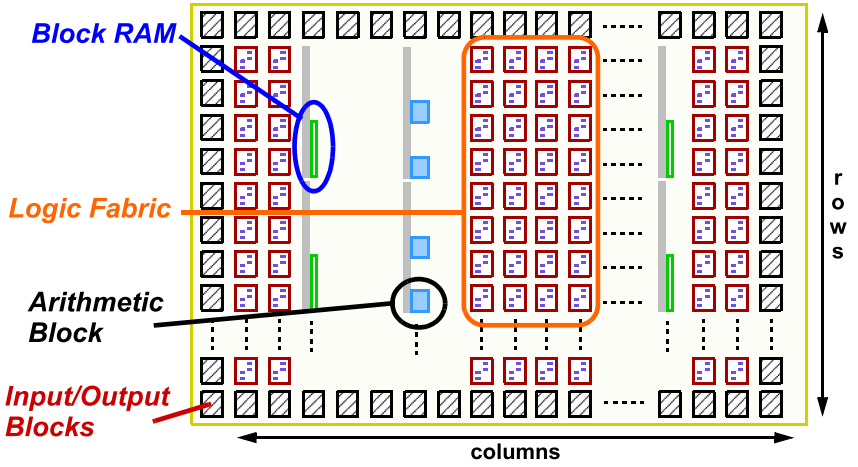
Generic FPGA Architecture

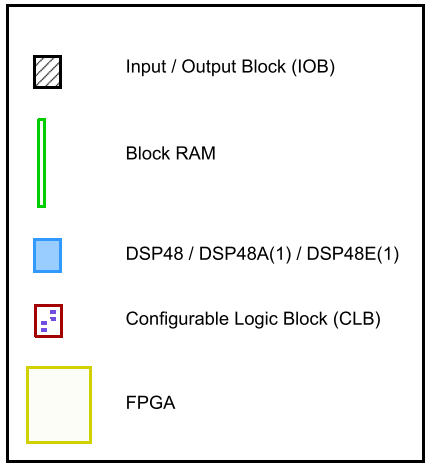
2.2

- Looking more specifically at recent Xilinx FPGAs, we also find block RAMs and dedicated arithmetic blocks... both very useful for DSP!



Notes:

Diagram Key



One of the major features of recent Xilinx FPGAs is the provision of dedicated arithmetic blocks, which enable arithmetic circuits to be implemented with lower power and higher clock frequency operation than equivalents constructed in the logic fabric (i.e. the array of CLBs). These arithmetic blocks can be configured to perform a number of different computations, and are especially suited to the Multiply Accumulate (MAC) operations upon which digital filters are built. The arithmetic blocks are named "DSP48", or a variation of this depending on the device family and series.

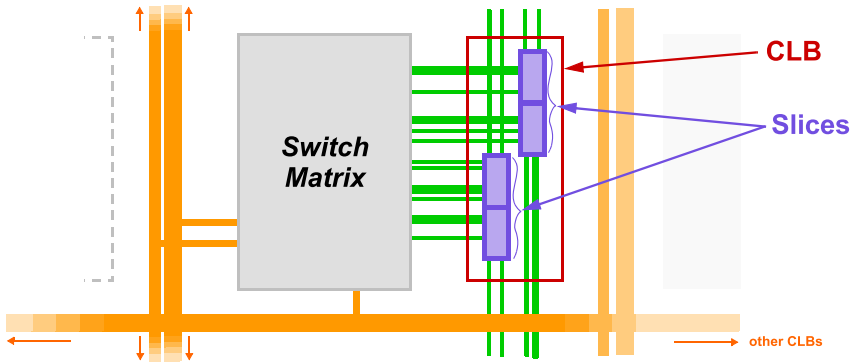
Block RAMs are also used extensively in DSP. Example uses are for storing filter coefficients, encoding and decoding, and other tasks.

Despite the inclusion of these additional resources, the logic fabric still forms the majority of the FPGA. We will now look at the CLBs which comprise the logic fabric, and how they are connected together, in further detail.

Xilinx Logic Blocks and Routing

2.3

- Xilinx FPGA logic fabric comprises ***Configurable Logic Blocks (CLBs)***, which are groups of ***Slices*** (e.g. 2 or 4 Slices per CLB).
- Signals travel between CLBs via routing resources.
- Each CLB has an adjacent ***switch matrix*** for (most) routing.



NOTE: Only a subset of routing resources is depicted above.

Notes:

The example in the main slide features a typical Xilinx FPGA architecture (architecture from other vendors are different). Logic units differ in size, composition and name! However, in all cases, their Logic Blocks include both combinational logic and registers, and routing resources are required for connecting blocks together.

Continuing with the Xilinx example, the combinational blocks are termed Lookup Tables (LUTs). In older devices these have 4 inputs, while more recent devices have 6-input LUTs. These LUTs can be utilised in four modes:

- To implement a combinatorial logic function
- As Read Only Memory (ROM)
- As Random Access Memory (RAM)
- As shift registers

The register can be used as:

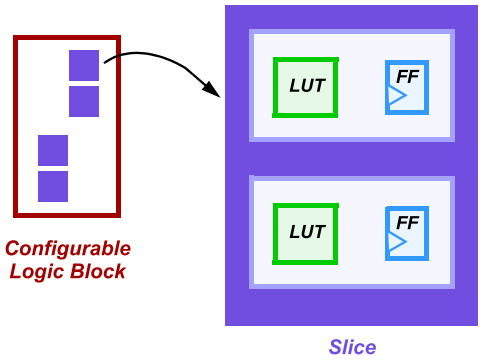
- A flip-flop
- A latch

Over the next few slides, the functionality of Slices, LUTs and registers will be described.

Logic Fabric: CLBs and Slices

2.4

- Xilinx FPGAs have several different slice configurations, which depend on the ***family*** (e.g. Spartan, Virtex) and ***generation*** (e.g. 5, 6, 7...).
- A basic slice consists of ***2 Lookup Tables (LUTs)*** and ***2 flip flops***.



- More recent devices, e.g. the ***Virtex-6*** and ***-7*** series, have more of both!

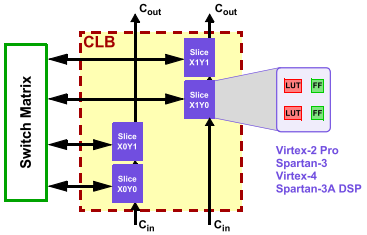
Slice Variations

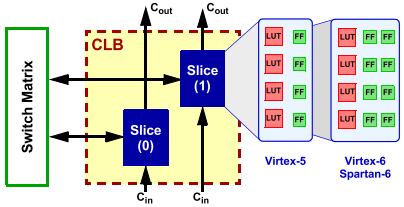
2.5

- Not all slices are the same! In most FPGAs, there are several ***different types of slices*** arranged in columns across the device.
- The exact ***slice types*** and their functionality depends on the device family (e.g. Virtex-5, Spartan-6....)
  - LUTs can act as ***RAM*** and ***Shift Registers (SRLs)*** only in some of the slice types.
  - Some slices may not contain the ***carry logic*** necessary to support arithmetic operations.
- For example, the ***Spartan-6*** has 3 different slice types:
  - ***SLICEM*** (25%) - fully featured slice.
  - ***SLICEL*** (25%) - LUTs cannot behave as RAMs or SRLs.
  - ***SLICEX*** (50%) - as SLICEL, plus there is no carry chain.

Notes:

The grouping of LUTs and flip-flops (FFs) into slices, and of slices into CLBs, also differs depending on the device family. For example, early Virtex devices had CLBs containing 4 slices, and each slice contained 2 LUTs and 2 FFs, whereas in Virtex-6 devices, each CLB contains 2 slices, and each slice contains 4 LUTs and 8 FFs.



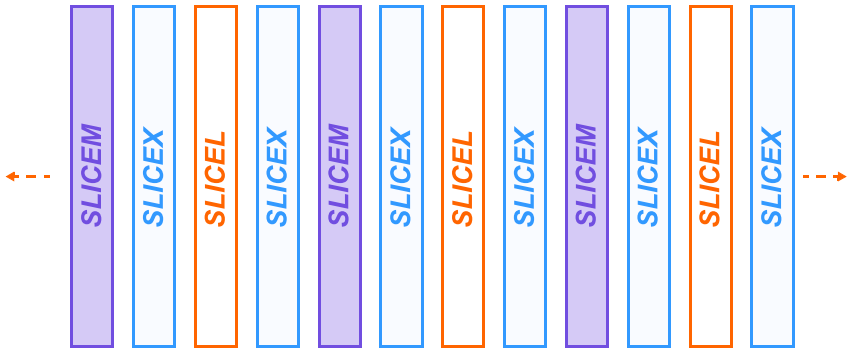


The table below summarises the logic fabric configurations of selected Xilinx series, in chronological order.

	Virtex-2 Pro	Spartan-3	Virtex-4	Virtex-5	Spartan 3A DSP	Spartan-6	Virtex-6
No of LUT inputs	4	4	4	6	4	6	6
No. of slices per CLB	4	4	4	2	4	2	2
No. of LUTs per slice	2	2	2	4	2	4	4
No. of FFs per slice	2	2	2	4	2	8	8
Total LUTs per CLB	**8**	**8**	**8**	**8**	**8**	**8**	**8**
Total FFs per CLB	**8**	**8**	**8**	**8**	**8**	**16**	**16**

Notes:

Continuing with the Spartan-6 example, the different types of Slices are arranged in columns on the FPGA as shown below. Notice that there are more SLICEX columns than SLICEM or SLICEL!

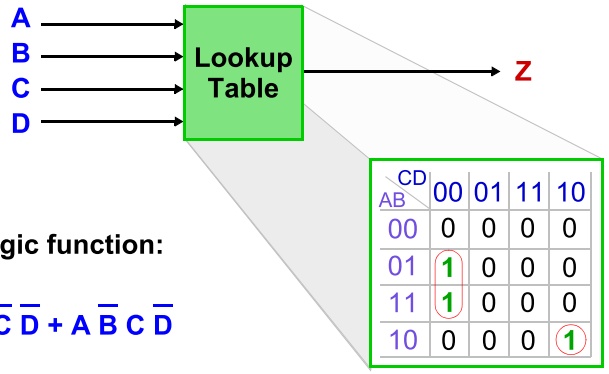


Notably, the SLICEX in the Spartan-6 does not feature carry logic, hence does not support high speed arithmetic. The Virtex-6 and 7-series devices do not contain SLICEX, and hence all slices support carry logic.

The Lookup Table

2.6

- Lookup tables may have ***4-bit*** or ***6-bit*** inputs.
- For example, when used to implement a logic function, the 4-bit input addresses the LUT to find the correct output, ***Z***, for that combination of ***A***, ***B***, ***C*** and ***D***.



Example logic function:

$$Z = B\,\overline{C}\,\overline{D} + A\,\overline{B}\,C\,\overline{D}$$

LUTs as Distributed RAM

2.7

- LUTs can also be configured as distributed RAM, in ***single port***, ***dual port*** or in some devices, ***quad port*** modes.
  - ***Single port:*** 1 address for both ***synchronous write*** operations and ***asynchronous read*** operations.
  - ***Dual port:*** 1 address for both ***synchronous write*** operations and ***asynchronous read*** operations, and 1 address for ***asynchronous reads*** only.
  - ***Quad port:*** 1 address for both ***synchronous write*** operations and ***asynchronous read*** operations, and 3 further addresses for ***asynchronous reads*** only.
- ***Dual port RAMs*** and ***quad port RAMs*** require more resources than ***single port RAMs.***
- ***Larger RAMs*** can be constructed by connecting two or more LUTs together.

Notes:

The 6-bit LUT can implement either one logic function of 6 inputs, or two logic functions of 5 inputs provided that the inputs are common to both functions.

A lookup table can also implement a ROM. Again starting with the 4-bit LUT example, this would contain ***sixteen 1-bit*** values. Instead of the four inputs representing inputs of a logic function, they can be thought of as a 4-bit address. A 1-bit value is stored within each memory location, and the appropriate output is supplied for any input address.

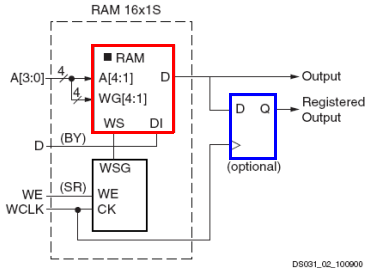
In this example, ***A*** is considered the Most Significant Bit (MSB) and ***D*** the Least Significant Bit (LSB), and the output is ***Z***.

Similarly, 6-input LUTs can be used to realise ROMs with $2^6 = 64$ entries.

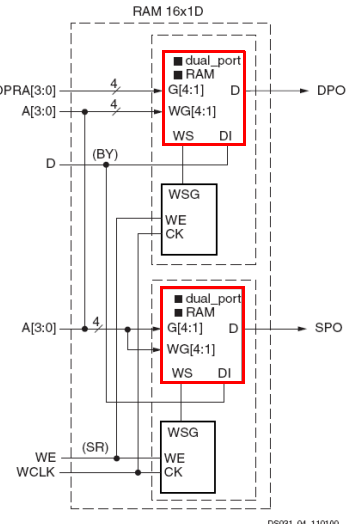
A	B	C	D	Z
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Notes:

The two diagrams below demonstrate the implementation of 16x1 single and dual port RAMs in the Virtex II Pro device, respectively. Notice that the dual port RAM requires twice as many resources as the single port RAM.



Single Port

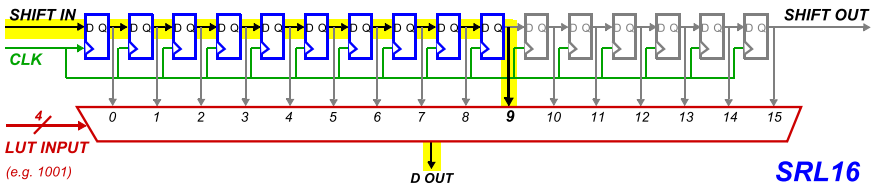


Dual Port

Source: *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet, DS083 (v4.7), November 5, 2007.*

LUTs as Shift Registers (SRL16s & SRL32s) 2.8

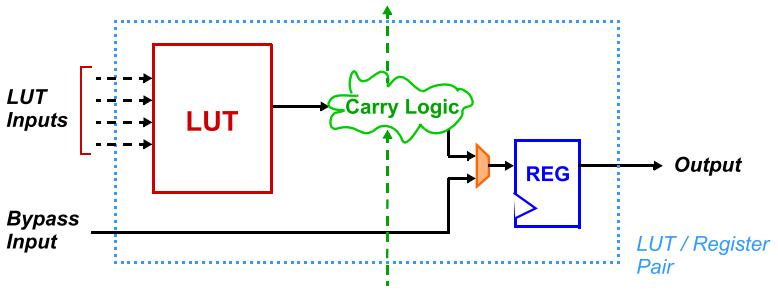
- A final alternative is to use the LUT as a ***Shift Register***.
- Additional ***Shift In*** and ***Shift Out*** ports are used, and the input defines the memory location which is asynchronously read.
- For example, if the input to a 4-input LUT (configured as an SRL16) is ***1010***, the output from the 10th register is read, as depicted below.



- The slice register at the output from the LUT can be used to add another 1 clock cycle delay. Using the register also synchronises the read operation.

Registers 2.9

- The sequential logic element which follows the lookup table can be configured as either:
  - An edge-triggered ***D-type flip flop***; or
  - A level-sensitive ***latch***
- The input to the register may be the output from the LUT, or alternatively a direct input to the slice (i.e. the LUT is bypassed).

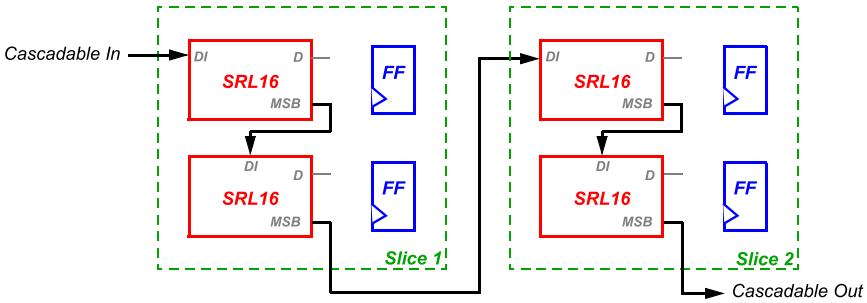


Notes:

Note that older devices with 4-input LUTs support 16-bit shift registers (SRL16s) only, while newer devices with 6-input LUTs support both 32-bit and 16-bit shift registers (SRL32s and SRL16s respectively).

As with the other LUT configurations, larger Shift Registers can be constructed by combining several LUTs together.

For example, a 64-bit shift register segment can be constructed by combining four 16-bit Shift Registers together, as shown below. The cascadable ports allow further interconnections for larger Shift Registers. SRL32s could be chained together in the same way.



Notes:

A D-type flip flop provides a delay of one clock cycle, as confirmed by the truth table below (***D(t)*** is the register input at time ***t***, and ***Q(t=1)*** is the output 1 clock cycle later). A clock signal and control inputs (set, reset, etc.) are also provided.

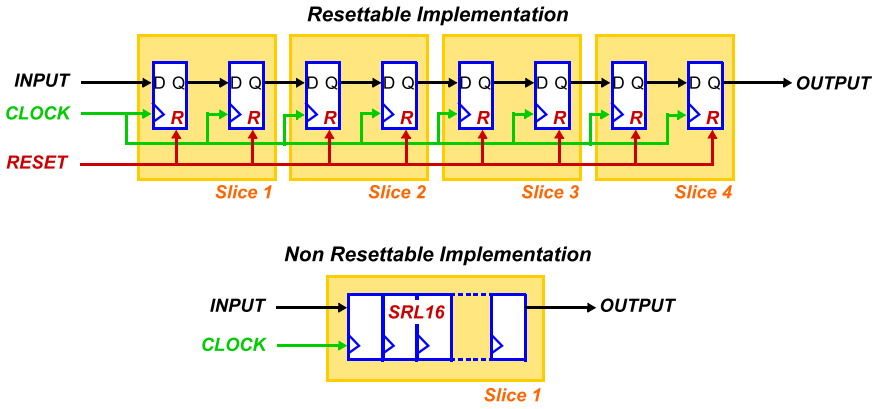
D(t)	Q(t+1)
0	0
1	1

When configured as a latch, the control inputs define when data on the *D* input is "captured" and stored within the register. The *Q* output thereafter remains unchanged until new data is captured.

If you would like to review fundamentals of Flip Flops and Registers, please consult the ***Digital Logic Review*** notes chapter.

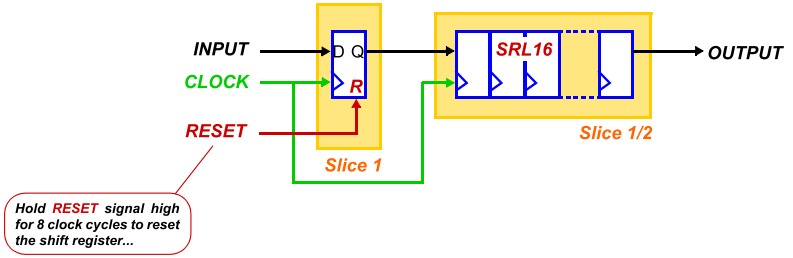
Resets: Registers and SRL16s/32s

- While ***registers*** can be ***reset***, ***SRL16s/32s cannot***. Hence including reset capabilities in a design has implications for resource utilisation.
- For example, consider an 8-bit shift register. If resettable, then each element requires a ***slice register***. If not, an ***SRL16/32*** can be used.



Notes:

We can still design a resettable 8 element shift register with an SRL16, by using a slightly more sophisticated design. Instead of making all elements resettable, we can implement the first element using a slice register, and the subsequent ones using an SRL16. The reset signal is held high for 8 clock cycles, which allows the 0 input to propagate through the shift register. Instead of using 4 slices, this design would require 2 slices at most.



Block RAMs

- Block RAMs are ***dedicated high speed memory***. Block RAMs are more suitable than distributed RAM (i.e. using LUTs as memory elements) for larger memories.
- The width of Xilinx memories is ***18-bits***. However, they can also be configured to operate in a number of different dimensions.
  - For example, the number of entries can be ***doubled*** if the storage wordlength is shortened to ***9 bits***.
  - Block RAMs can also be ***combined*** to form ***larger memories***.
- Block RAMs are usually situated in columns ***adjacent to DSP48s*** - the embedded high speed arithmetic resources.
  - DSP applications often require ***arithmetic operations*** to be performed on data which has been ***fetched from memory***, or which will be ***written to memory*** after a calculation.

Notes:

Taking the example of a 36KB Block RAM, this can take various forms ranging from 18-bit words, to 1-bit words! As the wordlength is decreased, the number of entries (addresses) increases.