDL CheatSheet – Lib

Stream



Interface	valid, ready, payload
Example	val myStream = Stream(Bits(32 bits)) val myInput = master Stream(UInt(3 bits))
Connection	
slave << master master >> slave	Connect two streams together
slave <-< master master >-> slave	Connect with a register stage (1 latency). Equivalent to m2sPipe()
slave <-/< master master >-/> slave	Bandwith divided by 2. Equivalent to s2mPipe.m2sPipe
slave </< master master >/> slave	Connect with a register stage + mux (0 latency). Equivalent to s2mPipe()
Function	.haltWhen(cond), .throwWhen(cond), .continueWhen(cond), .takeWhen(cond), .queue(size), .fire, .stall, .halfPipe(), .stage(), .translateFrom(T)(dataAssignment)
Fifo	StreamFifo(...), StreamFifoCC(...), StreamCCByToggle(...), StreamFifoLowLatency(...)
Arbitrer	val arbitrer = StreamArbiterFactory.roundRobin.noLock. onArgs(streamA, streamB, streamC) ***Arbitration*** *: lowerFirst, roundRobin, sequentialOrder* ***Lock*** *: noLock, transactionLock, fragmentLock*
Fork	val fork = new StreamFork(T, 2)
Dispatcher	StreamDispatcherSequenctial()
Mux	val outStream = StreamMux(UInt, Seq/Vec[Stream[T])
Demux	val demux = StreamDemux(T, portCount: Int)
Join	val wJoin = StreamJoin.arg(bus.aw, bus.w)

Flow

Interface	valid, payload
Example	val myFlow = Flow(Bits(32 bits)) val myInput = slave Flow(UInt(3 bits))
Connection	
slave << master master >> slave	Connect two flows together
s <-< m , m >-> s	Connect with a register stage
Function	.throwWhen(cond), .toReg(), .fire, .toStream, .takeWhen(cond), .translateFrom(T)(dataAssignement), .push()
Fifo	FlowCCByToggle

Fragment

Interface	last, payload
Example	val myStream = Stream(Fragment(T)) val myFlow = Flow(Fragment(T))
Function	.first, .tail, .isFirst, .isTail, .insertHeader(T)

State Machine

Style A

```
val sm = new StateMachine{
  always{
    when(cond){ goto(s1)}
  }
  val s1: State = new Sate with EntryPoint{
    onEntry{}
    whenIsActive{ when(cond) { goto(s2) } }
    onExit{}
  }
  val s2: State = new State{
    whenIsActive{ goto(s1) }
  }
}
```

Style B

```
Val sm = new StateMachine{
  val s1 = new State with EntryPoint
  val s2 = new State
  always{
    when(cond){ goto(s1)}
  }
  s1
    .onEntry()
    .whenIsActive{ goto(s2) }
    .onExit()
  s2.whenIsActive{ goto(s1)}
}
```

Delay	new StateDelay(40){ whencompleted{...}}
Inner SM	new StateFsm(fsm=internalFsm()){whenCompleted{...}}
Parallel SM	new StateParallelFsm(fsmA(), fsmB()){whenCompleted{...}}

Bus Slave Factory

Primitive	.read(), .write(), .readAndWrite(), .onWrite(), .onRead() .isWriting(), isReading(), .readMultiWord(), .writeMultiWord(), .readAndWriteMultiWord(), factory.read(mySignal, address = 0x00)
Create	.createWriteOnly(), .createReadOnly(), .createReadAndWrite(), .createReadAndClearOnSet(), .readAndClearOnSet(), .clearOnSet(), .createAndDriveFlow(), .createWriteMultiWord(), .createReadMultiWord(), .createWriteAndReadMultiWord(), val reg = factory.createWriteOnly(UInt(2 bits), 0x00)
Drive	.drive(), .driveAndRead(), .driveMultiWord(), .driveAndReadMultiWord(), .driveFlow() factory.drive(uart.io.config.frame, 0x10)
Misc	.readStreamNonBlocking(), .doBitsAccumulationAndClearOnRead(), .multiCycleRead(), .readAddress, .writeAddress, .readSyncMemWordAligned(), .writeMemWordAligned()

```
class AvalonUartCtrl(...)   extends Component{
  val io = new Bundle{
      val bus  = slave(AvalonMM(....))
      val uart = master(Uart())
   }
   val uartCtrl = new UartCtrl(uartCtrlConfig)
  io.uart <> uartCtrl.io.uart
  val busCtrl = AvalonMMSlaveFactory(io.bus)
  busCtrl.driveAndRead(uartCtrl.io.config.clockDivider,address = 0)
  busCtrl.driveAndRead(uartCtrl.io.config.frame,address = 4)
  ...
}
```

Utils

Delay(x, c)	Delay x of c cycle
fromGray/toGray(x: Uint)	Return the gray value
Reverse(T)	Reverse all bits
OHToUInt()	One hot number to UInt
MuxOH()	Mux for One hot number
MajorityVote()	True if number of bit set is > x.size
LatencyAnalysis(Node*)	Return the length of the path
History(T, len)	Return a vector of len element of T
EndiannessSwap()	Endianness swap
CountOne()	Return the number of bit set
BufferCC(T)	Synchronized with the current clock domain by using 2 flip flop
Counter()	Counter
Timeout(10 ms)	Timeout
NoData	Empty bundle

Lib

Bus	
AhbLite3	Arbiter, Decoder, Interconnet, AhbLite2<->Apb3, Rom, Ram, SlaveFactory
Apb3	Decoder, GPIO, Router, SlaveFactory
Axi4	Arbiter, Crossbar, Decoder, Axi4<->Apb3
AxiLite4	SlaveFactory
Avalon	SlaveFactory
AsyncMemoryBus	SlaveFactory
PipelinedMemoryBus	-
Com	I2C, Jtag, SPI, UART
Graphic	VGA
Math	Divider
Misc	InterruptCtrl, PDM, Timer, Prescaler
Debugger	SystemDebugger
EDA	Xilinx, Microsemi, Altera