Q. What is a Datapath configuration tool?

A. A UDB-based PSoC device datapath is a very small 8-bit wide processor with 8 states defined in a "control store." There are also five static configuration registers that help define the overall operation. The control store and static configuration registers within the datapath can be configured using the Datapath Configuration Tool. The Datapath Configuration Tool is used to edit datapath instance configurations in a Verilog implementation of a PSoC component. For more details on datapath tool, refer to section B of component author guide.

Q. What are the limitations of datapaths which need to be considered while creating a digital component?

A. Following are the limitations of datapath:

1. Parallel input into the datapath is limited. This restricts the ability to use the datapath where other hardware needs to provide a parallel value. Alternatives to parallel hardware loading may be possible. If enough cycles are available, then a value can be serially shifted in. If the CPU or DMA can get access to the value, then they can write the value to a FIFO or register in the datapath.

2. Parallel output is possible, but the parallel output value is always the left input to the datapath ALU. The left input to the ALU can only be A0 or A1, so that limits the output to being A0 or A1, and it restricts the ALU operation that can be performed while using parallel output to also use that value as the left input of the ALU function.

3. Only one ALU function can be performed at a time. If multiple operations are required, then a multi-cycle implementation using an multiple of the effective clock may be possible.

4. There are 8 dynamic operations available. This is typically enough operations, but for some complex multi-cycle calculations (shift, add, inc) this can be a limitation.

5. Only 2 registers are read/writable by the datapath. There are up to 6 register sources (A0, A1, D0, D1, F0, F1), but only 2 registers that can be written and then read back (A0, A1).

Q. Can I simulate a Verilog based component?

A. Warp tool present inside the PSoC Creator can only be used for the synthesis of Verilog source designs. There is no verilog simulator available with PSoC Creator. Although Cypress does not at

this time provide a Verilog simulator, some system files necessary for third party simulators to provide pre-synthesis simulations have been made available. ***Chapter 5: Simulating the Hardware*** of Component Author Guide provide details about some of the simulators which can be used to simulate the PSoC Creator based Verilog design.

Q. What are the limitations of PLD which need to be considered while creating a digital component?

A. Following are the limitations of using PLD logic:

1. The PLD does not have a direct path from or to the CPU. To get data from the CPU a control register is used. To send data to the CPU a status register is used.

2. Maximum number of register bits equal to the number of UDBs * 8 (depends on the selected device). The control and status registers can be used to augment this number of bits, but neither of those resources provides a register that can be both written and read by the PLD.

3. 12 input bits per PLD limits the efficiency and performance of wide functions. For example a wide mux function does not map well into the PLD.

Q. What are the primitive gates which can be used in Verilog file?

A. Following are the primitive gates which can be used in a verilog file:

and	-	AND logic
nand	-	NAND logic
or	-	OR logic
nor	-	NOR logic
xor	-	XOR logic
xnor	-	XNOR logic
buf	-	Buffer
not	-	NOT logic
bufif0	-	Tri-state buffer with active low enable (c1)
bufif1	-	Tri-state buffer with active high enable (c2)
notif0	-	Tri-state buffer with inverted output and active low enable (c3)
notif1	-	Tri-state buffer with inverted output and active high enable (c4)

Examples:

```
and i1 (f, a, b, c) ;   // 3-input (a, b, c) and gate
and i2 (f, a, b, c, d); // 4-input (a, b, c, d) and gate
```

```
xor i3 (f, a, b) ;        // 2-input (a, b) xor gate
buf i1 (f1, f2, a) ;      // 2 output (f1, f2) buf gate
not i2 (x, y, a) ;        // 2 output (x, y) not gate
```

Q. What are the limitations on using Datapath config tool in a State machine Design?

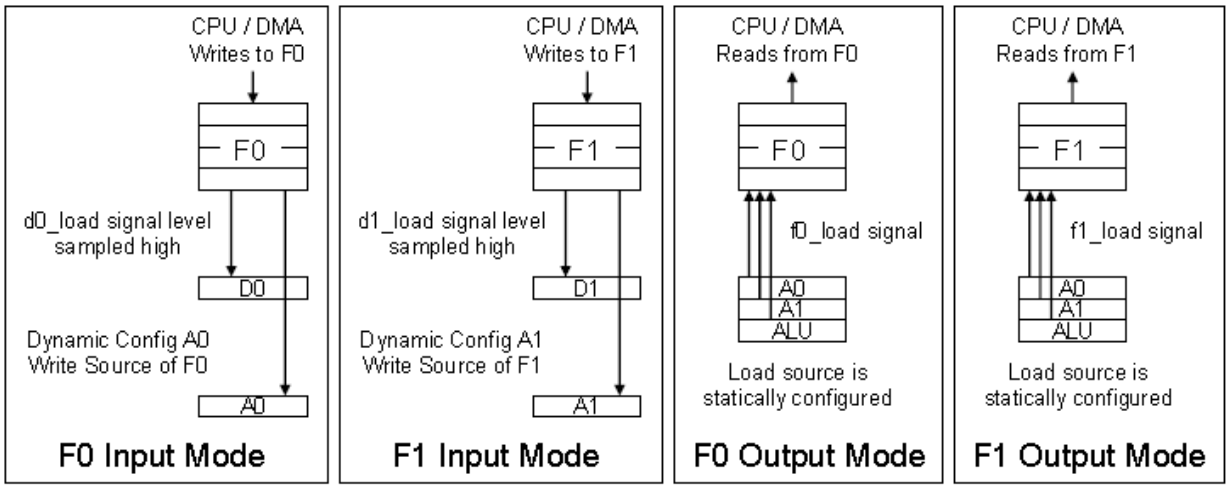
A. Datapath can be configured dynamically. The maximum possible such configurations are 8. The Configuration RAM is of 8-word x 16-bits size. And each 16-bit is required for one particular configuration. If there are more states in the design, then some states has to be merged together.

CFGRAM

Reset	Reg	Binary Value	FUNC	SRCA	SRCB	SHIFT	A0 WR SRC	A1 WR SRC	CFB EN	CI SEL	SI SEL	CMP SEL	Comment
	Reg0	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg1	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg2	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg3	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg4	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg5	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg6	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg7	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	

Q. Does the Datapath have FIFO capability for streaming applications?

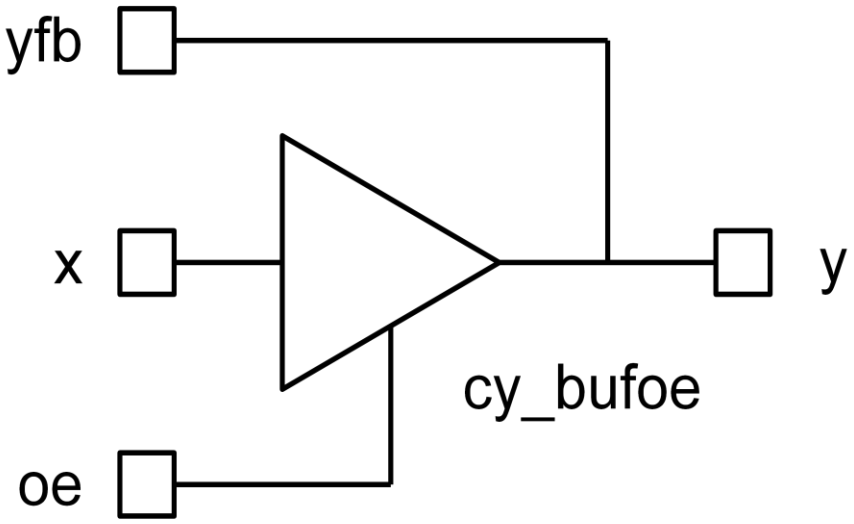
A. Yes. It has two 4-word depth FIFOs. They can be used in both Input & output modes.



More details on using these FIFOs are explained in the Component Author Guide & Training videos (refer to previous question on FIFOs).

Q. Can I synthesize tri-state logic in Verilog?

A. Warp does not synthesize tri-state logic. In order to include tri-state logic in a Verilog module the cy_bufoe must be instantiated. The tri-state output of this module, y, must then be connected to an inout port on the Verilog module. That port can then be connected directly to a bidirectional pin on the device. The feedback signal of the cy_bufoe, yfb, can be used to implement a fully bidirectional interface or can be left floating to implement just a tri-state output.

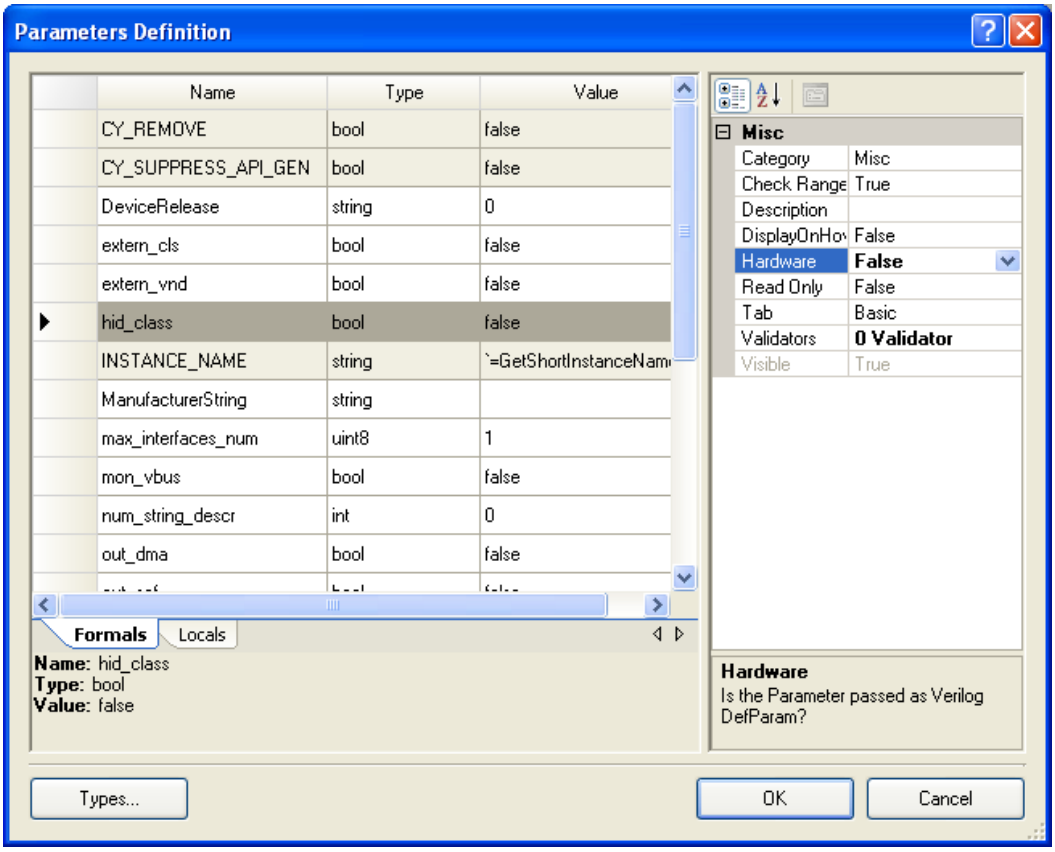



```
module ex_tri_state (
      out1,
      en,
      in1
);
      output  out1;
      input   en;
      input   in1;
cy_bufoe buf_instance (
      .x(in1),          // (input) Value to send out
      .oe(en),          // (input) Output Enable
      .y(out1),         // (inout) Connect to the bidirectional pin
      .yfb()            // (ouptut) Value on the pin brought back in
);
Endmodule
```

Q. How to access the configuration parameters from Verilog code?

A. Following is the procedure to configure the parameters from Verilog code:

1. Make the symbol file active by clicking the Symbol Editor canvas or the symbol file tab.
2. Right-click and select **Symbol Parameters...** to open the Parameters Definition dialog.



3. Under the Misc option (shown on the Right Hand Side), set the parameter to “True”. This will pass the Parameter as a Verilog DefParam.