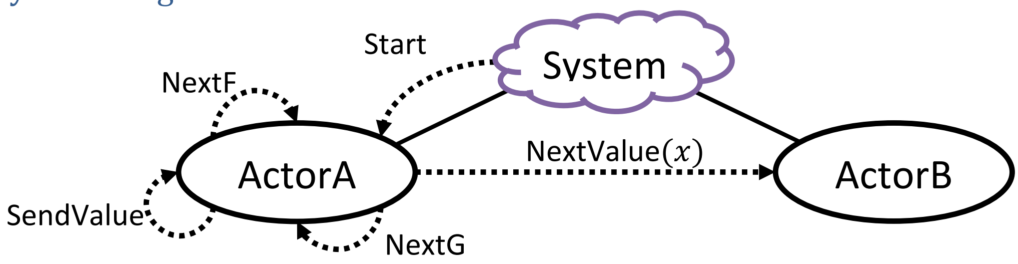
CPS2001
-
Assignment 4

Full name: Miguel Dingli
I.D: 49997M
Course: B.Sc. (Hons) in Computing Science

Table of Contents

Task A – Cyclic Functions

1.1 Actor System Diagram



Note: in the actor system diagram, **System** serves as the implicit actor that is the creator of ActorA and ActorB and the sender of the first message, *Start*, to ActorA.

1.2 Message Flow Excerpt

The following is a message flow excerpt in the form of a list of messages sent/received, starting from the first message send to ActorA up to when the fourth value in the sequence, 666, is generated:

	Messages sent/received	ActorA state	ActorB state
1.	**System** sends **Start** to **ActorA**	/	Nil
2.	**ActorA** receives **Start** from **System** and… …sends **SendValue** to self …sends **NextF** to self	0	Nil
3.	**ActorA** receives **SendValue** from self and… …sends x to **ActorB**	…	Nil
4.	**ActorB** receives x from **ActorA**	…	List(0)
5.	**ActorA** receives **NextF** from self and… …sends **SendValue** to self …sends **NextG** to self	1000	…
6.	**ActorA** receives **SendValue** from self and… …sends x to **ActorB**	…	…
7.	**ActorB** receives x from **ActorA**	…	List(0, 1000)
8.	**ActorA** receives **NextG** from self and… …sends **SendValue** to self …sends **NextF** to self	333	…
9.	**ActorA** receives **SendValue** from self and… …sends x to **ActorB**	…	…
10.	**ActorB** receives x from **ActorA**	…	Nil
11.	**ActorA** receives **NextF** from self and… …sends **SendValue** to self …sends **NextG** to self	666	…

Task B – Circuit Simulator

2.1 Assumptions

The following is a list of assumptions taken:

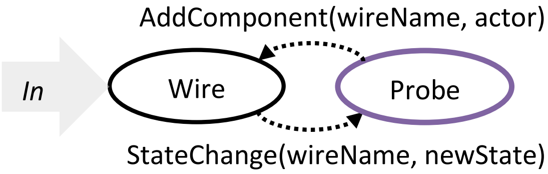
- It is not required that when a component is subscribed to a wire, it is initialized to the current state of the wire. State of components changes when the input wire/s are sent a StateChange message.
- No component will subscribe to the same wire more than once, since the component's ActorRef would have to be entered more than once in the Map as a key, which violates the uniqueness of the keys.
- It is not required that all actors should be explicitly given a name. Actors created by component actors (i.e. by OrAlt, HalfAdder, FullAdder, Demux, and Demux2) were not given an explicit name, so as to avoid problems of duplicate actor names.

2.2 Actor System Diagram

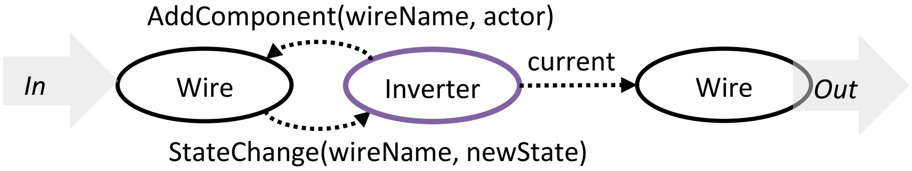
The following is a list of considerations for the diagrams:

- By default, it is assumed that actors have a *'System'* parent actor, unless otherwise indicated.
- **Purple** actors represent the main actor of a component, created by the *System* actor.
- **Blue** actors represent general actors, created by the component's main (**purple**) actor.
- In some cases, to make diagrams clearer, parent-child relationships were not visualised and are instead assumed from the definition of the blue actors (i.e. their parent is the purple actor).
- It is assumed that input Wire actors receive boolean messages from the *outside world*.
- Two *Wire* actors on top of each other means that the indicated message types going out of or into any of the two actors can be sent from and received by both actors involved.
- Finally, the following will be assumed from the OrAlt component, onwards:
    - AC = AddComponent(wireName, actor)
    - SC = StateChange(wireName, newState)
    - C = current: Boolean

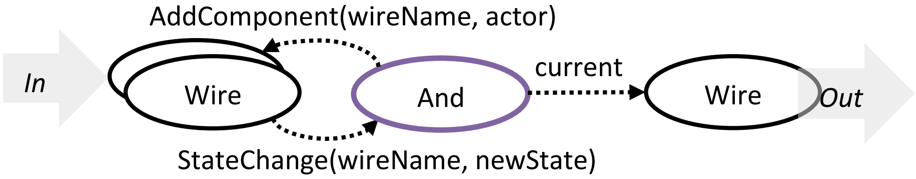
2.2.1 Probe Diagram


AddComponent(wireName, actor)
In
Wire
Probe
StateChange(wireName, newState)

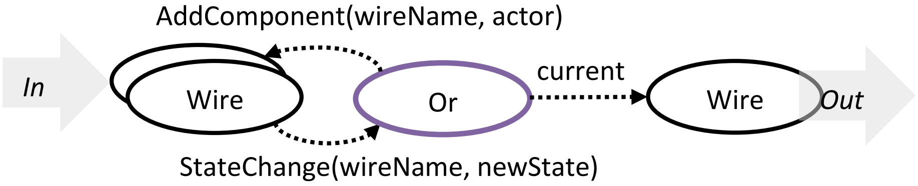
2.2.2 Inverter Diagram


AddComponent(wireName, actor)
In
Wire
Inverter
current
Wire
Out
StateChange(wireName, newState)

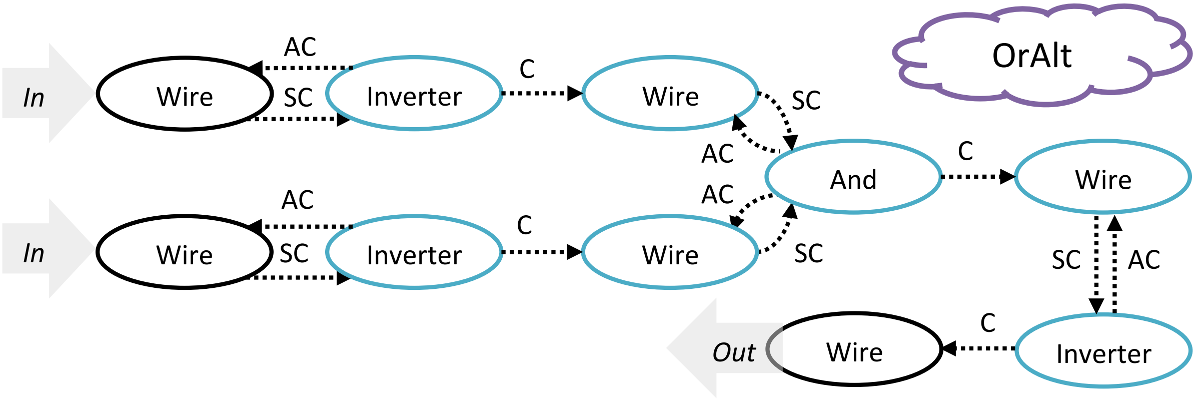
2.2.3 And Diagram


AddComponent(wireName, actor)
In
Wire
And
current
Wire
Out
StateChange(wireName, newState)

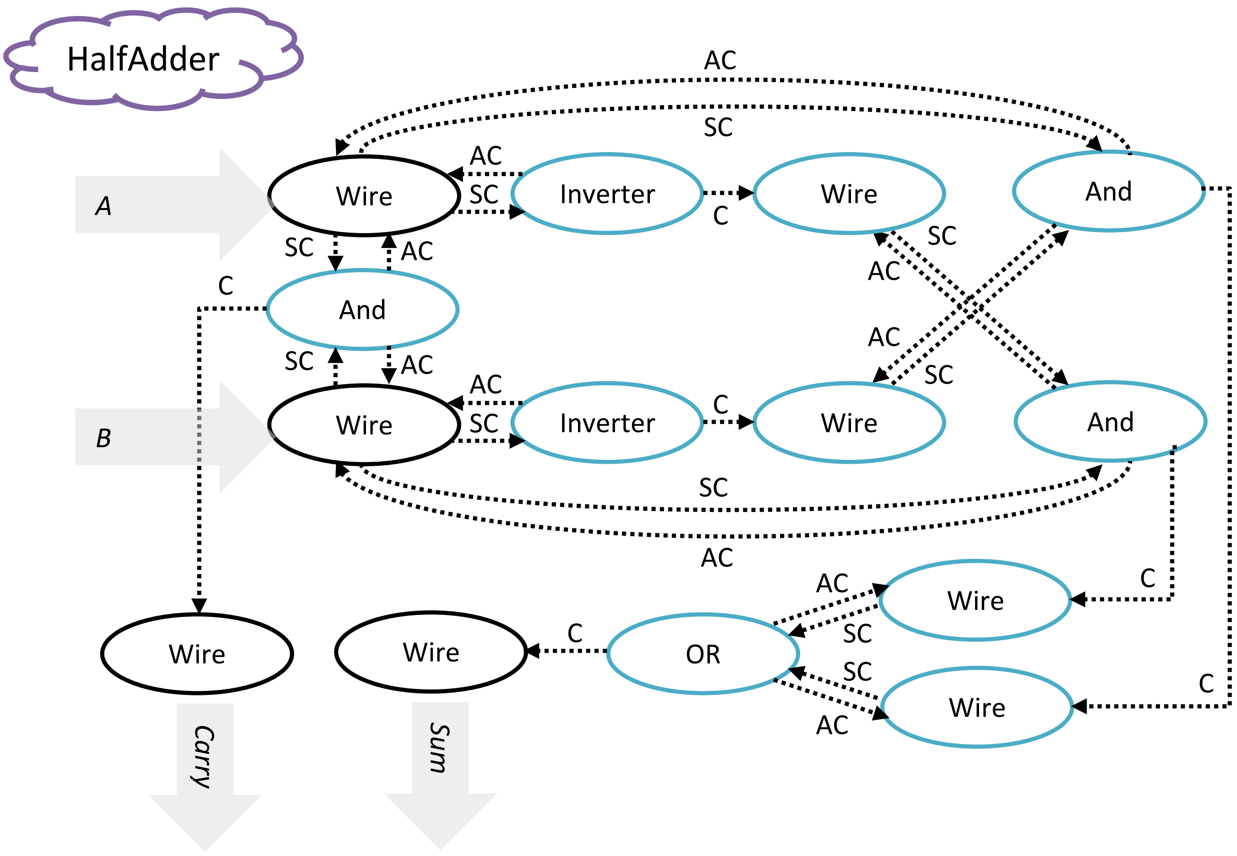
2.2.4 Or Diagram


AddComponent(wireName, actor)
In
Wire
Or
current
Wire
Out
StateChange(wireName, newState)

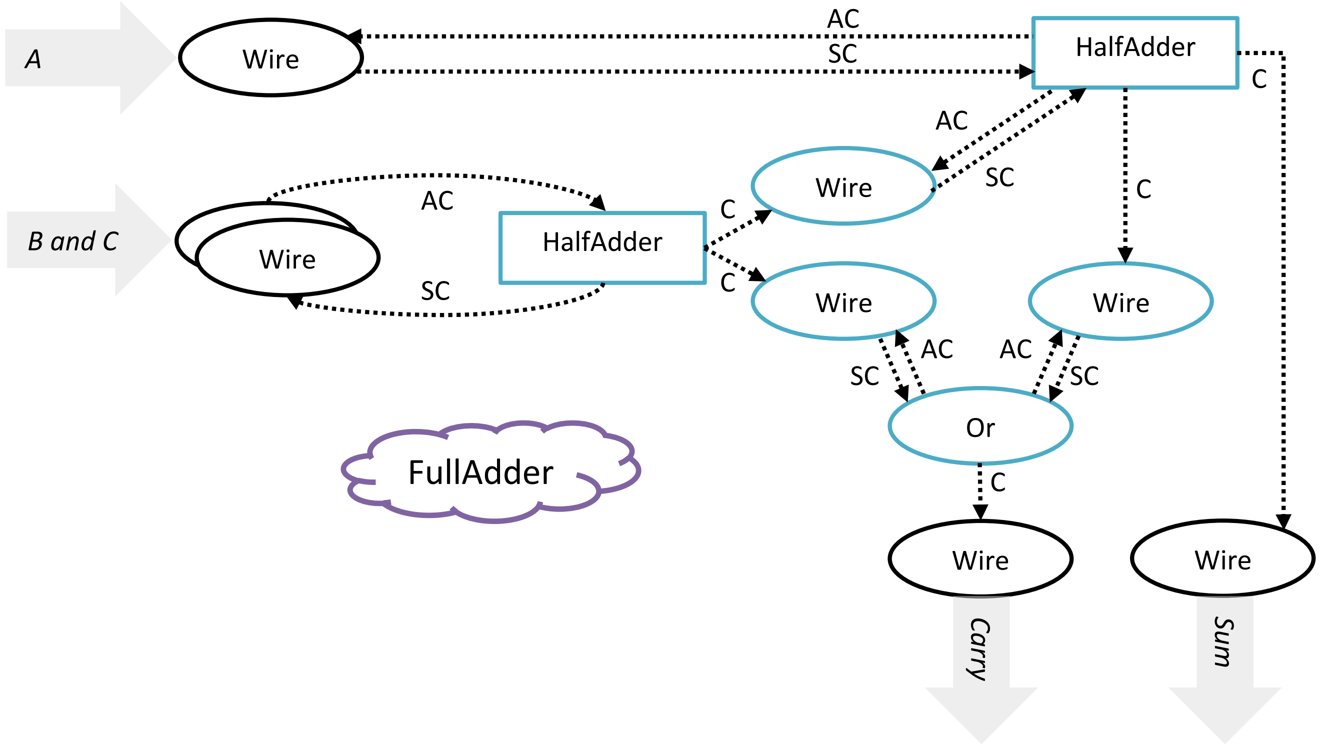
2.2.5 OrAlt Diagram



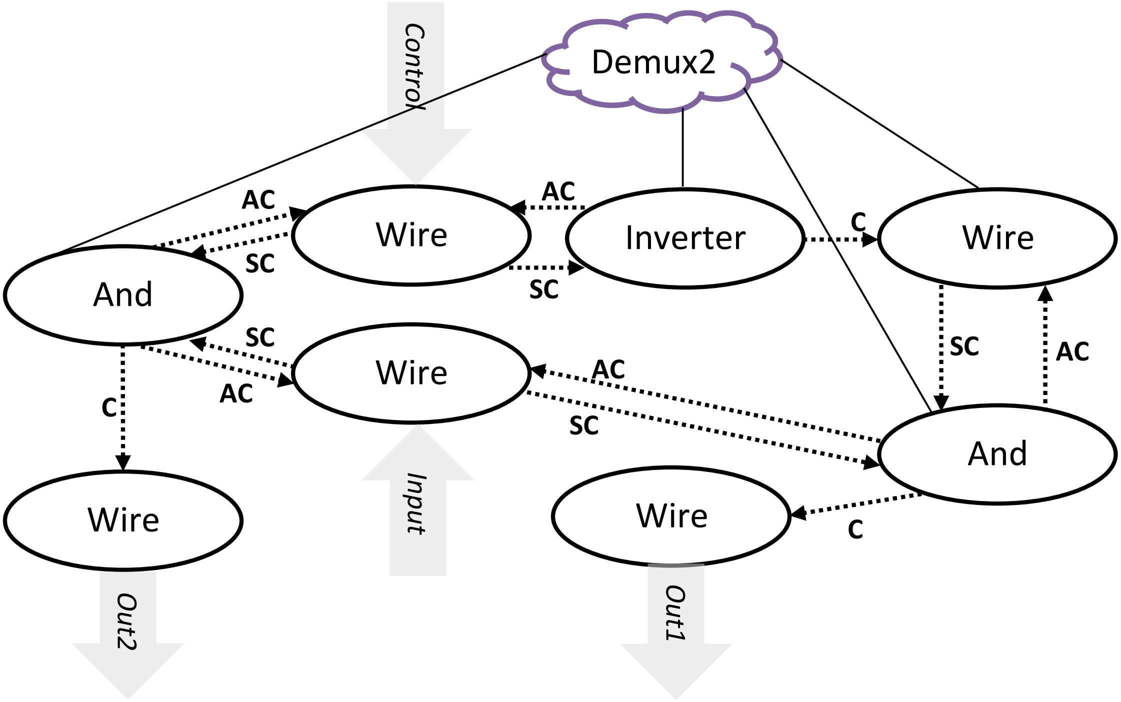
2.2.6 Half Adder Diagram



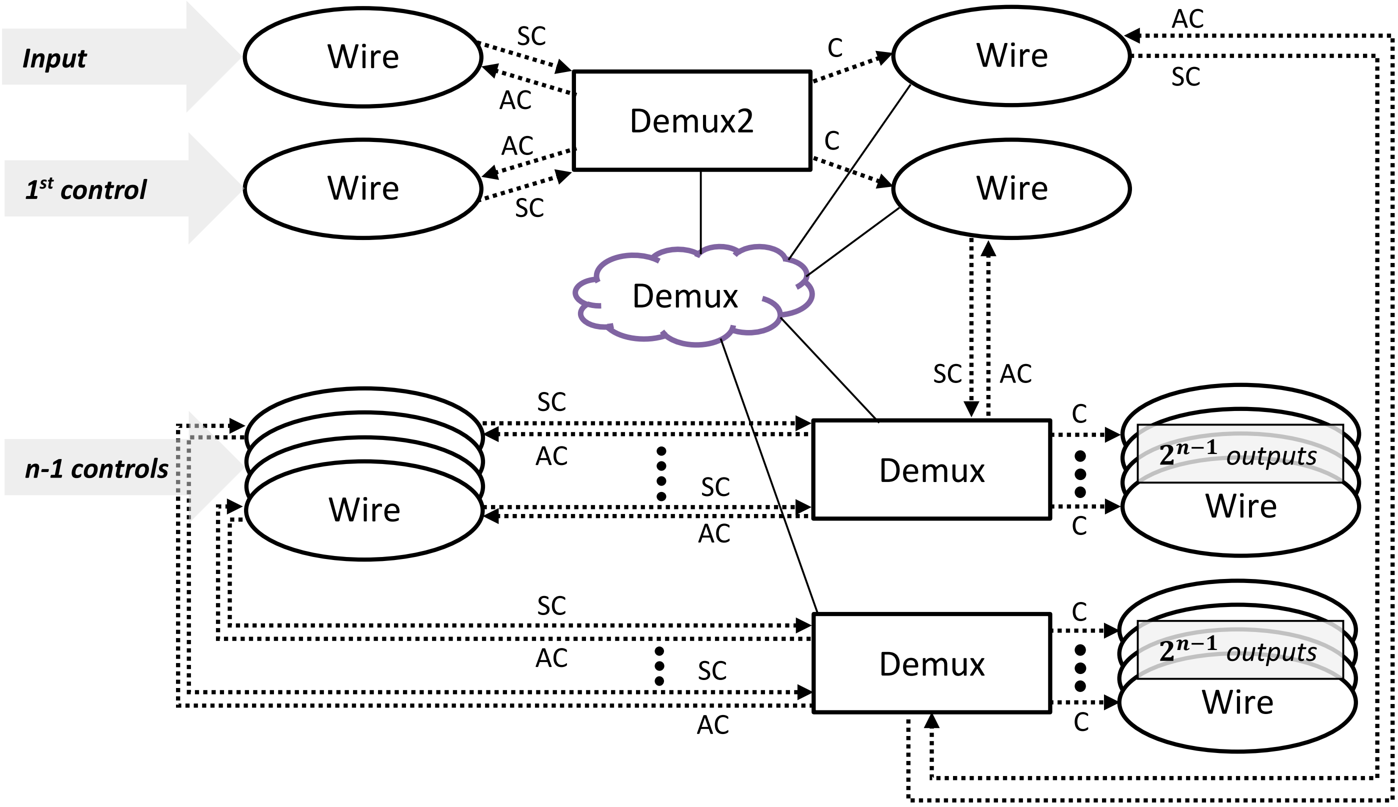
2.2.7 Full Adder Diagram



2.2.8 Demux2 Diagram



2.2.9 Demux Diagram



2.3 Message Flow Excerpt

2.3.1 Probe Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the creation of the Probe, up to the when a StateChange message is received:

	Messages sent/received	Input0 state
1.	**Probe** sends **AddComponent(name, self)** to *input0* **Wire**	False
2.	*Input0* **Wire** receives **AddComponent(name, actor)** from **Probe**	…
3.	**System** sends *true* to *input0* **Wire**	…
4.	*Input0* **Wire** receives *true* from **System** and… …sends **StateChange(true)** to **Probe**	True
5.	**Probe** receives **StateChange(true)** from *input0* **Wire**	…

2.3.2 Inverter Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the creation of the Inverter, up to when the final values are reached:

	Messages sent/received	Input0 state	Output0 state
1.	**Inverter** sends **AddComponent(name, self)** to *input0* **Wire**	False	False
2.	*Input0* **Wire** receives **AddComponent(name, actor)** from **Inverter**	…	…
3.	**System** sends *true* to *input0* **Wire**	…	…
4.	*Input0* **Wire** receives *true* from **System** and… …sends **StateChange(true)** to **Inverter**	True	…
5.	**Inverter** receives **StateChange(true)** from *input0* **Wire** and… …sends *!true* to *output0* **Wire** after a delay	…	…
6.	*Output0* **Wire** receives *false* from **Inverter**	…	False

2.3.3 And Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the creation of the And, up to when the final values are reached:

	Messages sent/received	Input0 state	Input1 state	Output0 state
1.	**And** sends **AddComponent(name, self)** to *input0* **Wire**	False	False	False
2.	**And** sends **AddComponent(name, self)** to *input1* **Wire**	…	…	…
3.	*Input0* **Wire** receives **AddComponent(name, actor)** from **And**	…	…	…
4.	*Input1* **Wire** receives **AddComponent(name, actor)** from **And**	…	…	
5.	**System** sends *true* to *input1* **Wire**	…	…	…
6.	*Input1* **Wire** receives *true* from **System** and… …sends **StateChange(true)** to **And**	…	True	…
7.	**And** receives **StateChange(true)** from *input1* **Wire** and… …sends *false && true* to *output0* **Wire** after a delay	…	…	…
8.	*Output0* **Wire** receives *false* from **And**	…	…	False

2.3.4 Or Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the creation of the Or, up to when the final values are reached:

	Messages sent/received	**Input0 state**	**Input1 state**	**Output0 state**
1.	**Or** sends **AddComponent(name, self)** to *input0* **Wire**	False	False	False
2.	**Or** sends **AddComponent(name, self)** to *input1* **Wire**	…	…	…
3.	*Input0* **Wire** receives **AddComponent(name, actor)** from **Or**	…	…	…
4.	*Input1* **Wire** receives **AddComponent(name, actor)** from **Or**	…	…	
5.	**System** sends *true* to *input1* **Wire**	…	…	…
6.	*Input1* **Wire** receives *true* from **System** and… …sends **StateChange(true)** to **Or**	…	True	…
7.	**Or** receives **StateChange(true)** from *input1* **Wire** and… …sends *false || true* to output0 **Wire** after a delay	…	…	…
8.	*Output0* **Wire** receives *true* from **Or**	…	…	True

2.3.5 OrAlt Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the first message from System, up to when the final values are reached:

	Messages sent/received	**Input0 state**	**Input1 state**	**Output0 state**
1.	**System** sends *true* to *input1* **Wire**	False	False	False
2.	*Input1* **Wire** receives *true* from **System** and… …sends **StateChange(true)** to **Inverter**	…	…	…
3.	**Inverter** receives **StateChange(true)** from input1 **Wire** and… …sends *!true* to *not1_out* **Wire** after a delay	…	…	…
4.	*Not1_out* **Wire** receives *false* from **Inverter** and… …sends **StateChange(false)** to **And**	…	True	…
5.	**And** receives **StateChange(false)** from *not1_out* **Wire** and… …sends *false && false* to *and_out* **Wire** after a delay	…	…	…
6.	*and_out* **Wire** receives *false* from **And** and… …sends **StateChange(false)** to **Inverter**	…	…	…
7.	**Inverter** receives **StateChange(false)** from *and_out* **Wire** and… …sends *!false* to *output0* **Wire** after a delay	…	…	…
8.	*Output0* **Wire** receives *true* from **Inverter**	…	…	True

2.3.6 Half Adder Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the first message sent by System, up to when the final values are reached. StateChange messages and details about messages received were excluded.

	Messages sent/received	**Input0**	**Input1**	**Output -Sum**	**Output -Carry**
1.	**System** sends *false* to *input0* **Wire**	False	False	False	False
2.	**System** sends *true* to *input1* **Wire**	…	True	…	…
3.	…**Inverter** sends *!false* to *not0_Out* **Wire**	…	…	…	…
4.	…**Inverter** sends *!true* to *not1_Out* **Wire**	…	…	…	…
5.	…**And** sends *false && false* to *and_0Not1_Out* **Wire**	…	…	…	…
6.	…**And** sends *true && true* to *and_1Not0_Out* **Wire**	…	…	…	…
7.	…**Or** sends *false || true* to *outputSum* **Wire**	…	…	True	…
8.	…**And** sends *false* && *true* to output*Carry* **Wire**	…	…	…	False

2.3.7 Full Adder Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the first message sent by System, up to when the final values are reached. StateChange messages and details about messages received were excluded.

	Messages sent/received	**Input0**	**Input1**	**Input-Carry**	**Output -Sum**	**Output -Carry**
1.	**System** sends *false* to *input0* **Wire**	False	False	False	False	False
2.	**System** sends *true* to *input1* **Wire**	…	True	…	…	…
3.	**System** sends *true* to *carry* **Wire**	…	…	True	…	…
4.	…**HalfAdder** (based on input1 and carry) sends *false* to *sum1_Out* **Wire**	…	…	…	…	…
5.	…**HalfAdder** (based on input1 and carry) sends *true* to *carry1_Out* **Wire**	…	…	…	…	…
6.	…**HalfAdder** (based on input0 and sum1_Out) sends *false* to *outputSum* **Wire**	…	…	…	False	…
7.	…**HalfAdder** (based on input0 and sum1_Out) sends *false* to *carry2_Out* **Wire**	…	…	…	…	…
8.	…**Or** (based on carry1_Out and carry2_Out) sends *true* to *outputCarry* **Wire**	…	…	…	…	True

2.3.8 Demux2 Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the first message sent by System, up to when the final values are reached. StateChange messages and details about messages received were excluded.

	Messages sent/received	**Input**	**Control**	**Output1**	**Output0**
1.	**System** sends *true* to *input* **Wire**	True	False	False	False
2.	**System** sends *false* to *control* **Wire**	…	False	…	…
3.	…**Inverter** (based on control) sends *!false* to *notCtrl_Out*	…	…	…	…
4.	…**And** (based on input and notCtrl_Out) sends *true && true* to *output0* **Wire**	…	…	…	True
5.	…**And** (based on input and control) sends *true && false* to *output1* **Wire**	…	…	False	…

2.3.9 Demux Message Flow Excerpt

The following is a message flow excerpt for a sample input combination in the form of a list of messages sent/received starting from the first message sent by System, up to when the final values are reached. StateChange messages and details about messages received were excluded.

	Messages sent/received	**Input**	**Ctrl1**	**Ctrl0**	**Out3**	**Out2**	**Out1**	**Out0**
1.	**System** sends *true* to *input* **Wire**	True	False	False	False	False	False	False
2.	**System** sends *false* to *control1* **Wire**	…	False	…	…	…	…	…
3.	**System** sends *true* to *control0* **Wire**	…	…	True	…	…	…	…
4.	…**Demux2** (based on input and ctrl1) sends *false* to *demux2_out1* **Wire**	…	…	…	…	…	…	…
5.	…**Demux2** (based on input and ctrl1) sends *true* to *demux2_out0* **Wire**	…	…	…	…	…	…	…
	…Demux based on demux2_out1 as input, and ctrl0 as output is now created… …Demux based on demux2_out0 as input, and ctrl0 as output is now created…							
6.	…**Demux2** (based on demux2_out1 and ctrl0) sends *false* to *output3* **Wire**	…	…	…	False	…	…	…
7.	…**Demux2** (based on demux2_out1 and ctrl0) sends *false* to *output2* **Wire**	…	…	…	…	False	…	…
8.	…**Demux2** (based on demux2_out0 and ctrl0) sends *true* to *output1* **Wire**	…	…	…	…	…	True	…
9.	…**Demux2** (based on demux2_out0 and ctrl0) sends *false* to *output0* **Wire**	…	…	…	…	…	…	False