why FPGAs for dsp?
santa clara university

dr chris dick
dsp chief architect
wireless and signal processing group
xilinx inc.

Moore’s Law
Electronics, Volume 38 No. 8, April 19 1965

Moore’s Law plot from original 1965 paper
**Moore's Law Redux**

- Increased capacity
- Improved QoS
- Media-centric services
- Reduced cost to subscriber
- Maximally leverage system degrees of freedom
- Space: MIMO, STBC, Beam Forming
- Time: Turbo, LDPC
- Frequency: OFDM
- Integrated Design Flows – SoC Dev/Ver
- DSP to cost reduce RF
goingmore
- CFR, DPD

**Cost Constraints**

- Deliver low-cost media-rich services to end-user
- Cost of infrastructure under extreme pressure
  - Reduce CAPX
  - Reduce OPEX

*why FPGA's for DSP?*
**Schedule Constraints**

- TTM critical
  - First to market wins
- Standards evolving at a greater pace than we have ever experienced
  - Need to deliver increased complexity in shorter time

**Form Factor Constraints**

- Need to insert more technology into space occupied by existing installation
- Reduce OPEX
  - Cost of land occupied by basestation significant component of operating cost
why fpga’s for dsp?   7

“Last year, more transistors were produced – and at a lower cost – than grains of rice.”

Semiconductor Industry Association Annual Report 2005
why FPGA's for DSP?   9

ITRS Forecasts No End

Moore’s Law

Source: Intel
Speed Has Grown Respectably but has hit a ceiling

Aren’t software processors improving with Moore’s law?

Primary means of performance increase of software processors is by increasing clock rate.
Power Density

- Nuclear Reactor
- Toaster Oven
- Hot Plate
- Pentium® proc

Technology Trends/Future Compute Requirements

"Shannon's Law"
Drivers for Next Generation Wireless Systems

| High Data Rates at High Speed | Bit rates: 50(UL), 100Mbps (DL)  
Speed: Walking to bullet-train |
| Reduced Cost/GByte            | Higher system capacity  
Lower cost/GByte               |
| Reduced Latency              | Quick response time     |
| Optimized for packet-switching| Better support for VoIP & data |
| Cost Efficient Roll-out      | Reuse 3G/2G spectrum  
Bandwidth flexibility  
Minimum Frequency planning |

image from eurosoutheastasia-ict.org

why fpga’s for dsp?  15

FPGAs are

- extremely powerful high-performance computing platforms
- flexibility is a key strength and value proposition
  - one device can address multiple applications in multiple markets
- look at the fpga as a state-of-the-art embedded computing platform
- can integrate a lot of requirements
  - compute
    - signal processing: wireless, modems, radar, medical imaging,…
  - connectivity
    - PCI express, serial rapid IO (SRIO), …
  - embedded processing
    - TCP/IP stacks, real-time operating system (RTOS), application code,…
 Agenda

- What are the traditional (conventional) technologies employed to implement real-time digital signal processing (DSP)?
- What are the limitations of these technologies?
- How does the FPGA address the deficiencies of these technologies?
- Establish a conceptual framework for understanding FPGA signal processing

FPGA Architecture (1)

- Generic FPGA architecture consists of an array of logic tiles
- Tile typically consists of
  - lookup table(s)
  - register(s)
  - multipliers/multiply-accumulate unit (MAC)
- Routing resources in the channels between the logic tiles provide the connectivity between tiles, I/O, on-chip memory & other resources
why fpga’s for dsp? 19

Quick Look at a New Generation FPGA
details will be covered in the fpga architecture section of course

In addition to the LUT/FF elements shown on the previous slides new generation FPGAs have a richer set of tiles for supporting arithmetic (DSP) and gigabit connectivity.
FPGA: A Heterogeneous Computing Platform

- Modern FPGAs are huge
- They have a zoo of different blocks
  - including embedded processors (hard/soft)

- 1985
  - 128 4-LUTs
- 2009
  - 759000 LCs
  - 2016 ALUs

Example FPGA Application/Software Defined Radio
The multimode basestation becomes a reality

- Large number of streams
- High sample rates
- Low ops/sample
- MPY/MAC intensive
- High compute requirements

- Large number of streams
- Low (ish) sample rates
- High data rates
- Large ops/sample
- Not always MPY/MAC intensive
- High compute requirements

- eNode-B protocol stack
- IP network stack
- System monitoring/control
- ISA-style compute … maybe with embedded acceleration

---

Figure reproduced from (and with permission of) SDR forum: www.sdrforum.org
Application Example

- Examine an application to provide some context for the questions on the previous slide
  - Finite impulse response (FIR) filter
- Digital filters form the workhorse of many DSP applications including
  - Cable modems
  - Satellite modems
  - Microwave links
  - Adaptive filters
  - 2-D FIR filters employed in image/video processing
  - ...

Digital Filter: Frequency Domain

- $|H(e^{j\Omega})|$
- $1 + \delta_1$
- $1 - \delta_1$
- PASSBAND
- TRANSITION BAND
- STOPBAND
- $\delta_2$
- $\Delta f$
- $f_p$
- $f_s$
- $f$
why FPGAs for DSP?

### FIR Filter Signal Flowgraph

\[
\begin{align*}
x(n) & \rightarrow z^{-1} & a_0 & \rightarrow a_1 & \rightarrow a_2 & \rightarrow a_{N-2} & \rightarrow a_{N-1} & \rightarrow y(n) \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow y(n)
\end{align*}
\]

\[
y(n) = a_0 x(n) + a_1 x(n-1) + \ldots + a_{N-1} x(n-(N-1)) \\
= \sum_{i=0}^{N-1} a_i x(n-i)
\]

\[
H(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{N-1} a_i z^{-i}
\]

### Implementing Real-Time DSP

- **Conventional approaches**
  - DSP microprocessor
  - Application Specific Standard Part (ASSP)
    - e.g.:
      - Digital down-converter in a modem
      - 2-D filter chip in an image processing system
  - Application Specific Integrated Circuit (ASIC)
Implementing a FIR Filter (1)

- Basic computational element require fir FIR filter is a multiply-accumulate (MAC) engine

Implementing a FIR Filter (2)

- Time shared (multiplexed) MAC unit is a common approach, used by
  - DSP microprocessors
  - General purpose processors (GPPs)
  - ASSPs
  - ASICs
- That is, the datapath is constructed using a single MAC, and this single functional unit is utilized to compute all of the N MACs required to realize an N-tap FIR filter
Traditional DSP Architectures

- **Von Neumann architecture**
  - Address generator
  - Address Bus
  - I/O Devices
  - ALU
  - Multiplier
  - Product register
  - Program and data Memory
  - Data Bus

- **Harvard architecture**
  - X data Address Bus
  - X data Address Bus
  - Program Address Bus
  - X data Memory
  - X data Memory
  - Program Memory
  - ALU
  - Product register
  - I/O devices
  - X data Address Bus
  - X data Address Bus
  - Program Address Bus

- **Severe limitations**
  - von Neumann bottleneck
  - Functional unit type
  - Number of functional units
  - Functional unit precision
  - Interconnectivity between functional units

Time Shared MAC

- One MAC unit is shared between the required N computations
- N clock cycles required to compute an output sample \( y(n) \)
- Finite state machine (FSM) generates control and address sequences

- **Input Data** \( x(n) \)
- **Data or Regressor Vector Memory**
- **Read/write control**
- **Data Address**
- **Filter Coefficient Memory**
- **Coefficient Address**
- **Clear**
- **Output** \( y(n) \)
Von Neumann Architecture

- What are the limitations of implementing real-time signal processing using a device that is based on the von Neumann paradigm?

FPGA Density Evolution (1)
FPGA DSP capability increased ~3-4 orders of magnitude in past 18 yrs

- Extraordinary increase in LUT density over time
- Introduction of embedded MPY in 2001
  - opened-up new opportunities for FPGAs
  - in particular accelerated adoption of FPGAs in cellular basestations
FPGA Density Evolution

FPGA DSP capability increased ~3-4 orders of magnitude in past 18 yrs

- Extraordinary increase in LUT density over time
- Introduction of embedded MPY in 2001
  - opened-up new opportunities for FPGAs
  - in particular accelerated adoption of FPGAs in cellular basestations
  - made the programming model more widely acceptable as a designer could work with MPY/MACs instead of distributed arithmetic

Time sharing a multiplier ... is that a good idea?

DSP processor (25 mm$^2$) 12x12 multiplier (.05 mm$^2$)

ENIAC: 20b (28 tube) accumulator

Multiplier Density

<table>
<thead>
<tr>
<th>Year</th>
<th>Virtex-II</th>
<th>Virtex-4</th>
<th>Virtex-5</th>
<th>Virtex-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>2002</td>
<td>2003</td>
<td>2004</td>
<td>2005</td>
</tr>
<tr>
<td>Num. MPYs</td>
<td>5000</td>
<td>10000</td>
<td>15000</td>
<td>20000</td>
</tr>
<tr>
<td>GMACs/Sec</td>
<td>0.00E+00</td>
<td>2.00E+02</td>
<td>4.00E+02</td>
<td>6.00E+02</td>
</tr>
<tr>
<td>#MPY</td>
<td>0.00E+00</td>
<td>2.00E+02</td>
<td>4.00E+02</td>
<td>6.00E+02</td>
</tr>
<tr>
<td>GMACs/Sec</td>
<td>0.00E+00</td>
<td>2.00E+02</td>
<td>4.00E+02</td>
<td>6.00E+02</td>
</tr>
</tbody>
</table>

FIR Filter Realized in FPGA

Spatial Computing

- Parallel programming in space across the fabric of FPGAs
- In contrast to serial programming in time of instruction set processors

Method:
- Start with a parallel representation of the functionality
- Explore architectural parallelism
- Map explicitly to hardware

Conventional DSP Processor - Serial Implementation

Virtex-4 Parallel Implementation Consumes Zero Logic Resources

1 GHz 256 clock cycles = 4 MSPS
8.5 GHz 1 clock cycle = 250 MSPS
Back to the Future

• “...This was a highly parallel machine, before von Neumann spoiled it”

- Tubes: 17,468
- add time: 200 microseconds
- multiply time: 2,800 microseconds
- divide time: 24,000 microseconds
- arithmetic mode: parallel ... later serial

Historical View

- In the past 20 years real-time signal processing has almost become synonymous with the instruction set architecture digital signal processor (DSP)
  - TI
  - ADI
  - Motorola
- While a number (decreasing annually) of industrial/mil/aero/comm. applications that rely on signal processing employ ASICs
Issues with the ISA Approach

- **DSP processors**
  - Architecture definition at fabrication time imposes many restrictions
- The rich body of reduced complexity algorithms is relegated entirely to the realm of theory
- Configurable systems remove this limitation and allow different computation models for using the transistor budget

Observation about instruction set processors (ISP)

- Moore's Law brings more than increases in the number of transistors per chip; it also brings dramatic increases in power consumption and power density. If current trends continue, you would have a device with 425 million transistors in 2005 and a processor with 1.8 billion transistors by 2010, said Pat Gelsinger, Intel's vice president and chief technology officer ...

  ... Even using 0.1-micron technology, Gelsinger envisions a 425-million-transistor die, 40 mm per side, which, clocked at 30 GHz, would dissipate 3,000 to 5,000 watts. In terms of power density, its heat would be close to that of a rocket nozzle, Gelsinger said ... "We can't keep building these things with ever increasing power budgets," he lamented.


- The article is a report on Gelsinger’s presentation at ISSCC 2001.
why fpga’s for dsp?

Programmable Imperative
era of FPGA adoption in embedded applications

- wireless basestations
- medical imaging
- wireline (FEC/other)
- automotive
- military SDR, sigInt, radar

Architecture

Time

Main frame-like, microprogrammed machines

1st gen DSPs: TMS320C10, DSP16
2nd gen DSPs
3rd gen DSPs
4th gen DSPs


Virtex® Product & Process Evolution

Delivering balanced Performance, Power, and Cost
## Virtex-6 FPGA Family

**optimized for diverse set of applications**

<table>
<thead>
<tr>
<th>Virtex-6</th>
<th>LXT</th>
<th>SXT</th>
<th>Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized for:</td>
<td>Logic/Serial</td>
<td>DSP/Serial</td>
<td>High SIO B/W</td>
</tr>
<tr>
<td>Logic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On-chip RAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSP Capabilities</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel I/Os</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial I/Os</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Right mix of features leveraging ASMBL™ architecture
- Flexibility through pin compatibility

---

## Virtex-4 FPGA

- **200,000 Logic Cells**
- **500 MHz BRAM with FIFO & ECC**
- **Greater than 1 Gbps Differential I/O with ChipSync™**
- **AES Design Security**
- **Optimized for Logic**
- **Optimized for DSP**
- **Optimized for Serial I/O and Processing**

- **0.6-11.1 Gbps Transceivers**
- **10/100/1000 Ethernet MAC**
- **500 MHz XtremeDSP™ Slice**
- **680 DMIPS PowerPC™ Processor with APU**
Virtex-4 FPGA MAC++: DSP48

Three Virtex-4 Platforms

<table>
<thead>
<tr>
<th>Device</th>
<th>Logic Cells</th>
<th>Block RAM (Kb)</th>
<th>DSP</th>
<th>SelectID</th>
<th>Slices</th>
<th>Power PC</th>
<th>RocketIO/10G EMAC</th>
<th>BlockIO Terminator</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC4VFX128</td>
<td>1,936</td>
<td>1,024</td>
<td>6</td>
<td>204</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XC4VFX96</td>
<td>1,472</td>
<td>819</td>
<td>6</td>
<td>204</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XC4VFX64</td>
<td>940</td>
<td>512</td>
<td>6</td>
<td>204</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XC4VFX48</td>
<td>588</td>
<td>320</td>
<td>6</td>
<td>204</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XC4VFX36</td>
<td>384</td>
<td>208</td>
<td>6</td>
<td>204</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XC4VFX28</td>
<td>296</td>
<td>160</td>
<td>6</td>
<td>204</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Three Virtex-4 Platforms

why fpga's for dsp? 43

Three Virtex-4 Platforms

why fpga's for dsp? 44
Recent Generation FPGA Families

- As of 2010 Virtex-6 is the most recent FPGA family
- In contrast to Virtex-4 and to a first-order approximation Virtex-6 has
  - Increased lookup table density
    - Different flavor of LUT LUT4 in Virtex-4 compared to LUT6 in Virtex-6
  - Increased number of multipliers
  - Increased memory density
  - Higher-speed IO’s and gigabit rate transceivers (‘RocketIO’)

- The following device family chart puts some numbers on the above capabilities

Virtex-6 Base Platform

<table>
<thead>
<tr>
<th>Part Number</th>
<th>LX110T</th>
<th>LX130T</th>
<th>LX160T</th>
<th>LX190T</th>
<th>LX240T</th>
<th>LX300T</th>
<th>LX360T</th>
<th>SX310T</th>
<th>SX470T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Cells</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>74.5K</td>
<td>129K</td>
<td>200K</td>
<td>241K</td>
<td>364K</td>
<td>590K</td>
<td>759K</td>
<td>315K</td>
<td>478K</td>
</tr>
<tr>
<td>Maximum Onboard RAM (Kbits)</td>
<td>1,045</td>
<td>1,760</td>
<td>3,060</td>
<td>3,800</td>
<td>4,130</td>
<td>6,200</td>
<td>8,280</td>
<td>5,690</td>
<td>7,640</td>
</tr>
<tr>
<td>Block RAM X FIFO (SHR32 Each)</td>
<td>156</td>
<td>264</td>
<td>344</td>
<td>416</td>
<td>416</td>
<td>632</td>
<td>722</td>
<td>704</td>
<td>1,064</td>
</tr>
<tr>
<td>Total Block RAM (Kbits)</td>
<td>5,616</td>
<td>9,504</td>
<td>12,384</td>
<td>14,976</td>
<td>14,976</td>
<td>22,732</td>
<td>25,020</td>
<td>25,344</td>
<td>35,504</td>
</tr>
<tr>
<td>Mixed Mode Clock Manager (MMC)</td>
<td>9</td>
<td>18</td>
<td>18</td>
<td>12</td>
<td>12</td>
<td>18</td>
<td>18</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>DSP Array Cells</td>
<td>288</td>
<td>480</td>
<td>640</td>
<td>768</td>
<td>576</td>
<td>614</td>
<td>864</td>
<td>1,344</td>
<td>2,016</td>
</tr>
<tr>
<td>PCI Express Interface Blocks</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10/100/1000 Ethernet MAC Blocks</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>GTX Low-Power Transceivers</td>
<td>12</td>
<td>20</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Package</th>
<th>Size (Pin)</th>
<th>Maximum Use I/O: Select I/O Interface Pins (GTX Transceivers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF484</td>
<td>23 x 23 mm (1.0 mm)</td>
<td>240 (8) 240 (8)</td>
</tr>
<tr>
<td>FF796</td>
<td>29 x 29 mm (1.0 mm)</td>
<td>400 (12) 400 (12) 400 (12) 400 (12)</td>
</tr>
<tr>
<td>FF1150</td>
<td>35 x 35 mm (1.0 mm)</td>
<td>660 (20) 660 (20) 660 (20) 660 (20)</td>
</tr>
<tr>
<td>FF1704</td>
<td>42 x 42 mm (1.0 mm)</td>
<td>720 (24) 720 (24) 720 (24) 840 (36) 720 (24) 840 (36)</td>
</tr>
<tr>
<td>FF2350</td>
<td>49 x 49 mm (1.0 mm)</td>
<td>1,200 (60) 1,200 (60)</td>
</tr>
</tbody>
</table>
Examples of DSP Frequently Implemented in FPGAs (1)

- Multirate filters
  - Polyphase implementations
  - Multichannel
- FFT
- 3G systems
  - Spectrum access
    - DDC/DUC
    - Polyphase transform
  - RACH processing
  - Searcher
  - Rake receiver
  - Pre-distortion
  - TCC/Viterbi
  - Adaptive antennas
  - Adaptive interference cancellation
  - Crest factor reduction
- 4G
  - OFDM
  - MIMO
- Narrowband (QAM)
  - Matched filter (RRC)
  - Channelization
    - DDC/Polyphase filter bank
  - Adaptive channel equalizer
    - Fractionally spaced FFE
    - Decision feedback
    - Blind CMA
  - Carrier recovery
  - Timing recovery
  - Concatenated FEC
    - Reed-Solomon
    - Convolution codes
- Cable systems
  - High channel density head-end modulator arrays

DSP Implemented in FPGAs (2)

- Video processing and image processing
  - MPEG-2/4 encoding/decoding
  - 2-D filtering
  - Image scaling using multirate filters
    - HDTV up/down converters
    - Projection systems
    - Plasma displays
  - Color space conversion
  - Gamma correction
  - Target recognition
FPGA Custom Computing
one page summary of the time-area tradeoff that fpgas bring to the table

• Typical processor based approach
  • Sequential
    – Possibly low-order parallelism
    – Functional units $M_i$ and $A_i$ time shared across all $N$ filter operations

• Define datapath precision to meet system requirements
  – Use small bit fields when possible, e.g. 1b
  – Use high precision arithmetic when needed, e.g. 40b

• FPGA filter architecture
  – One option
  – Functional units $(M_1, ..., M_N, A_1, ..., A_{N-1})$
    scheduled concurrently

• Data/Compute parallelism

  
  ![FPGA Filter Architecture Diagram]

  
  6b coefficients: 30 dB
  14b coefficients: 70 dB

Conclusion

• Modern FPGAs are capable of meeting the requirements of demanding signal processing systems and are used in
  – Communications
  – Image/video processing
  – High-performance computing
• Place the silicon technology back in the hands of the system designer
• High degree of flexibility
• Inherent parallelism of the device enables extremely high performance … but of course a sequential architecture can be implemented in an FPGA if that is the best option for the design under consideration
• Rich set of resources in the FPGA cover system requirements:
  – arithmetic
  – connectivity
  – embedded software
why fpga’s for dsp?  51

Reference

why fpga’s for dsp?  52

FPGAs versus DSP Processors

- H. Arslan, Cognitive radio, software defined radio and adaptive systems.
  http://books.google.com/books?hl=en&lr=&id=yMgGGe-z7mYC&pg=PA136&lpg=PA136&dq=dsp+processor+performance&source=bl&ots=Tp22YromYc&sig=VOhYQ4LP8XWuftvXlrOhhbl1ixg&hl=en&ei=PqJvStOVCISuswPZi5jOCA&sa=X&oi=book_result&ct=result&resnum=7

### Table 4.1: Comparison of FPGAs and DSPs technologies.

<table>
<thead>
<tr>
<th>Strengths</th>
<th>FPGAs</th>
<th>DSPs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Investigative and flexible processing architecture</td>
<td>High functional capability</td>
</tr>
<tr>
<td></td>
<td>Small silicon footprint</td>
<td>Adaptable to a wide range of applications</td>
</tr>
<tr>
<td></td>
<td>Relatively high bandwidth</td>
<td>Controllable embedded operand size (i.e., mathematical shift)</td>
</tr>
<tr>
<td></td>
<td>High-speed clock</td>
<td>Efficient program loading</td>
</tr>
<tr>
<td></td>
<td>Low power usage</td>
<td>Relatively simple</td>
</tr>
<tr>
<td></td>
<td>Parallel/serial architecture</td>
<td>Relatively short development cycle</td>
</tr>
<tr>
<td></td>
<td>Parallel/serial architecture</td>
<td>Relatively cheap</td>
</tr>
<tr>
<td></td>
<td>Relatively small footprint</td>
<td>Relatively short development cycle</td>
</tr>
<tr>
<td></td>
<td>Can use on-chip memory</td>
<td>Relatively cheap</td>
</tr>
<tr>
<td></td>
<td>Supports interfacing and memory management</td>
<td>Relatively short development cycle</td>
</tr>
<tr>
<td></td>
<td>Many Intellectual Property (IP) cores available</td>
<td>Relatively short development cycle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weaknesses</th>
<th>FPGAs</th>
<th>DSPs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High power consumption</td>
<td>Limited performance due to finite processing power of the device</td>
</tr>
<tr>
<td></td>
<td>Lack of knowledge process</td>
<td>Limited performance due to finite processing power of the device</td>
</tr>
<tr>
<td></td>
<td>Less efficient branching</td>
<td>Limited performance due to finite processing power of the device</td>
</tr>
<tr>
<td></td>
<td>Non-deterministic execution time</td>
<td>Limited performance due to finite processing power of the device</td>
</tr>
<tr>
<td></td>
<td>Irreversible mathematical output</td>
<td>Limited performance due to finite processing power of the device</td>
</tr>
<tr>
<td></td>
<td>Large devices are expensive</td>
<td>Limited performance due to finite processing power of the device</td>
</tr>
<tr>
<td></td>
<td>Relatively long development cycle</td>
<td>Limited performance due to finite processing power of the device</td>
</tr>
<tr>
<td></td>
<td>Relatively costly debugging and maintenance</td>
<td>Limited performance due to finite processing power of the device</td>
</tr>
</tbody>
</table>

why fpga’s for dsp?  52