13

Low-Power FPGA Implementation

13.1 Introduction

A key trend in the introduction was technology scaling and how increasing power consumption has become a worrying aspect of modern computing design. As was highlighted in Chapter 1, this has led to serious interest by major computing companies such as Intel, IBM and Microsoft in exploiting FPGA technology. Whilst in some cases FPGA implementations may offer only moderate computational improvement over CPU/GPU implementations, the equivalent designs tend to operate at much lower clock rates and power is directly proportional to this factor.

Power consumption scales down with technology evolution, and so for much of the 1980s and 1990s the new technology evolution offered an increased number of transistors operating not only at increased speed, but also at reduced power consumption. As scaling increased, though, leakage power, caused by the increasingly imperfect performance of the gate oxide thickness, increased. As the gate leakage is inversely proportional to the gate oxide thickness, this became and continues to be an increasingly important problem.

Even though some would argue to the contrary, the switch to FPGAs could be considered to be the low-power solution for high-performance computing companies, and there are a number of important reasons to reducing FPGA power consumption. As power consumption is directly related to increased temperature, improved FPGA implementations have immediate benefits for the design of the power supply to the complete system; this can result in cheaper systems and fewer components, giving a reduction in PCB area and a reduction in thermal management costs.

System reliability is related to the issue of heat dissipation, and low values result in improved chip lifetimes. Xilinx indicates that "a decrease of 10° in device operating temperature can translate to a 2X increase in component life" (Curd 2007), thus reducing the power consumption of the FPGA implementation has clear cost and reliability implications.

Section 13.2 looks at the various sources of power and introduces the concepts of static and dynamic power consumption. Section 13.3 outlines some of the approaches applied by FPGA vendors in reducing power consumption. An introduction to power

FPGA-based Implementation of Signal Processing Systems,
Second Edition. Roger Woods, John McAllister, Gaye Lightbody and Ying Yi.

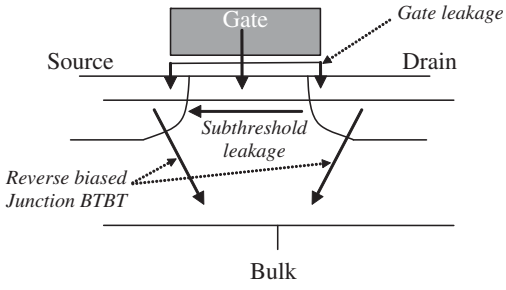


Figure 13.1 Sources of leakage components in CMOS transistor

consumption minimization techniques is given in Section 13.4, and the concepts of dynamic voltage scaling and switched capacitance are introduced. The rest of the chapter is dedicated to two core topics, namely dynamic voltage scaling in Section 13.5 and switched capacitance in Section 13.6. Section 13.7 makes some final comments with regard to power consumption in FPGAs.

13.2 Sources of Power Consumption

CMOS technology comprises *static* power consumption which is that consumed when the circuit is switched on but not processing data, and *dynamic* power consumption which is when the chip is actively processing data. The static form comprises a number of components as shown in Figure 13.1: *gate leakage* is the current that flows from gate to substrate; *source-to-drain leakage*, also known as the sub-threshold current, is the current that flows in the channel from the drain to source even though the device is deemed to be off (i.e. the gate-to-source voltage, V_{GS}, is less than the threshold voltage of the transistor V_t); and *reverse biased BTBT current* is the current that flows through the source–substrate and drain–substrate junctions of the transistors when the source and drain are at higher potential than the substrate. The static power consumption is important for battery life in standby mode, and dynamic power is particularly relevant for battery life when operating.

13.2.1 Dynamic Power Consumption

For dynamic power consumption, we consider the leakage through a simple inverter as given in Figure 13.2. Assume that a pulse of data is fed into the transistor, charging up

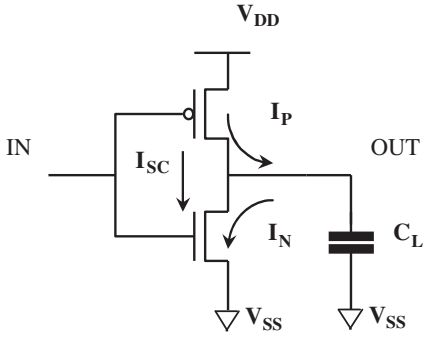


Figure 13.2 Simple CMOS inverter

and charging down the device. Power is consumed when the gates drive their output to a new value and is dependent on the resistance values of the p and n transistors in the CMOS inverter.

Thus the current through the capacitor, $i_C(t)$, is given by (Wolf 2004)

$$i_C(t) = \frac{V_{\mathrm{DD}} - V_{\mathrm{SS}}}{R_{\mathrm{p}}} e^{(-t/R_{\mathrm{p}}C_{\mathrm{L}})}, \tag{13.1}$$

where V_{DD} represents the supply voltage, V_{SS} represents the ground voltage, R_{p} is the resistance of the p-type transistor, and C_{L} is the load capacitance. The voltage, $v_C(t)$, is given by

$$v_C(t) = V_{\mathrm{DD}} - V_{\mathrm{SS}}\left[1 - e^{(-t/R_{\mathrm{p}}C_{\mathrm{L}})}\right], \tag{13.2}$$

and so the energy required to charge the capacitor, E_C, is

$$\begin{aligned} E_C &= \int i_{C_{\mathrm{L}}}(t) V_{C_{\mathrm{L}}}(t))dt, \\ &= C_{\mathrm{L}}(V_{\mathrm{DD}} - V_{\mathrm{SS}})^2 \left(e^{-t/RpC_{\mathrm{L}}} - \frac{1}{2}e^{-2t/RpC_{\mathrm{L}}}\right)\Big|_0^{\infty}, \\ &= \frac{1}{2}C_{\mathrm{L}}(V_{\mathrm{DD}} - V_{\mathrm{SS}})^2, \end{aligned} \tag{13.3}$$

where $V_{C_{\mathrm{L}}}$ and $i_{C_{\mathrm{L}}}$ is the voltage and current, respectively, needed to charge the load capacitance.

The same charge will then be dissipated through the n-type transistor when the capacitance is discharging; therefore, in a cycle of operation of the transistor, the total energy consumption of the capacitance will be $\frac{1}{2}C_{\mathrm{L}}(V_{\mathrm{DD}} - V_{\mathrm{SS}})^2$. When this is factored in with the normal operation of the design, which can be assumed to synchronous and operating at a clock frequency of f, this will define the total power consumed, namely $\frac{1}{2}C_{\mathrm{L}}(V_{\mathrm{DD}} - V_{\mathrm{SS}})^2 f$. However, this assumes that every transistor is charging and discharging at the rate of the clock frequency, which will never happen. Therefore, a quantity denoting what proportion of transistors are changing, namely α, is introduced. For different applications, the value of α will vary as shown in Table 13.1.

This gives the expression for the dynamic power consumption of a circuit,

$$P_{\mathrm{dyn}} = \frac{1}{2}C_{\mathrm{L}}(V_{\mathrm{DD}} - V_{\mathrm{SS}})^2 f\alpha \tag{13.4}$$

which when V_{SS} is assumed to be 0, reduces to the more recognized expression

$$P_{\mathrm{dyn}} = \frac{1}{2}C_{\mathrm{L}}V_{\mathrm{DD}}^2 f\alpha. \tag{13.5}$$

Table 13.1 Typical switching activity levels

Signal	Activity (α)
Clock	0.5
Random data signal	0.5
Simple logic circuits driven by random data	0.4–0.5
Finite state machines	0.08–0.18
Video signals	0.1(msb)–0.5(lsb)
Conclusion	0.05–0.5

In addition, short-circuit current can be classified as dynamic power consumption. Short-circuit currents occur when the rise/fall time at the gate input is larger than the output rise/fall time, causing imbalance and meaning that the supply voltage, V_{DD}, is short-circuited for a very short time. This will happen in particular when the transistor is driving a heavy capacitative load which, it could be argued, can be avoided in good design. To some extent, therefore, short-circuit power consumption is manageable.

13.2.2 Static Power Consumption

Technology scaling has provided the impetus for many product evolutions as it means that transistor dimensions will be adjusted as illustrated in Figure 13.3. Simply speaking, scaling by k means that the new dimensions shown in Figure 13.3(b) are given by $L' = 1/k(L)$, $W' = 1/k(W)$ and $t'_{ox} = 1/k(t_{ox})$. It is clear that this translates to a k^2 increase in the number of transistors, an increase in transistor speed and an expected decrease in transistor power as currents will also be reduced.

The expected decrease in power consumption, however, does not transpire. In order to avoid excessively high electric fields, it is necessary to scale the supply voltage, V_{DD}, which in turn requires a scaling in the threshold voltage V_t, otherwise the transistors will not turn off properly. As well as a reduction in system voltage, V_{DD}, there is a reduction in V_t which results in an increase in sub-threshold current. In order to cope with short channel effects, the oxide thickness is scaled, resulting in high tunneling through the gate insulator leading to the gate leakage. Thus, this gate leakage is inversely proportional to the gate oxide which will continue to decrease for improving technologies therefore exacerbating the problem.

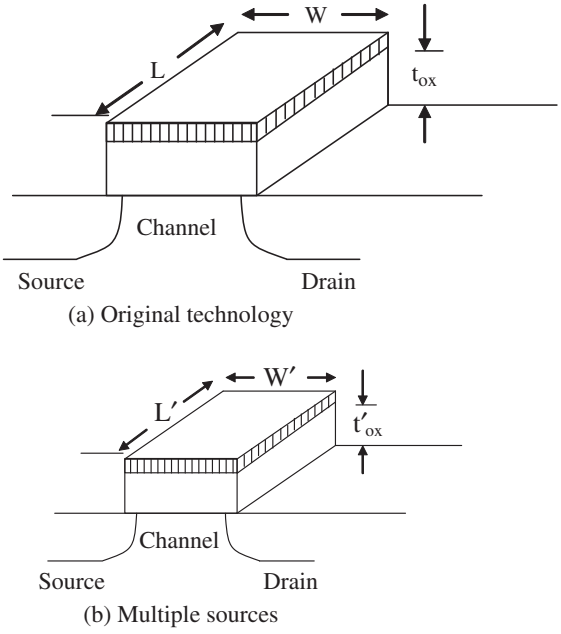


Figure 13.3 Impact of transistor scaling

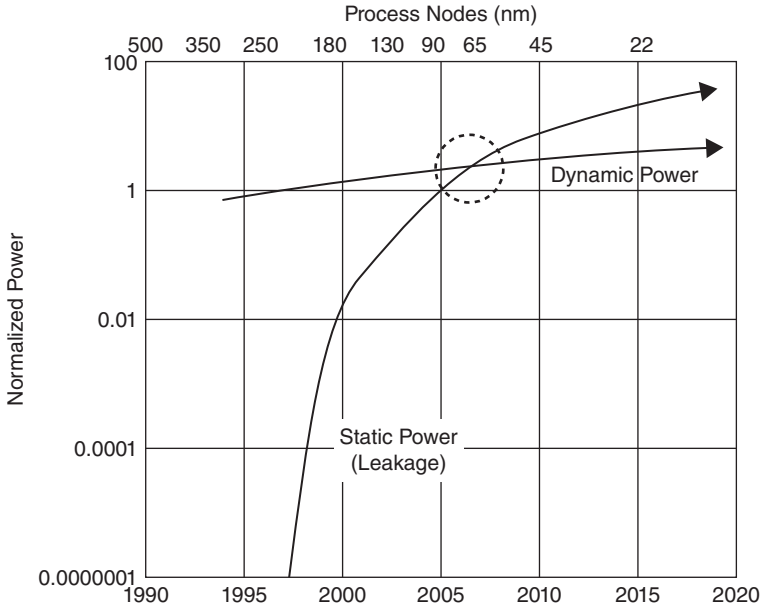


Figure 13.4 Impact of static versus dynamic power consumption with technology evolution (ITRS, 2003; Kim *et al.* 2003)

Scaled devices also require high substrate doping densities to be used near the source–substrate and drain–substrate junctions in order to reduce the depletion region. However, under high reversed bias, this results in significantly large BTBT currents through these junctions (Roy *et al.* 2003). The result is that scaling results in a dramatic increase in each of these components of leakage and with increasing junction temperatures, the impact is worsened as the leakage impact is increased (Curd 2007).

The main issue with increasing numbers of transistors is that their contribution to static power consumption is also growing. This is illustrated by the graph in Figure 13.4 which shows that a cross-over point has occurred for 90 nm and smaller technology nodes where static power began to eclipse dynamic power for many applications.

This graph has a major impact for many technologies as it now means that unlike power consumption in the previous decade where the problem was largely impacted by the operation of the device, allowing designers to reduce the impact of dynamic power consumption, it will be predicated on the normal standby mode of operation. This will have an impact on system design for fixed – but particularly for wireless – applications. A number of approaches have been actively pursued by FPGA vendors to address this.

13.3 FPGA Power Consumption

Xilinx has addressed the impact of high static power in its Virtex-5 and subsequent devices by employing a triple oxide (Curd 2007). Triple oxide is used to represent the three levels of oxide thickness used in FPGAs. A thin oxide is used for the small, fast transistors in the FPGA core, a thick oxide is used for the higher-voltage swing

transistors in the I/O which do not have to be fast, and a third or middle-level oxide called a *midox* oxide is used for the configuration memory cells and interconnect pass transistors.

The midox slightly thicker gate oxide dramatically reduces leakage current when compared to the thin oxide equivalent, as the V_t reduces both the source-to-drain leakage and gate leakage. The midox transistor is used in non-critical areas in the FPGA, such as in the configuration memory used to store the user design, which do not need to be updated during device operation. In addition, it is used for the routing transistors, once again as the speed of operation is not critical. The impact is to reduce the leakage current of millions of transistors in the FPGA, thus dramatically reducing the power consumption.

In addition to the triple-oxide innovation, some architectural trends have acted to also reduce the power. This has been based largely around the shift to the six-input LUT in Virtex-5 which the company argues gives a 50% increase in logic capacity per LUT, and the shift to the larger eight-input LUT for Altera. The key effect is that more logic is mapped locally within the LUT where smaller transistors are used. Since transistor leakage is measured in current per unit width, smaller transistors will have less leakage and fewer large transistors are needed.

The Xilinx 7 series was characterized by a shift to TSMC's high-performance, low-power 28 nm process called 28HPL. It was the first to use a high-K metal gate (HKMG) process. The company argues that the shift to 20 nm in UltraScale™ should result in 40% overall device-level power savings over Xilinx 7 series FPGAs and up to 60% savings at 16 nm. This then gives an indicative reduction in static power consumption as indicated by the estimated power figures for mobile backhaul in Artix-7 (Figure 13.5).

Altera takes advantage of Intel's 14 nm Tri-Gate process. It comprises 3D tri-gate transistors and provides good dimensional scaling from the 22 nm process. The transistor "fins" are taller, thinner, and more closely spaced; this gives improved density and lower capacitance, resulting in an SRAM cell size that is almost half that for 22 nm.

With Arria 10 devices, a programmable power technology (PPT) is employed which is the company's patented approach for tuning the switching speed of logic elements in the speed-critical paths of a user design. This tuning allows the transistor's threshold voltage in a higher-speed path to be set to a lower value, increasing its switching speed. Transistors in a lower-speed path can be tuned to a higher threshold voltage, thereby reducing the static power consumption by up to 20%.

Other features include using transistors with a lower voltage threshold and small minimal channel length for high-speed operation in the DSP blocks and logic elements; a

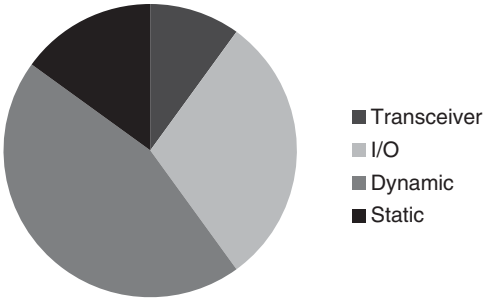


Figure 13.5 Estimated power consumption for mobile backhaul on Artix-7

low-power transistor is then used in the less demanding areas, namely configuration RAM and memory blocks.

Whilst addressing static power consumption is therefore vital in developing a low-power FPGA solution, it is somewhat predetermined by the underlying application of many of the above technologies at fabrication; with a fixed architecture, there is little scope available to the FPGA designer. There are, however, some techniques that can act to reduce the power consumption from a dynamic power consumption designer perspective.

13.3.1 Clock Tree Isolation

One of the main contributors to static power consumption is the clock signal through its distribution network, namely a clock tree and the circuits connected to it which it will act to toggle on a regular basis, particularly if the design is synchronous. As the description in Chapter 5 indicated, most FPGAs have a number of clock signals with PPLs and individual dedicated clock tree networks which can be turned off and on as required. Static power consumption reduction is achieved by turning off parts of the clock networks on the FPGA using, for example, multiplexing and then employing clever placement and routing techniques to ensure this can be achieved effectively (Huda *et al.* 2009).

Xilinx has provided an automated capability, in its Vivado® Design Suite v2013.1 and onwards, to its standard place and route flow. It performs an analysis on all portions of the design, detecting sourcing registers that do not contribute to the result for each clock cycle. It then utilizes the abundant supply of clock enables (CEs) available in the logic to create fine-grained clock-gating. This is successful as each CE typically drives only eight registers, providing a good level of granularity to match most bus sizes. Intelligent clock-gating optimization can also be used for dedicated BRAM in simple or dual-port mode by using additional logic to control the array and avoiding unnecessary memory accesses.

13.4 Power Consumption Reduction Techniques

It is clear from equation (13.5) that a number of factors impact dynamic power consumption. The voltage, V_{DD}, will have been predetermined by the FPGA vendor (and optimized to provide low-power operation), and any scope to reduce this voltage will be made available to the user via the design software. The process for achieving this has been outlined by the vendors (Chapman and Hussein 2012).

This only leaves scope for adjusting the other parameters, namely the toggling rate presumed to be the clock frequency, f times the switching activity α and the load capacitance, C_L. However, any technique that acts to adjust the clock frequency f and/or switching activity α should be developed on the understanding that the overall clock rate for the system will generally have been determined by the application and that the switching activity will be governed again by the application domain, meaning that levels shown in Table 13.1 should be given consideration.

Generally speaking, power reduction techniques (Chandrakasan and Brodersen 1996) either act to minimize the switched capacitance (Cf) or employ techniques to increase

performance by reducing the supply voltage, thereby achieving a squared reduction in power at the expense of a linear increase in area, i.e. C_L or frequency, f. In voltage minimization techniques, transformations are used to speed up the system's throughput beyond that necessary; the voltage is then reduced, slowing up performance until the required throughput rate is met but at a lower power consumption budget. This has considerable scope for SoC systems where the system can be developed knowing that the circuit will be implemented using a range of voltage values. It is a little more difficult in FPGAs, but work by Chow *et al.* (2005) and Nunez-Yanez (2015) suggests viable techniques which are described later.

There is also some scope to reduce the capacitance and switching activity, but rather than consider this separately, it is useful to think about reducing the *switched capacitance* of a circuit, i.e. the sum of all of toggling activity of each node multiplied by the capacitance of that node. This is an important measure of power consumption as opposed to just circuit capacitance alone, as a circuit can either have a large capacitative net with a low switching activity which will not contribute greatly to power consumption, or a number of low-capacitance nets with a lot of switching activity which can make a not insubstantial contribution to power consumption. The same argument applies to switching activity levels as some nets can have high switching activity but low capacitance, and so on. A large proportion of the techniques fall into this domain and so more of the discussion is centered upon this aspect.

13.5 Dynamic Voltage Scaling in FPGAs

As the name indicates, dynamic voltage scaling involves reducing the supply voltage of the circuit in such a way that it can still operate correctly. Typically, the designer will have exploited any voltage capacity by applying design techniques to slow down the circuit operation, presumably by achieving an area reduction or some other gain. Thus reducing the voltage may cause a circuit failure as the critical path timing may not be met. This is because scaling the voltage causes as impact of circuit delay, t_d, as given by the expression (Bowman *et al.* 1999)

$$t_d = \frac{kV_{DD}}{(V_{DD} - V_t)^2}, \tag{13.6}$$

where k and α are constants with $1 < \alpha < 2$. As V_{DD} is scaled, the circuit delay increases.

Voltage scaling should be only applied to the FPGA core as it is important that the I/O pins operate to the specifications to which they have been designed. Whilst the circuit delay increases, only parts of the circuit need to operate at the shortest circuit delay, which means that there is scope for reducing the voltage for a large portion of the circuit without impacting performance. Of course, the design has to be reliable across a range of devices, and there can be a variation in delay times as well as operating temperature.

The approach shown in Figure 13.6 outlines how a transformation can be applied to reduce power consumption. Parallelism and voltage scaling can be employed to reduce the power consumption of a circuit. If the performance is met with the functional units shown in Figure 13.6(a), then parallelism can be used to give the circuit shown in Figure 13.6(b); as this circuit can now operate much faster than the original, there is scope to

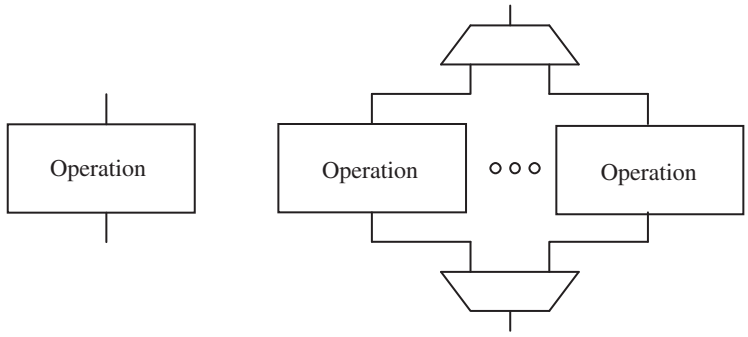


Figure 13.6 Use of parallelism (and voltage scaling) to lower power consumption

reduce power by scaling the voltage to the resulting circuit. This may seem counterproductive as the circuit has a larger area, resulting in increased capacitance and switching activity, but the power reduction due to the V_{DD}^2 scaling in addition to the scaling in frequency will more than outweigh any increase.

The impact of adaptive voltage scaling can be addressed in a number of ways, adding circuitry to detect exactly when this happens with a specific FPGA implementation and detecting the correct voltage threshold to achieve lower power operation (Chow *et al.* 2005; Ryan and Calhoun 2010) or applying design techniques to speed up circuit operation. Chow *et al.* (2005) work on the first principle of assuming the original circuit, scaling down the internal voltage supply and then checking for any possible errors on correct operation. Of course, the approach can be used for parallelization as shown in Figure 13.6.

The authors argue that on the basis that the designer can observe any two types of design errors as a result of the voltage scaling in normal operation, it is just a case of trying to work out two other types of error, namely *I/O errors* (due to the lower-voltage core circuit having to interface with the I/O which is operating at the original voltage) and *delay errors* (occurring as a result of the critical path now possibly not meeting the timing). In the case of I/O errors, the danger is that a high output signal from the core will be too small for the threshold voltage of the I/O buffer to correctly detect its value.

The lowest supply voltage is estimated at runtime and adjusted accordingly. A logic delay measurement circuit (LDCM) (Gonzalez *et al.* 1997) is used with an external monitor to adjust the FPGA internal voltage at 200 ms intervals. Typical power savings of 20–30% were achieved with a Xilinx Virtex 300E-8 device in a number of experiments. Nunez-Yanez *et al.* (2007) present a similar approach that applies dynamic voltage scaling (DVS) by adjusting first the voltage, then searching for a suitable operating frequency using the LDCM. Energy savings of up to 60% on an XC4VSX35-FF668-10C FPGA, were achieved by scaling down from 1.2 V to 0.9 V.

Nunez-Yanez (2015) extends this work by proposing adaptive voltage scaling (AVS) which uses voltage scaling together with dynamic reconfiguration and clock management. By exploiting the available application-dependent timing margins, a power reduction up to 85% from operating at 0.58 V (compared to a nominal 1 V) is achieved. He

argues that the energy requirements at 0.58 V are approximately five times lower compared to the nominal voltage. This is achieved through the development of an AVS unit which monitors the delay properties and adjusts the voltage to operate at the lowest-energy point for a given frequency.

These techniques are very design-specific and involve implementation of dedicated circuitry to achieve the power reduction. They are probably relevant for designs where strict power consumption needs to be achieved and the design is less likely to change. Moreover, they may have to be checked for every FPGA component used as chip performance tends to vary.

13.6 Reduction in Switched Capacitance

The previous techniques require that the voltage is scaled (typically only the internal voltage) but do not deal with the results of the application of this scaling. However, as suggested earlier, it is also possible to reduce the switched capacitance of the circuit. A number of techniques are considered which are well understood in the literature even though, in some cases, some have not been directly applied to FPGAs.

13.6.1 Data Reordering

In DSP processor implementations described in Chapter 4, the architecture is typically composed of data and program memory connected to the processor via data buses; this is also the case in the increasing number of FPGA-based processors such as the one described in detail in Chapter 12. In these architectures, therefore, the capacitance of the buses will be fixed, but in some cases it may be possible to reorder the data computation in order to minimize the Hamming difference and thereby achieve a reduction in the switching activity on large capacitative buses. Consider the trivial example of a four-tap filter given by $y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) + a_3x(n-3)$ where the coefficient are as follows:

$a0$	1011		$a0$	1011	
		|3			|1
$a1$	0110		$a2$	1001	
		|4			|3
$a2$	1001		$a3$	0100	
		|3			|1
$a3$	0100		$a1$	0110	
		|4			|3
$a0$	1011		$a0$	1011	
		14 transitions			8 transitions

It can be seen that if the coefficients are loaded in the normal numerical order, namely a_0, a_1, a_2, a_3 and back to a_0, then this will require 14 transitions which will involve charging and discharging of the line capacitance of the interconnect, which could be considerable. By changing the order of loading, the number of transitions can be reduced as shown above. The main issue is then to resolve the *out-of-order* operation of the filter.

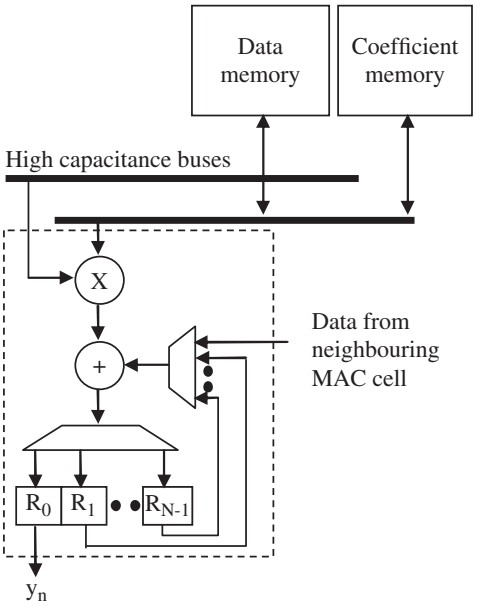


Figure 13.7 Generic MAC time domain filter implementation

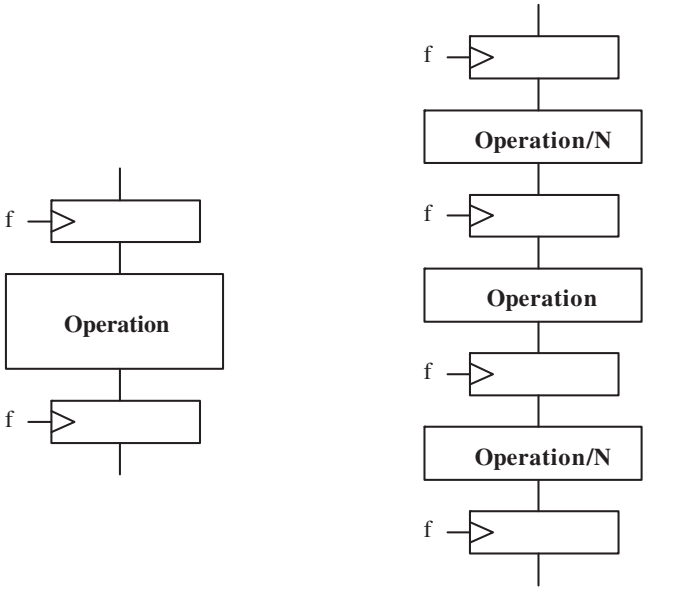
In Erdogan and Arslan (2002), the authors show how this can be applied to the design of FIR filters to achieve a reported 62% reduction in power consumption. The architecture reproduced from their earlier paper (Erdogan and Arslan 2000) presents a MAC structure comprising a multiplier and adder which are fed by program and coefficient data from the memory. This type of structure is shown in Figure 13.7. The structure can be made cascadable by feeding the output of the previous section into the current block.

This structure thus allows an out-of-order operation to be performed on the data accumulating in the structure, resulting in a reduction of the switching data from the coefficient memory via the large-coefficient data bus. By exploiting the direct-form FIR filter implementation rather than possibly the transposed form, this also reduces the switching on the data bus as one data word is loaded and then reused.

13.6.2 Pipelining

An effective method to reduce power consumption is by using pipelining coupled with power-aware component placement. Pipelining, as illustrated in Figure 13.8, breaks the processing of the original circuit (Figure 13.8(a)) into short stages as illustrated in Figure 13.8(b), thereby providing a speedup but with an increase in the latency in terms of the number of clock cycles, although this does not necessarily mean a large increase in time (as the clock period has been shortened). The increase in processing speed can be used in a similar way to the use of parallelism in Figure 13.6, to allow the voltage to be reduced, thereby achieving a power reduction (Chandrakasan and Brodersen 1996).

In addition to providing the speedup, though, pipelining provides a highly useful mechanism to reduce power consumption (Keane *et al.* 1999; Raghunathan *et al.* 1999). Work by Wilton *et al.* (2004) has outlined in detail the application of pipelining to FPGA



(a) Original implementation (b) Pipelined implementation

Figure 13.8 Application of pipelining

and the impact that can be achieved. Whilst this works well for SoC, it has greater potential in FPGAs, as the registers were previously available and would have been contributing to power even if unused, as they are in the FPGA in any case.

Glitches can contribute to dynamic power consumption and occur when logic signals do not arrive at time at the LUTs as they traverse different levels of logic and perhaps different routing paths. They contribute significantly to dynamic power consumption (from 20% to 70%) through the unplanned switching activity of signals (Shen *et al.* 1992). It is argued that pipelining acts to reduce the logic depth of combinatorial paths and thus decrease the probability of glitches and prevent them being propagated from one pipeline stage to the next (Boemo *et al.* 2013). Moreover, there is a reduction in the net lengths and an overall reduction in the switching activity of longer net activity; by reducing a high contributor to power and shortening high-frequency nets, the dynamic power dissipated can be significantly reduced.

This technique is particularly effective in FPGA technology because the increasing flexibility comes at a power budget cost due to the long routing tracks and programmable switches. These features provide the programmability but are laden with parasitic capacitance (Chen *et al.* 1997) as illustrated by Figure 13.9, which shows a model of a typical FPGA route. Another benefit of implementing pipelining in a FPGA is that it may be using an underutilized resource, namely the flip-flop at the output of the logic cell, thereby only providing a small area increase (Wilton *et al.* 2004).

There are also other advantages to applying pipelining in FPGA designs. The aim of the place and route tools is to achieve the best placement in order to achieve the required speed. By applying pipelining, this provides a more rigorous structure to the design and allows faster placement of the logic (McKeown and Woods 2008). It also acts to reduce

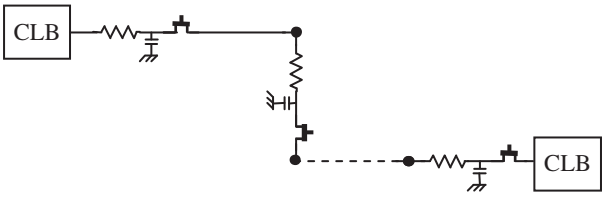


Figure 13.9 Typical FPGA interconnection route

the number of longer nets that result in the design, allowing speed targets to be more easily met.

Pipelining Examples

Work by Boemo *et al.* (2013) reports a set of experimental measurements on the impact of applying pipelining on a range of FPGA technologies for a range of integer multipliers. Some of the results are presented in Table 13.2. The power consumption was obtained by measuring the actual current from an internal core power supply on the FPGA board. The *power reduction factor* is expressed by the ratio between the power consumption of the best pipeline version and the power consumption of the original combinatorial circuit.

In the paper, they review a range of publications reporting the use of pipelining across a number of applications. They indicate that power reductions around 50% tend to be obtained. In addition, higher values of the power reduction factor tend to be achieved with larger wordlengths.

Consider the application of pipelining to the FIR filter shown in Figure 13.10. A number of filter realizations and designs were investigated, namely a 4-, 8-, 16-, and 32-tap FIR filter implemented on a Virtex-II XC-2V3000bf 957-6 FPGA. The filter was initialized by loading coefficients using an address bus, data bus and enable signal so that this was consistent for all implementations. No truncation was employed and the output word length is 24 and 27 bits for the 4-tap and 32-tap filters, respectively. Expansion is handled by including a word growth variable which is defined for each filter size to prevent truncation.

Xilinx ISE™ Project Navigator (version 6.2) was used to translate, map, place and route the designs, and sub-programs of the ISE™ design suite were used to compile component libraries, manually place and route and generate post place and route VHDL files for XPower. Xpower was then used to generate simulation results for Table 13.3.

Table 13.2 Pipelined, 54-bit multiplier running at 50 MHz in Spartan-6 FPGA

Pipeline stages	mA	Flip-flops	Power reduction factor
1	67.2	216	1.00
3	34.9	666	0.52
5	27.9	2346	0.42
7	—	—	—
9	23.1	3206	0.34

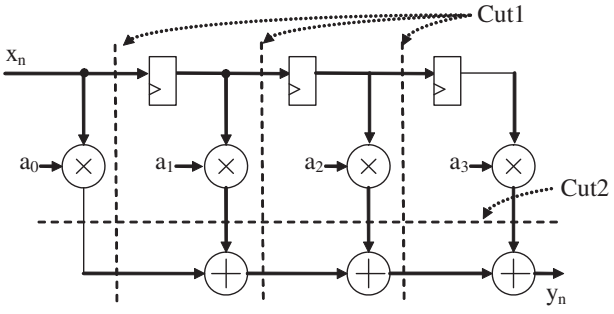


Figure 13.10 Pipelined FIR filter implementation for low power

The unpipelined version (PL0) was used to benchmark the results, with the only change being the latency imposed by pipelining stages. A speech data file was used to simulate the designs, the filter clock speed was 20 MHz and the simulation time was 200 μs. Two levels of pipelining were investigated; a layer of pipelining after the multipliers as shown by Cut2 and given as PL1 in Table 13.3; another pipeline cut in the adder (and delay) chain, Cut1, in addition to the first level of pipelining was then applied and defined as PL2 in Table 13.3, i.e. PL2 encompasses Cut1 *and* Cut2.

The results show power reductions of 59–63% for single-stage pipelined versions and 82–98% for two-stage pipelined versions, depending on filter size. Whilst this is quite dramatic and only based on simulation, it gives some idea of the power reduction possible. It is clear that pipelining becomes more effective as filter size and thus design area and interconnection lengths increase, giving greater opportunity for power-aware placement. Net lengths are shortened by placing interconnected components closely together.

This acts to reduce power consumption in two ways: firstly by decreasing the net capacitance, and secondly by reducing toggling. Partitioning the design into pipelined stages further reduces power consumption by diminishing the ripple effect of propagation delays. This can be seen in Figure 13.11, which shows post place and route capacitance in pF rounded to the nearest integer plotted against the summation of the toggling activity on nets with equivalent capacitance. These values are plotted for PL0, PL1 and PL2, showing that not only is toggle activity reduced on high-capacity nets but overall there are fewer toggles in the design when power reduction techniques are implemented.

Table 13.3 Internal signal/logic power consumption of various filters

	FIR filter tap size			
Technique	**4**	**8**	**16**	**32**
PL0	8.4	89.7	272.0	964.2
PL1	3.1 (−63%)	29.1 (−68%)	89.7 (−67%)	391.7 (−59%)
PL2	1.5 (−82%)	6.7 (−93%)	8.1 (−97%)	16.4 (−98%)

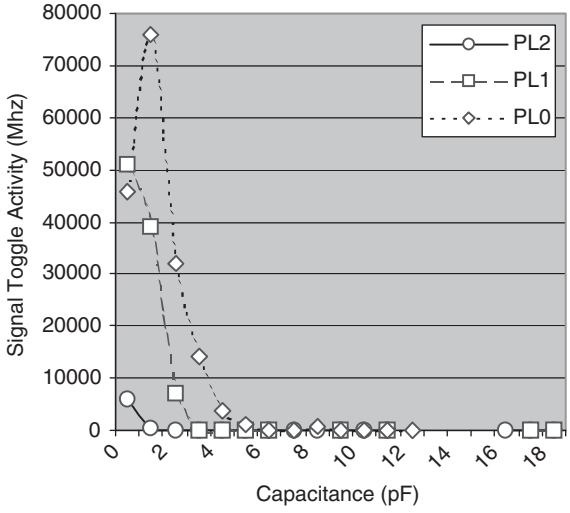


Figure 13.11 Tap signal capacitance and toggle activity

The results presented in Wilton *et al.* (2004) are equally impressive but more thorough; they are based on data from a real board setup. Results are presented for a 64-bit unsigned integer array multiplier, a triple-DES encryption circuit, an eight-tap floating-point FIR filter and a CORDIC circuit to compute the sine and cosine of an angle. Power results just for the FIR filter and the CORDIC circuit are given in the Table 13.4. They were taken from the circuits implemented on an Altera Nios Development Kit (Stratix Professional Edition) which contains a 0.13 μm CMOS Stratix EP1S40F780C5 device which gave the original FPGA power results, and the estimated power was taken from a Quartus simulator and power estimator.

The results show the impact of applying of different levels of pipelining. The authors quote savings overall of 40–82% and indicate, when they factor out quiescent power from the results, that the savings on the dynamic logic block energy can be as high as 98%. They indicate that lower-level physical design optimizations presented in the work in Lamoureux and Wilton (2003) can achieve energy savings of up 23%, highlighting the importance of applying system-level optimizations and highlighting the impact of pipelining generally.

Table 13.4 Pipelining results for 0.13 μm FPGA

Benchmark circuit	Pipeline stages	Estimated power	Original FPGA power
8-tap floating point FIR filter	2	4,420	7,866
	4	2,468	5,580
	Max.	776	3,834
CORDIC circuit to compute sine and cosine of angle	4	971	5,139
	8	611	4,437
	16	565	4,716
	Max.	567	4,140

13.6.3 Locality

It is clear from the discussion on switched capacitance and indeed from Figure 13.11 that the length of interconnection can have a major impact on power consumption. Thus, locality can be a highly attractive feature in circuit architectures for signal and data processing architectures. One class that features localized interconnection is systolic arrays (Kung and Leiserson 1979; Kung 1988); they were initially introduced to address issues of design complexity and the increasing problem of long interconnect in VLSI designs (Mead and Conway 1979).

In addition to locality, systolic array architectures also employ pipelining which makes them attractive for implementing regular computations such as matrix–matrix multiplication and LU decomposition. They benefit immensely from the highly regular nature of DSP computations and offer huge performance potential. The concept was extended to the *bit level*, resulting in bit-level systolic arrays (Woods *et al.* 2008).

The key challenge is to be able to map algorithms into these structures (Choi and Prasanna 2003; Woods *et al.* 2008). Kung (1988) classified DSP algorithms as "locally recursive," for example as with matrix multiplication, and "globally recursive." In locally recursive algorithms, data dependency is limited to adjacent elements, whereas in globally recursive algorithms inherently complex communication networks are required as some cells need to communicate with numerous others.

In McKeown and Woods (2011), some attempt is made to define the locality in terms of a parameter called the *index space separation*. Index space is defined as a lattice of points in an n-dimensional discrete space (Parashar and Browne 2000). Strictly, then, index space separation is defined as a measure of total distance values between indices. Defining the hierarchical space where each position in the lattice vector space is a Cartesian coordinate allows the index space to be defined in Euclidean geometry: the index space separation is the Euclidean distance between indices. The index space separation, η, between the two indices $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ is thus defined as

$$\eta = \sqrt{(a_1 + b_1)^2 + (a_2 + b_2)^2 + \ldots + (a_n + b_n)^2} = \sqrt{\sum (a_n + b_n)^2}. \tag{13.7}$$

McKeown and Woods (2011) argue that data dependency can be measured using the index space separation as the relative distances over which data must be passed between consecutive operations. This is not particularly important for power consumption in processor-style implementations, but in FPGA designs it relates directly to the separate individual interconnections created as a result of the FPGA place and route process. This directly relates to the FPGA dynamic power consumption.

Application to FFT Implementation

The Cooley–Tukey algorithm (Cooley and Tukey 1965) is commonly used to compute the FFT (see Section 2.3.2). The problem is that its globally recursive nature manifests itself in the irregular routing where data are routinely passed to non-adjacent global PEs as shown in Figure 2.5.

By removing the in-place restriction of the Cooley–Tukey algorithm, matrix decomposition can be used to increase data locality, thereby minimizing the index space separation. By identifying periodicity and symmetry in the structured transform matrix along which the algorithm can be factorized, a decomposition is achieved with a smaller

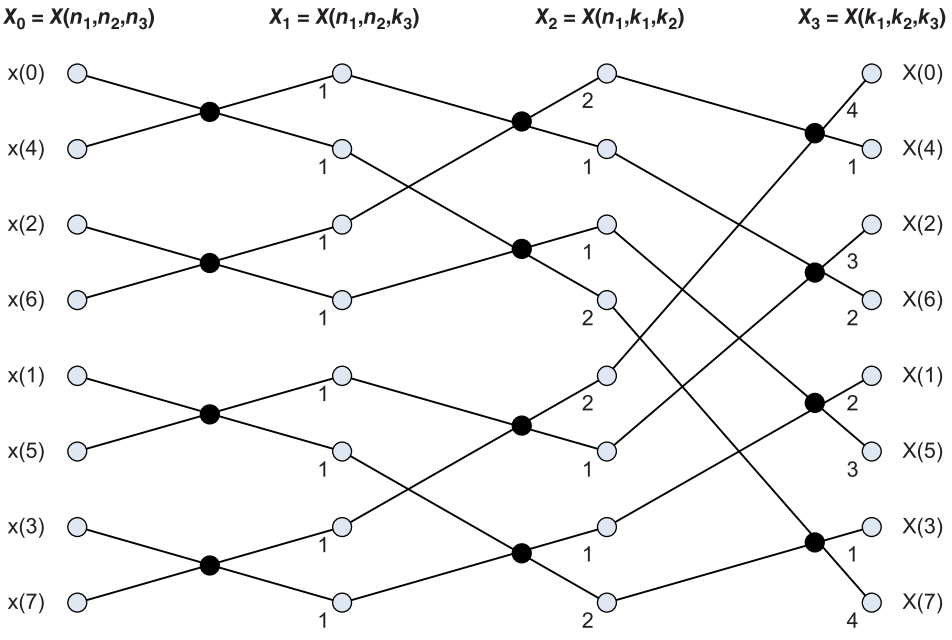


Figure 13.12 Eight-point radix-2 modified flow graph

radix which requires only N/r_m radix-r_m elements at stage m, with the elementary radix operations mapped as a small locally interconnected array.

This gives the modified version shown in Figure 13.12, where the index space separation is reduced by 28.6% from 56 to 40 for the eight-point version. The numbers on the figure represent the index space separation and sum to 40; examination of Figure 2.5 reveals that it is 56 there. It is shown that this can be generalized to a reduction from $O(N(N-1))$ to $O(N(N+\log_2 N-1)/2)$ in radix-2 for all N, representing a reduction of 40% or more for point sizes greater than 32. The range of values is given in Table 13.5.

Table 13.5 Radix-2 FFT index space separation

N	Figure 13.12	Figure 13.13	Reduction (%)
2	2	2	0
4	12	10	16.7
8	56	40	28.6
16	240	152	36.7
32	992	576	41.9
64	4032	2208	45.2
128	16,256	8,576	47.2
256	65,280	33,664	48.4
512	261,632	133,120	49.1
1024	1,047,552	528,896	49.5
2048	4,192,256	2,107,392	49.7
4096	16,773,120	8,411,136	49.9

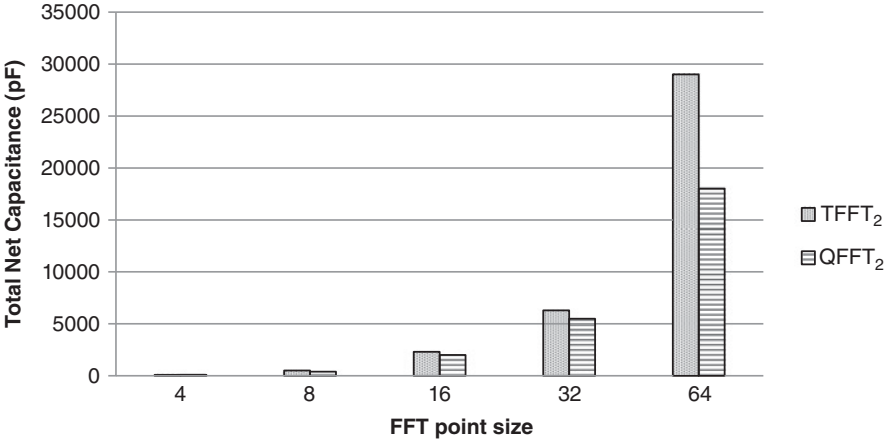


Figure 13.13 Interconnect capacitance versus point size for FFT designs in Xilinx Virtex-7 FPGA technology

The index schedule, where $n_1, n_2, n_3 = 0, 1$ and $k_1, k_2, k_3 = 0, 1$ for iterations I_1, I_2, I_3, I_4, for the in-place mapping schedule, is given by

$$\begin{aligned} x(n) &= I_1(n_1 + 2n_2 + 4n_3) \\ X(k) &= I_4(4k_3 + 2k_2 + 1k_1), \end{aligned} \tag{13.8}$$

and the out-of-place mapping schedule, given by

$$\begin{aligned} x(n) &= I_1(n_1 + 2n_2 + 4n_3) \\ X(k) &= I_4(1k_1 + 2k_2 + 4k_3), \end{aligned} \tag{13.9}$$

is shown in Figure 13.12.

Validation of the research is given in McKeown and Woods (2008). The TFFT design was described in Figure 2.5, and Figure 13.12 represents the QFFT design. The designs were synthesized using Version 11 of the Xilinx ISE tools, giving the results in Figure 13.13. The chart shows the saving in accumulated routing capacitance for QFFT design as the FFT point sizes increases. In addition, Figure 13.14 gives the detailed distribution of the number of capacitance values against sizes for a specific point size, namely a 64-point FFT. It clearly shows how the lower index space separation results in placed and routed designs with lower capacitance. Power savings of 36–37% were achieved.

13.6.4 Data Mapping

FPGA vendors have developed optimized dedicated processing elements such as the DSP48E blocks in the Xilinx FPGA families and DSP blocks in the Altera family. The data size has been predetermined to be 9–18 bits in most cases, and so it is a case of FPGA vendors trying to predetermine the largest finite word size that will work well for a range of applications. Their aim is to provide sufficient dynamic range to meet all operating conditions.

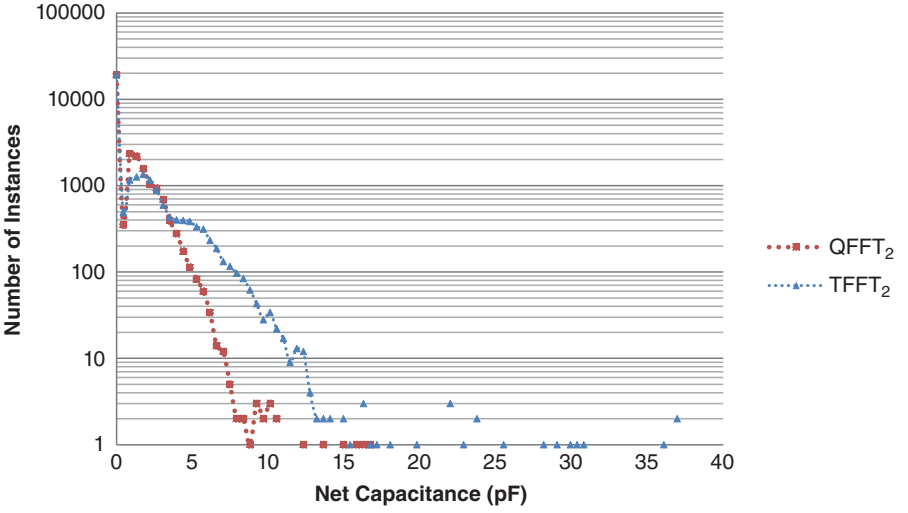


Figure 13.14 Interconnect capacitance for a 64-point FFT design implemented using Xilinx Virtex-7 FPGA technology

There are some applications, however, such as in radar applications that do not even require this dynamic range but require the processing of data at multi-GSPS. One example is in electronic warfare (EW), where radar is used to detect and classify airborne, naval and land-based threats; flexibility is needed to meet changeable tactical and mission requirements to ensure high intercept probability. Power is critical and systems typically limit the maximum functionality, resulting in a narrow and inflexible design space (McKeown and Woods 2013).

In these cases, there is an opportunity to process more than one data stream in the dedicated resources in the dynamic range, typically 18 bits in Altera and Xilinx FPGA technology. For example, the Altera DSP block supports numerous wordlengths (9 bits, etc.). It would seem a little contrived but, as the next section indicates, there is a low-precision mode which operates on the large portion of the data and a high-precision mode when an event is detected.

FFT-Based Digital Receiver

The EW receiver demonstrator (Figure 13.15) contains four distinct functions, including ADCs, conversion into the frequency domain using the FFT and frequency domain analysis with selection and detection of signals of interest. For this application, power consumption is the dominating factor and FFT is the key factor to such an extent that system functionality is designed around the power requirements of the FFT cores.

It is critical in EW environments to entirely encapsulate a pulse in as short a window as possible for time of arrival (TOA) and sensitivity reasons. This creates a clear need

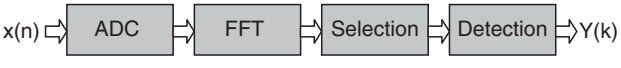


Figure 13.15 Digital receiver architecture for radar system

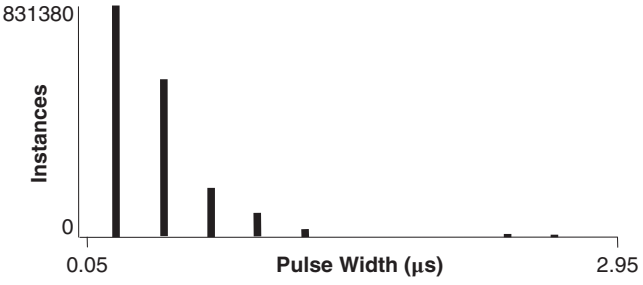


Figure 13.16 Pulse width instances: mixture of nautical and airborne radars

to run lots of FFTs of different point sizes and also to run overlapping FFT functions in real time. The time spent at each point size depends on the precise mission criteria and operating environment and ultimately determines the FPGA resources. The pulse width shown in Figure 13.16 gives the aggregated number of measured instances for various pulse widths in a typical extremely active radar environment. The majority of pulse widths are relatively short, suggesting high numbers of shorter- length FFTs with a need to perform overlapping FFTs.

To be effective, pulse envelopes must be captured in a high-resolution digital representation which requires a large dynamic range to encapsulate pulse envelope peaks during worst-case operating conditions, leading to underutilization during significant time periods. Figure 13.17 outlines the typical dynamic range of aggregated instances and shows that only 8% of samples exceed half of the dynamic range, meaning that there is 50% underutilization for 92% of the time. This is typical of this application domain as, although the radar environment is extremely active, the vast majority of peak pulses are in the sub-microsecond range.

There is a conflict between making the window large enough to minimize the probability of a pulse being split across two windows, and yet still small enough to ensure that the TOA is contained in the window. This is overcome by using the processing resources in a much more dynamic fashion for variable-length algorithms. This is achieved by exploiting the commonality in the fundamental operations of DSP algorithmic variations, mapping them to a static architecture of underlying processing elements, and using a flexible routing structure to implement the required functionality.

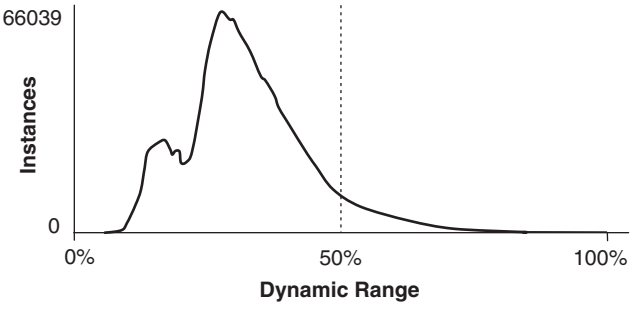


Figure 13.17 DR instances: nautical and airborne radars

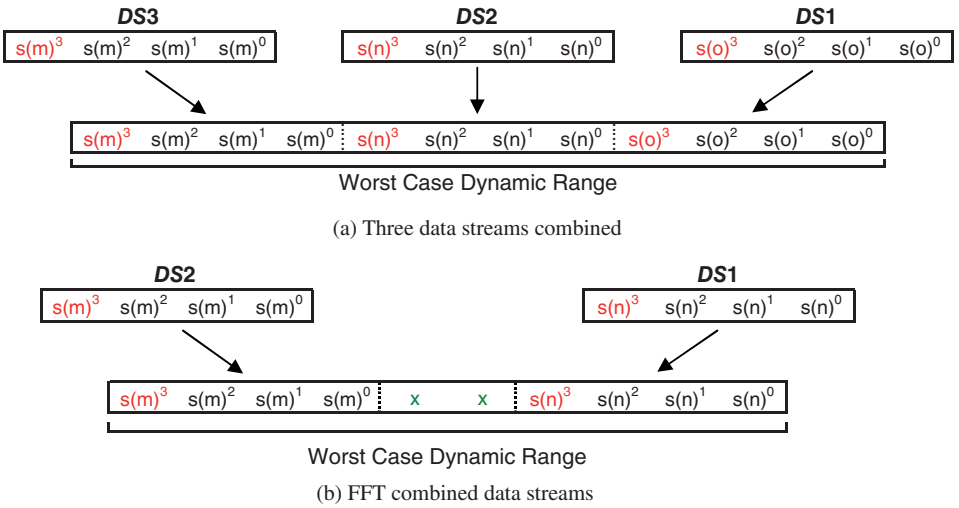


Figure 13.18 Combining data streams

Depending on the exact correlation of the standard deviation between worst-case and normal operation dynamic ranges, two or more data streams can be computed by incorporating them into a single word stream. This is shown for three input streams (DS1, DS2 and DS3) in Figure 13.18(a) and two internal streams which allows for word growth (Figure 13.18(b)). This is done by dynamically grouping processing threads to meet real-time operating requirements.

The difficulty is that while multiple data streams can be routed through some elements such as delays and multiplexers directly, others like adders and multipliers require special consideration to prevent overflow due to word growth during operations. Fixed hardware in the FPGA fabric means carry-chain modification is not possible, but an alternative approach is presented which overcomes these limitations by cleverly exploiting the existing hardware at the bit level to achieve computational separation as outlined in McKeown and Woods (2013).

13.7 Final Comments

Since the first edition of this book, power consumption has become an even more critical aspect of FPGAs. The lower power consumption of FPGA has made them highly attractive compared to their computing alternatives such as CPUs and GPUs and has largely been instrumental in the interest of such organizations such as IBM, Intel, Amazon and Microsoft in adopting the technology. Of course, FPGAs have not been immune to the problems that have driven this interest and have had to address particularly static power consumption by adopting technology, circuit and architectural solutions as covered in this chapter.

Unlike many computing alternatives, it is possible to derive the FPGA architecture to best match the algorithmic requirements and thus optimize the dynamic power consumption. This can be achieved to some extent by employing approaches to speed up the

throughput rate and then reducing the voltage to achieve reduction in power consumption, but this has limited potential as the FPGA technology has already been optimized for low-voltage operation.

A number of other approaches exist for reducing the switch capacitance. One of the most natural is to employ pipelining, as the availability of registers in the FPGA fabric makes this an obvious approach. Introducing pipeline registers has a number of attractive attributes from a power consumption perspective; it acts to reduce glitching and also the interconnection capacitance. The increase in throughput rate also enables a reduction in the amount of computation needed to provide the solution and employ hardware sharing. The chapter has also outlined some DSP-specific optimizations which have been applied in a real FFT-based application.

Bibliography

Boemo E, Oliver JP, Caffarena G 2013 Tracking the pipelining-power rule along the FPGA technical literature. In *FPGAWorld '13*, Stockholm, Sweden.

Bowman KA, Austin LB, Eble JC, Tang X, Meindl JD 1999 A physical alpha-power-law MOSFET model. *IEEE J. of Solid-State Circuits*, 32, 1410–1414.

Chandrakasan A, Brodersen R 1996 *Low Power Digital Design*, Kluwer, Dordrecht.

Chapman K, Hussein J 2012 Lowering power using the voltage identification bit. Application Note: Virtex-7 FPGAs, XAPP555 (v1.1) (accessed February 28, 2016).

Chen C-S, Hwang TT, Liu CL 1997 Low power FPGA design – a re-engineering approach. In *Proc. of Design Automation Conf.*, pp. 656–661.

Choi S, Prasanna VK 2003 Time and energy efficient matrix factorization using FPGAs. In *Proc. Int. Conf. on Field Programmable Logic and Applications*, pp. 507–519.

Chow CT, Tsui LSM, Leong PHW, Luk W, Wilton S 2005 Dynamic voltage scaling for commercial FPGAs. In *Proc. Int. Conf. on Field-Programmable Technology*, pp. 215–222.

Cooley JW, Tukey JW 1965 An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19, 297–301.

Curd D 2007 Power consumption in 65 nm FPGAs. White Paper: Virtex-5 FPGAs. Available from www.xilinx.com (accessed February 28, 2016).

Erdogan AT, Arslan T 2000 High throughput FIR filter design for low power SoC applications. In *Proc. 13th Annual IEEE Conf. on ASIC/SOC*, pp. 374–378.

Erdogan AT, Arslan T 2002 Implementation of FIR filtering structures on single multiplier DSPs. *IEEE Trans. on Circuits and Systems II*, 49(3), 223–229.

Gonzalez R, Gordon B, Horowitz M 1997 Supply and threshold scaling for low power CMOS. *IEEE J. of Solid-State Circuits*, 32(8), 1210–1216.

Huda S, Mallick M, Anderson JH 2009 Clock gating architectures for FPGA power reduction. In *Proc. Int. Conf. on Field Programmable Logic and Applications*, pp. 112–118.

ITRS 2003 International Roadmap for Semiconductors. Available from http://public.itrs.net (accessed February 28, 2016).

Keane G, Spanier JR, Woods R 1999 Low-power design of signal processing systems using characterization of silicon IP cores. In *Proc of 33rd Asilomar Conference on Signals, Systems and Computers*, pp. 767–771.

Kim NS, Austin T, Baauw D, Mudge T, Flautner K, Hu JS, Irwin MJ, Kandemir M, Narayanan V 2003 Leakage current: Moore's law meets static power. *IEEE Computer*, 36(12), 68–75.

Kung HT, Leiserson CE 1979 Systolic arrays (for VLSI). In *Proc. on Sparse Matrix*, pp. 256–282.

Kung SY 1988 *VLSI Array Processors*. Prentice Hall, Englewood Cliffs, NJ.

Lamoureux J, Wilton SJE 2003 On the interaction between power-aware FPGA CAD algorithms. In *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*.

McKeown S, Woods R 2008 Algorithmic factorisation for low power FPGA implementation through increased data locality. In *Proc. of IEEE Int. Symp. VLSI Design, Automation and Test*, pp. 271–274.

McKeown S, Woods R 2011 Low power FPGA implementation of fast DSP algorithms: Characterisation and manipulation of data locality. *IET Proc. on Computer and Design Techniques*, 5(2), 136–144.

McKeown S, Woods R 2013 Power efficient FPGA implementations of transform algorithms for radar-based digital receiver applications. *IEEE Trans. on Industrial Electronics*, 9(3), 1591–1600.

Mead C, Conway L 1979 *Introduction to VLSI Systems*. Addison-Wesley, Reading, MA.

Nunez-Yanez J 2015 Adaptive voltage scaling with in-situ detectors in commercial FPGAs. *IEEE Trans. on Computers*, 641, 45–53.

Nunez-Yanez J, Chouliaras V, Gaisler J 2007 Dynamic voltage scaling in a FPGA-based system-on-chip. In *Proc. Int. Conf. on Field Programmable Logic and Applications*, pp. 459–462.

Parashar M, Browne JC 2000 Systems engineering for high performance computing software: The HDDA/DAGH infrastructure for implementation of parallel structured adaptive mesh refinement. In Baden SB, Chrisochoides NP, Gannon DB, Norman ML (eds) *Structured Adaptive Mesh Refinement (SAMR) Grid Methods*, IMA Volumes in Mathematics and its Applications, 117, pp. 1–18. Springer, New York

Raghunathan A, Dey S, Jia NK 1999 Register transfer level power optimization with emphasis on glitch analysis and reduction. *IEEE Trans. Computer Aided Design*, 18(8), 114–131.

Roy K, Mukhopadhyay S, Meimand H 2003 Leakage current mechanisms and leakage reduction technqiues in deep-submicron CMOS circuits. *Proc. IEEE*, 91(2), 305–327.

Ryan J, Calhoun B 2010 A sub-threshold FPGA with low-swing dual-VDD interconnect in 90nm CMOS. In *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 1–4.

Shen A, Kaviani A, Bathala K 1992 On average power dissipation and random pattern testability of CMOS combinational logic networks. In *IEEE Int. Conf. on Computer Aided Design*, pp. 402–407.

Wilton SJE, Luk W, Ang SS 2004 The impact of pipelining on energy per operation in field-programmable gate arrays. *Proc. of Int. Conf. on Field Programmable Logic and Applications*, pp. 719—728.

Wolf W 2004 *FPGA-Based System Design*. Prentice Hall, Upper Saddle River, NJ.

Woods RF, McCanny JV, McWhirter JG 2008 From bit level systolic arrays to HDTV processor chips. *J. of VLSI Signal Processing*, 53(1–2), 35–49.