The material in this book shows core ideas of DSP architecture design. Techniques presented can aid in extracting performance, energy and area efficiency in future applications. New ideas will also have to emerge to solve future problems. In the future, we feel, the main design problem will be how to achieve hardware flexibility and energy efficiency simultaneously.

We can no longer rely on technology scaling to improve energy efficiency. The plot on the slide shows energy efficiency in GOPS/mW, which is how many billion operations per second you can do in one milliwatt of power, versus technology generation. Such energy efficiency represents the intrinsic computational capability of silicon technology. The numbers on the horizontal axis represent channel length ($L$).

In the past (e.g. 1990s), the number of digital computations per unit energy greatly improved with smaller transistors and lower operating voltage ($V_{DD}$). Things are now different. Energy efficiency is tapering off with scaling. Scaling of voltage has to slow down due to increased leakage and process variation, which results in reduced energy efficiency according to the formula. In the future, the rate of shrinking the channel length will be delayed due to increased development and manufacturing cost.

Technology scaling, overall, is no longer providing benefits in energy efficiency as in the past. This change in technology greatly emphasizes the need for energy-efficient design.
Applications are getting more complex. Increased functional diversity has to be supported on the same device. The amount of data processing for adding new features is exploding. Let’s take a recent (2010) example of an iPad. When Apple decided to put H.264 decoder on iPad, they had to use specialized hardware to meet the energy efficiency requirements. Software was not an option, because it would have consumed too much power. Future applications will be even more constraining. In applications such as high-speed wireless, multi-antenna and cognitive radio or mm-wave beamforming, software solutions wouldn’t even meet real-time requirement regardless of power. Software solutions wouldn’t even be an option. There are numerous other applications where real-time throughput and energy efficiency have to come from specialized hardware.

Adding too many pieces of specialized hardware, however, to support a diverse set of features would not be effective. Future designs must be energy efficient and flexible at the same time.

There are two ways to provide flexibility. We can use a programmable DSP processor and develop application-specific software. People have been doing that for a long time. This approach works well for low-throughput and ad-hoc operations, but falls short on delivering the performance and energy efficiency required from high-throughput applications such as high-speed wireless.

Alternatively, we could use an FPGA, which is a reconfigurable hardware that you customize each time you need to execute a new algorithm. Unlike programmable DSP where your hardware is fixed, now you have uncommitted resources, which you can configure to support large degrees of parallelism and provide very high throughput.
Since programmable DSPs can’t deliver the performance and efficiency for high-throughput applications, as we’ve discussed, we need to drive up efficiency of reconfigurable hardware towards the upper-right corner. It would be great if we could possibly eliminate the need for dedicated chips.

That’s why dedicated chips still use 90nm as the preferred technology. On the other hand, FPGA companies exploit the most advanced technology available. Due to their regularity, the development cost is not as high. So, if we retain the efficiency benefits of dedicated chips without giving up the flexibility benefits of FPGAs, that would be a revolutionary change.
FPGAs are energy inefficient, because of their interconnect architecture. This slide shows a small section of an FPGA chip representing key FPGA building blocks. The configurable logic block (CLB) consists of look-up table (LUT) elements that can be configured to do arbitrary logic. The switch-box array consists of bi-directional switches that can be configured to establish connections between CLBs.

The architecture shown on the slide is derived from $O(N^2)$ complexity, where $N$ represents the number of logic blocks. Clearly, full connectivity cannot be supported, because the number of switches would outpace Moore’s law. In other words, if the number of logic elements $N$ were to double, the number of switches $N^2$ would quadruple. This is why FPGAs never have full connectivity.

Depopulation and segmentation are two techniques that are used to manage connectivity. The switch-box array shown on the slide would have $12 \times 12$ switches for full connectivity, but only a few diagonal switches are provided. This is called depopulation. When two blocks that are physically close are connected, there is no reason to propagate electricity down the rest of the wire, so the wire is then split into segments. This technique is called segmentation. Both of the techniques are used heuristically to control connectivity. As a result, it is nearly impossible to fully utilize an FPGA chip without routing congestion and/or performance degradation.

Despite reduced connectivity, FPGA chips still have more than 75% of chip area allocated for the interconnect switches. The impact on power is also quite significant: interconnect takes up about 60% of the total chip power.
To improve energy efficiency, hierarchical networks have been considered. Two representative approaches, tree of meshes and butterfly fat tree, are shown on the slide. Both networks have limited connectivity even at local levels and also result in some form of a mesh.

Consider the tree of meshes, for example. Locally connected groups of 4 PEs are arranged in a network. As you can see, each PE has 3 wires. We would then need $4 \times 3 = 12$ switches, while only 6 are available. This means 50% of connectivity even at the lowest level. Also, the complexity of the centralized mesh grows quickly.

Butterfly fat tree attempts to provide more dedicated connectivity at each level of hierarchy, but still results in a large central switch. We, again, have very similar problem as in 2D mesh: new levels of hierarchy use centralized global resources for routing. Dedicated resources would be more desirable for new levels of hierarchy. This is a critical problem to address in the future in order to provide energy efficiency without giving up the benefits of flexibility.

In summary, we are near the end of CMOS scaling for both technical and economic reasons. Energy efficiency is tapering off, design cost is going up. We must investigate architecture efficiency in light of these challenges.

Applications are getting more complex and the amount of digital signal processing is growing rapidly. Future applications, therefore, require energy efficient flexible hardware. Architecture of the interconnect network is crucial for providing energy efficiency.

Design problem of the future, therefore, is how to simultaneously achieve hardware flexibility and energy efficiency.
References

- M.C. Chian, Int. Symp. on FPGAs, 2009, Evening Panel: CMOS vs. NANO, Comrades or Rivals?


<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute-value thresholding                                         16.6</td>
<td></td>
</tr>
<tr>
<td>Adaptive algorithms                                                 6.38</td>
<td></td>
</tr>
<tr>
<td>Adaptive equalization                                               7.40</td>
<td></td>
</tr>
<tr>
<td>Adjacent-channel power ratio                                         13.33</td>
<td></td>
</tr>
<tr>
<td>Alamouti scheme                                                     14.7</td>
<td></td>
</tr>
<tr>
<td>ALAP scheduling                                                     11.30</td>
<td></td>
</tr>
<tr>
<td>Algorithm-to-hardware flow                                           12.7</td>
<td></td>
</tr>
<tr>
<td>Aliasing                                                            13.20</td>
<td></td>
</tr>
<tr>
<td>Alpha-power model                                                   1.18, 1.19</td>
<td></td>
</tr>
<tr>
<td>ALU                                                                 2.19</td>
<td></td>
</tr>
<tr>
<td>Architecture-circuit co-design                                      2.27</td>
<td></td>
</tr>
<tr>
<td>Architecture flexibility                                            4.2</td>
<td></td>
</tr>
<tr>
<td>Architecture folding                                                15.15</td>
<td></td>
</tr>
<tr>
<td>(also see folding)</td>
<td></td>
</tr>
<tr>
<td>Architecture variables                                             2.24</td>
<td></td>
</tr>
<tr>
<td>Architecture optimization                                           2.25, 12.20</td>
<td></td>
</tr>
<tr>
<td>Architectural transformations                                       3.23, 11.2</td>
<td></td>
</tr>
<tr>
<td>Area-energy-performance tradeoff                                    3.28</td>
<td></td>
</tr>
<tr>
<td>Area efficiency                                                     4.19, 4.28, 15.46</td>
<td></td>
</tr>
<tr>
<td>Area per operation                                                  4.13</td>
<td></td>
</tr>
<tr>
<td>Array multiplier                                                    5.23</td>
<td></td>
</tr>
<tr>
<td>ASAP scheduling                                                     11.27</td>
<td></td>
</tr>
<tr>
<td>Asynchronous clock domains                                          12.41</td>
<td></td>
</tr>
<tr>
<td>BEE2 system blockset                                                12.32</td>
<td></td>
</tr>
<tr>
<td>Bellman-Ford algorithm                                              11.12, 11.40</td>
<td></td>
</tr>
<tr>
<td>BER performance                                                     15.22</td>
<td></td>
</tr>
<tr>
<td>Binary-point alignment                                              10.4</td>
<td></td>
</tr>
<tr>
<td>Blind-tracking SVD algorithm                                        14.16</td>
<td></td>
</tr>
<tr>
<td>Block RAM                                                           12.36</td>
<td></td>
</tr>
<tr>
<td>Brain-machine interface                                             16.4</td>
<td></td>
</tr>
<tr>
<td>Branching unit                                                      3.10</td>
<td></td>
</tr>
<tr>
<td>Billion instructions per second (BIPS)                               3.12</td>
<td></td>
</tr>
<tr>
<td>BORPH operating system                                              12.39</td>
<td></td>
</tr>
<tr>
<td>Butterfly fat tree                                                  17.8</td>
<td></td>
</tr>
<tr>
<td>Carrier multiplication                                              13.18</td>
<td></td>
</tr>
<tr>
<td>Carry-save adders                                                   8.25</td>
<td></td>
</tr>
<tr>
<td>Carry-save arithmetic                                               5.26, 12.25</td>
<td></td>
</tr>
<tr>
<td>Cascaded CIC filter                                                 13.22</td>
<td></td>
</tr>
<tr>
<td>Cascaded IIR filter                                                 7.29</td>
<td></td>
</tr>
<tr>
<td>Characterization                                                    2.29</td>
<td></td>
</tr>
<tr>
<td>CIC filter                                                          7.37, 13.19</td>
<td></td>
</tr>
<tr>
<td>Circuit optimization                                                1.27, 2.2, 2.21</td>
<td></td>
</tr>
<tr>
<td>Circuit variables                                                   2.24</td>
<td></td>
</tr>
<tr>
<td>Classification accuracy                                             16.19, 16.34</td>
<td></td>
</tr>
<tr>
<td>Clock gating                                                        15.14</td>
<td></td>
</tr>
<tr>
<td>Clustering                                                          16.5</td>
<td></td>
</tr>
<tr>
<td>Code compression                                                    7.56, 7.57</td>
<td></td>
</tr>
<tr>
<td>Cognitive radio                                                     13.3</td>
<td></td>
</tr>
<tr>
<td>Common sub-expressions                                              5.32</td>
<td></td>
</tr>
<tr>
<td>Complexity-performance tradeoff                                     16.18</td>
<td></td>
</tr>
<tr>
<td>Computational complexity                                            8.17</td>
<td></td>
</tr>
<tr>
<td>Configurable logic block                                            17.7</td>
<td></td>
</tr>
<tr>
<td>Connectivity                                                        17.8</td>
<td></td>
</tr>
<tr>
<td>Convergence                                                         6.28, 6.29, 6.38</td>
<td></td>
</tr>
<tr>
<td>Cooley-Tukey mapping                                               8.9, 8.10</td>
<td></td>
</tr>
<tr>
<td>Coordinate translation                                              6.8</td>
<td></td>
</tr>
<tr>
<td>CORDIC                                                               6.3</td>
<td></td>
</tr>
<tr>
<td>CORDIC iteration                                                    6.7</td>
<td></td>
</tr>
<tr>
<td>CORDIC rotation                                                     6.5</td>
<td></td>
</tr>
<tr>
<td>Cork function                                                       8.39</td>
<td></td>
</tr>
<tr>
<td>Correlation                                                         5.34</td>
<td></td>
</tr>
<tr>
<td>Cost of design                                                      17.6</td>
<td></td>
</tr>
<tr>
<td>Cost of flexibility                                                 4.4</td>
<td></td>
</tr>
<tr>
<td>Covariance matrix                                                   16.11</td>
<td></td>
</tr>
<tr>
<td>Critical path                                                       3.8</td>
<td></td>
</tr>
<tr>
<td>Critical-path delay                                                3.3</td>
<td></td>
</tr>
<tr>
<td>Cut-sets                                                            7.17, 11.7</td>
<td></td>
</tr>
<tr>
<td>Cycle latch                                                         2.20</td>
<td></td>
</tr>
<tr>
<td>Data path                                                           3.3</td>
<td></td>
</tr>
<tr>
<td>Data-rate reduction                                                 16.34</td>
<td></td>
</tr>
<tr>
<td>Data-stream interleaving                                            3.17, 14.19, 15.13</td>
<td></td>
</tr>
<tr>
<td>Data-flow graph                                                     9.4, 9.7-9, 12.13</td>
<td></td>
</tr>
<tr>
<td>Decimation filter                                                   7.35, 7.36, 13.12</td>
<td></td>
</tr>
<tr>
<td>Decimation in frequency                                             8.13</td>
<td></td>
</tr>
<tr>
<td>Decimation in time                                                  8.12, 8.13</td>
<td></td>
</tr>
<tr>
<td>Decision-feedback equalizers                                        7.46, 7.47</td>
<td></td>
</tr>
<tr>
<td>Dedicated chip                                                      4.8, 4.17</td>
<td></td>
</tr>
</tbody>
</table>
Delay line 15.44 (continued)
Denormals 5.7
Depopulation 17.7
Depth-first search 15.10
DFT algorithm 8.5
Digital front end 13.5
Digital mixer 13.17
Dimensionality reduction 16.5
Direct-form FIR filter 7.14
Discrete derivatives 16.14
Discrete Fourier transform (DFT) 8.2
Discrete wavelet transform 8.40, 16.13
Distributed arithmetic 7.50, 7.51, 7.52
Diversity gain 14.6
Diversity-multiplexing tradeoff 14.6, 14.10
Divide-and-conquer approach 8.6
Division 6.19
Down-sampling 13.7, 13.12
Down-sampler implementation 8.44
DSP chip 4.15
Dyadic discrete wavelet series 8.38

Eigenvalues 14.17, 14.30
Eigenvalue decomposition 16.11
Eigenvectors 14.17
Electronic interchange format (EDIF) 12.4
Electrophysiology 16.4
Energy 1.3, 1.4
Energy model 2.3
Energy-area-performance space 12.21
Energy-delay optimization 1.26, 2.12
Energy-delay product (EDP) 1.23
Energy-delay space 3.14
Energy-delay sensitivity 2.5
Energy-delay tradeoff 1.22, 1.25-27, 12.24
Energy efficiency 4.6, 14.26, 14.28, 15.25, 17.2
Energy per bit 15.24, 15.29
Energy per channel 16.27
Energy per operation 3.6
Equalizer 7.40, 7.42
Error vector magnitude (EVM) 13.37
Ethernet link 12.38
Euclidean distance 14.11, 15.6, 15.20

Farrow structure 13.27
Feature extraction 16.5, 16.6
Feedback loops 7.30, 7.32

FFT energy estimation 8.29
FFT timing estimation 8.28
Fine-grain pipelining 11.18
FIR filter 7.13, 12.25
Fixed point 10.3, 10.4
Fixed-point unit 3.10
Fixed-point arithmetic 5.2
Flexibility 15.24
Flexible radio architecture 14.2
Floating-point unit 3.10
Floating-point arithmetic 5.2
Flip-flop 2.20
Floating-to-fixed-point conversion 10.5
Floorplan 5.27
Floyd-Warshal algorithm 11.12
Folding 3.20, 3.22, 3.26, 11.36, 14.20, 14.25
Fourier transform 8.3
FPGA 12.3, 12.30, 12.38, 17.4
Fractional decimation 13.24
Fractionally-spaced equalizer 7.48
Frequency-domain representation 8.3
Functional verification 12.30

Gain error 6.12
Gate capacitance 1.12
Gate sizing 14.25
GOPS/mm² 14.26, 14.28
GOPS/mW 14.26, 14.28
GPIO blocks 12.33
Graphical user interface (GUI) 12.12
Gray coding 15.6, 15.9

H.264 decoder 17.3
Haar wavelet 8.42, 8.43
Hardware emulation 12.5
Hardware-in-the-loop tools 12.43
Hardwired logic 2.23
Hard-output sphere decoder 15.27
Hierarchical loop retiming 14.23
High-level architectural techniques 12.22
High-level retiming 7.18
High-throughput applications 17.4

Ideal filter 7.8
IEEE 754 standard 5.4
IIR filter architecture 7.26
IIR filter 7.23, 7.24, 7.25
Image compression 8.48
Implantable neural recording  16.17
(continued)
Input port  9.13
Interconnect architecture  17.7
Interpolation  7.35, 7.37
Initial condition  6.26, 6.31, 6.34, 6.38
Integer linear programming (ILP)  12.46-8
Interleaving  3.22, 3.26, 14.25, 16.26
Inter-iteration constraint  9.8, 11.35
Interpolation  13.25
Inter-symbol interference  7.11, 7.46
Interleaving  3.22, 3.26, 14.25, 16.26
Inter-iteration constraint  9.8
Integral transform  16.15
Inverter chain  2.10, 2.11, 2.12
Iteration  6.6, 6.33, 9.2
Iteration bound  11.17
Iterative square root and division  6.22
Jitter compensation  10.25
K-best search  15.10
Kogge-Stone tree adder  2.13
Latency vs. cycle time tradeoff  12.24
Leakage energy  1.6, 1.13
Leakage power  1.5
Least-mean square equalizer  7.43, 7.44
Leiserson-Saxe Algorithm  11.12
LIST scheduling  11.31
LMS-based tracking  14.17
Load/shift unit  3.10
Logarithmic adder tree  7.19
Logic depth  3.12, 3.15
Long-term evolution (LTE)  13.10
Look-up table  17.7
Loop retiming  14.21
Mallat’s wavelet  8.40
Master-slave latch  2.20
Maximum likelihood detection  14.10
Median choice probability  16.22
Medical implants  16.2
Memory decoder  2.10
Memory  5.35
Memory code compression  7.56, 7.58
Memory partitioning  7.55
Metric calculation unit  15.12
Metric enumeration  15.16
Microprocessor  2.23, 4.8
MIMO communication  14.5
MIMO transceiver  14.18
Minimum delay  2.7, 2.9
Minimum energy  1.17, 3.15, 16.25
Minimum EDP  1.24, 3.6
Mixed-radix FFT  8.18
MODEM  7.4
MOPS  4.5
MOPS/mm²  4.19
Morlet wavelet  8.37
MSE error  10.9
MSE-specification analysis  10.22
Multi-core search algorithm  15.13
Multiplier  5.22, 5.23, 5.25
Multi-path single-delay feedback  15.30
Multi-rate filters  7.35
Multidimensional signal processing  14.1
Neural data  16.32
Newton-Raphson algorithm  6.24, 14.17
NMOS header  16.28
Noise shaping  13.15
Nonlinear energy operator  16.8
Numerical complexity  8.18
Nyquist criterion  13.11
Off-path loading  2.10
Operation  4.5
Optimal $E_{1x}/E_{sw}$  3.15
Optimization flow  12.8
Out-of-band noise  13.20
Overflow  5.14, 5.15
Oversampling  13.9
Parallelism  3.2, 3.4, 3.14, 3.24, 11.19, 12.21, 15.40
Parallel architecture  12.18
Parasitic capacitance  1.11, 1.12
Path reconvergence  2.10
Perturbation theory  10.9
Phase detector  13.28
Pipelineing  3.2, 3.8, 3.14, 3.24, 7.16, 11.14
Polyphase implementation  8.45, 8.46
Post-cursor ISI  7.46, 7.47
Power  1.3, 1.4
Power density  16.2
Power gating  16.28
Power-delay product (PDP) 1.23 (continued)
PowerPC 4.15
Pre-cursor ISI 7.46, 7.47
Pre-processed retiming 11.41
Principal component analysis 16.11
Probability of detection 16.19
Probability of false alarm 16.19
Programmable DSP 2.23, 4.8

QAM modulation 15.22
Quadrature-mirror filter 8.41
Quantizer 5.16
Quantization noise 5.21, 13.31
Quantization step 5.17

Radio design 7.6
Radio receiver 7.5
Radix 2 8.12
Radix-2 butterfly 8.15, 8.25
Radix-4 butterfly 8.16
Radix-16 factorization 15.34
Radix factorization 15.41
Raised-cosine filter 7.3, 7.10, 7.11
Real-time verification 12.30, 12.41
Receiver operating characteristic 16.21
Reconfigurable FFT 15.30
Recursive flow graph 11.16
Region partition search 15.8
Register bank 15.18
Register file 15.44
Repetition coding 14.14
Resource estimation 10.12, 10.21
Retiming 3.30, 7.18, 7.31, 11.3-6, 12.21, 12.48
Retiming weights 11.10
Ripple-carry adder 5.24
ROACH board 12.38
Roll-off characteristics 7.8
Rotation mode 6.14
Rounding 5.6, 5.8, 5.18
RS232 link 12.36

Sampling 5.16
Saturation 5.19
Schedule table 12.17
Scheduling 11.23, 11.40, 12.48
Schnorr-Euchner enumeration 15.7
Search radius 14.12, 15.19

Segmentation 17.7
Sensitivity 2.6
Gate sizing 2.11
Short-time Fourier transform (STFT) 8.33
Sigma-delta modulation 13.14, 13.31-32
Sign magnitude 5.12
Signal band 15.26
Signal bandwidth 7.7
Signal-flow graph (SFG) 7.15, 8.14, 9.4
Signal-to-quantization-noise ratio (SQRN) 5.21
Simulink block diagram 9.11
Simulink design framework 12.4
Simulink libraries 9.12
Simulink test bench model 12.33
Singular value decomposition (SVD) 14.3, 14.9
Sizing optimization 2.11
Soft-output decoder 15.29
Spatial-multiplexing gain 14.6
Specification marker 10.16
Spectral leakage 7.4
Sphere decoding algorithm 14.12
Spike alignment 16.5
Spike detection 16.5
Spike sorting 16.4, 16.5, 16.6
Spike-sorting ASIC 16.18, 16.31
Spike-sorting DSP 16.17, 16.31
Split-radix FFT (SRFFT) 8.22

Stationary wavelet transform product 16.9
Step size 7.45
Square-root raised-cosine filter 7.13
SRAM 15.44
Sub-threshold leakage current 1.14, 1.15
Superscalar processor 3.9
Supply voltage scaling 1.16, 14.25
Switching activity 1.9, 5.33
Switched capacitance 1.11, 4.5
Switching energy 1.6, 1.7, 1.10, 2.4
Switching power 1.5
Subcarriers 15.14, 15.20
Sub-Nyquist sampling 13.13
SynplifyDSP 9.12

Tap coefficients 7.45
Taylor-series interpolation 13.27
Technology scaling 17.2 (continued)
Index

Time multiplexing 3.2, 3.16, 3.25, 12.15, 12.21
Time-invariant signal 8.3
Time-interleaved ADC 13.16
Time-frequency resolution tradeoff 8.34
Timing constraint 11.13
Throughput 3.6
Training phase 16.31
Transition probability 5.34
Transformed architecture 12.16
Transmit filter 7.4
Transposed FIR filter 7.21
Transposition 9.5
Tree adder 2.10, 2.13
Tree of meshes 17.8
Tree search 14.12
Truncation 5.18
Tuning variables 2.14
Twiddle factors 8.19, 8.26, 15.30
Two’s complement 5.10

Ultra-wide band (UWB) 12.27
Unfolding 7.32, 11.20-22, 12.26
Up-sampling filter 13.35
Unsigned magnitude 5.11

V-BLAST algorithm 14.7
Vectoring mode 6.14
Virtex-II board 12.37
Voltage scaling 12.21

Wallace-tree multiplier 5.28, 5.29
Wavelet transform 8.35, 8.36
WiMAX 13.10
Windowing function 8.34
Wordlength analysis 10.18
Wordlength grouping 10.20
Wordlength optimization 10.2, 14.18, 14.25, 16.29
Wordlength optimization flow 10.14
Wordlength range 10.16
Wordlength reader 10.17
Wrap-around 5.19

Xilinx system generator (XSG) 9.12

ZDOK+ connectors 12.35, 12.37, 15.21
Zero-forcing equalizer 7.42