Processors, FPGAs, and ASICs
Part 2: Processors to Fixed-Function

Stephen A. Edwards

Columbia University

Spring 2020
Spectrum of IC choices

Flexible, efficient
- Full Custom
- ASIC
- Gate Array
- FPGA
- PLD
- GP Processor
- SP Processor
- Multifunction
- Fixed-function

You choose
- Polygons (Intel)
- Circuit (Sony)
- Wires
- Logic network
- Logic function
- Program (e.g., ARM)
- Program (e.g., DSP)
- Settings (e.g., Accelerometer)
- Part number (e.g., 7400)

Cheap, quick to design
Euclid
Euclid’s Algorithm

```c
int gcd(int m, int n)
{
    int r;
    while ((r = m % n) != 0) {
        m = n;
        n = r;
    }
    return n;
}
```
The Intel 80386 c. 1985
### i386 Programmer’s Model

#### Registers

<table>
<thead>
<tr>
<th>31</th>
<th>0</th>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>eax</td>
<td>Mostly</td>
<td>cs</td>
<td>Code segment</td>
</tr>
<tr>
<td>ebx</td>
<td>General-</td>
<td>ds</td>
<td>Data segment</td>
</tr>
<tr>
<td>ecx</td>
<td>Purpose</td>
<td>ss</td>
<td>Stack segment</td>
</tr>
<tr>
<td>edx</td>
<td>Registers</td>
<td>es</td>
<td>Extra segment</td>
</tr>
<tr>
<td>esi</td>
<td>Source index</td>
<td>fs</td>
<td>Data segment</td>
</tr>
<tr>
<td>edi</td>
<td>Destination index</td>
<td>gs</td>
<td>Data segment</td>
</tr>
<tr>
<td>ebp</td>
<td>Base pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>esp</td>
<td>Stack pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eflags</td>
<td>Status word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eip</td>
<td>Instruction Pointer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
int gcd(int m, int n)
{
    int r;
    while ((r = m % n) != 0) {
        m = n;
        n = r;
    }
    return n;
}
Sun’s SPARC Processor c. 1987
### SPARC Programmer’s Model

<table>
<thead>
<tr>
<th>31</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td></td>
</tr>
<tr>
<td>r1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Always 0</td>
</tr>
<tr>
<td></td>
<td>Global Registers</td>
</tr>
<tr>
<td>r7</td>
<td></td>
</tr>
<tr>
<td>r8/o0</td>
<td>Output Registers</td>
</tr>
<tr>
<td></td>
<td>r14/o6</td>
</tr>
<tr>
<td></td>
<td>r15/o7</td>
</tr>
<tr>
<td></td>
<td>Stack Pointer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r16/l0</td>
<td></td>
</tr>
<tr>
<td>r17/l1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Local Registers</td>
</tr>
<tr>
<td></td>
<td>r23/l7</td>
</tr>
<tr>
<td></td>
<td>r24/i0</td>
</tr>
<tr>
<td></td>
<td>Input Registers</td>
</tr>
<tr>
<td></td>
<td>r30/i6</td>
</tr>
<tr>
<td></td>
<td>r31/i7</td>
</tr>
<tr>
<td></td>
<td>Frame Pointer</td>
</tr>
<tr>
<td></td>
<td>Return Address</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PSR</td>
<td>Status Register</td>
</tr>
<tr>
<td>PC</td>
<td>Program Counter</td>
</tr>
<tr>
<td>nPC</td>
<td>Next PC</td>
</tr>
</tbody>
</table>
# SPARC Register Windows

The output registers of the calling procedure become the inputs to the called procedure.

The global registers remain unchanged.

The local registers are not visible across procedures.
Euclid on the SPARC

```c
int gcd(m, n)
int m, n;
{
    int r;
    while ((r = m % n) != 0) {
        m = n;
        n = r;
    }
    return n;
}
```

```assembly
gcd: save %sp,-96,%sp
    mov %i0,%o0
    call .rem,2
    mov %i1,%o1
    mov %o0,%i5
    tst %i5
    be L2
    mov %i1,%o0
L1:  mov %i5,%i1
    call .rem,2
    mov %i1,%o1
    mov %o0,%i5
    tst %i5
    bne,a L1
    mov %i1,%o0
L2:  ret
    restore %g0,%i1,%o0
```
Motorola’s
DSP56000
c. 1986
DSP 56000 Programmer’s Model

Source Registers

x1 | x0
y1 | y0

Accumulator

a2 | a1 | a0
b2 | b1 | b0

Address Registers

r7 | n7 | m7
r4 | n4 | m4
r3 | n3 | m3
r0 | n0 | m0

Program Counter

15 | 0

Status Register

Loop Address

Loop Count

PC Stack

15

SR Stack

15

Stack pointer

0
Motorola DSP56000
Data ALU
FIR Filter in 56000

move  #samples, r0
move  #coeffs, r4
move  #n-1, m0
move  m0, m4
movep y:input, x:(r0)
clr  a x:(r0)+, x0 y:(r4)+, y0
rep  #n-1
mac  x0,y0,a x:(r0)+, x0 y:(r4)+, y0
macr x0,y0,a (r0)-
movep a, y:output
FIR in One ’C6 Assembly Instruction

FIRLOOP:

LDH .D1 *A1++, A2 ; Fetch next sample
||
LDH .D2 *B1++, B2 ; Fetch next coefficient
|| [B0] SUB .L2 B0, 1, B0 ; Decrement loop count
|| [B0] B .S2 FIRLOOP ; Branch if non-zero
|| MPY .M1X A2, B2, A3 ; Sample × Coefficient
|| ADD .L1 A4, A3, A4 ; Accumulate result
FIR in One ’C6 Assembly Instruction

FIRLOOP:

LDH .D1 *A1++, A2 ; Fetch next sample

LDH .D2 *B1++, B2 ; Fetch next coefficient

[B0] SUB .L2 B0, 1, B0 ; Decrement loop count

[B0] B .S2 FIRLOOP ; Branch if non-zero

MPY .M1X A2, B2, A3 ; Sample × Coefficient

ADD .L1 A4, A3, A4 ; Accumulate result

Run in parallel
FIR in One ’C6 Assembly Instruction

Load a halfword (16 bits)

FIRLOOP:

```
LDH .D1 *A1++, A2 ; Fetch next sample
LDH .D2 *B1++, B2 ; Fetch next coefficient
[B0] SUB .L2 B0, 1, B0 ; Decrement loop count
[B0] B .S2 FIRLOOP ; Branch if non-zero
MPY .M1X A2, B2, A3 ; Sample × Coefficient
ADD .L1 A4, A3, A4 ; Accumulate result
```
FIR in One ‘C6 Assembly Instruction

FIRLOOP:

- LDH .D1 *A1++, A2 ; Fetch next sample
- LDH .D2 *B1++, B2 ; Fetch next coefficient
- [B0] SUB .L2 B0, 1, B0 ; Decrement loop count
- [B0] B .S2 FIRLOOP ; Branch if non-zero
- MPY .M1X A2, B2, A3 ; Sample × Coefficient
- ADD .L1 A4, A3, A4 ; Accumulate result

Do this on unit D1

[9x235]FIR in One ‘C6 Assembly Instruction

[55x173]FIRLOOP:

| LDH .D1 *A1++, A2 ; Fetch next sample |
| LDH .D2 *B1++, B2 ; Fetch next coefficient |
| [B0] SUB .L2 B0, 1, B0 ; Decrement loop count |
| [B0] B .S2 FIRLOOP ; Branch if non-zero |
| MPY .M1X A2, B2, A3 ; Sample × Coefficient |
| ADD .L1 A4, A3, A4 ; Accumulate result |

Load a halfword (16 bits)

Do this on unit D1

Use the cross path

Predicated instruction (only if B0 non-zero)

Run in parallel
FIR in One ’C6 Assembly Instruction

FIRLOOP:

LDH .D1  *A1++, A2  ; Fetch next sample
||
LDH .D2  *B1++, B2  ; Fetch next coefficient
|| [B0] SUB .L2  B0, 1, B0  ; Decrement loop count
|| [B0] B .S2  FIRLOOP  ; Branch if non-zero
|| MPY .M1X A2, B2, A3  ; Sample × Coefficient
|| ADD .L1 A4, A3, A4  ; Accumulate result

Use the cross path
FIR in One 'C6 Assembly Instruction

FIRLOOP:

- **LDH .D1** *A1++, A2** ; Fetch next sample
- **LDH .D2** *B1++, B2** ; Fetch next coefficient
- **SUB .L2** B0, 1, B0 ; Decrement loop count
- **[B0] B .S2** FIRLOOP ; Branch if non-zero
- **MPY .M1X** A2, B2, A3 ; Sample × Coefficient
- **ADD .L1** A4, A3, A4 ; Accumulate result

Predicated instruction (only if B0 non-zero)
Analog Devices ADXL345 Accelerometer

3-AXIS SENSOR SENSE ELECTRONICS DIGITAL FILTER ADXL345 POWER MANAGEMENT CONTROL AND INTERRUPT LOGIC SERIAL I/O INT1 VS VDD I/O INT2 SDA/SDI/SDIO SDO/ALT ADDRESS SCL/SCLK CS GND ADC 32 LEVEL FIFO
14 pins, 3mm by 5mm
DE1-SoC Connections to the ADXL345 Accelerometer
I²C Bus Protocol
## ADXL345 Registers

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DEVID</td>
<td>R</td>
<td>11100101</td>
<td>Device ID</td>
</tr>
<tr>
<td>0x01 to 0x1C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td>Reserved; do not access</td>
</tr>
<tr>
<td>0x1D</td>
<td>THRESH_TAP</td>
<td>R/W</td>
<td>00000000</td>
<td>Tap threshold</td>
</tr>
<tr>
<td>0x1E</td>
<td>OFSX</td>
<td>R/W</td>
<td>00000000</td>
<td>X-axis offset</td>
</tr>
<tr>
<td>0x1F</td>
<td>OFSY</td>
<td>R/W</td>
<td>00000000</td>
<td>Y-axis offset</td>
</tr>
<tr>
<td>0x20</td>
<td>OFSZ</td>
<td>R/W</td>
<td>00000000</td>
<td>Z-axis offset</td>
</tr>
<tr>
<td>0x21</td>
<td>DUR</td>
<td>R/W</td>
<td>00000000</td>
<td>Tap duration</td>
</tr>
<tr>
<td>0x22</td>
<td>Latent</td>
<td>R/W</td>
<td>00000000</td>
<td>Tap latency</td>
</tr>
<tr>
<td>0x23</td>
<td>Window</td>
<td>R/W</td>
<td>00000000</td>
<td>Tap window</td>
</tr>
<tr>
<td>0x24</td>
<td>THRESH_ACT</td>
<td>R/W</td>
<td>00000000</td>
<td>Activity threshold</td>
</tr>
<tr>
<td>0x25</td>
<td>THRESH_INACT</td>
<td>R/W</td>
<td>00000000</td>
<td>Inactivity threshold</td>
</tr>
<tr>
<td>0x26</td>
<td>TIME_INACT</td>
<td>R/W</td>
<td>00000000</td>
<td>Inactivity time</td>
</tr>
<tr>
<td>0x27</td>
<td>ACT_INACT_CTL</td>
<td>R/W</td>
<td>00000000</td>
<td>Axis enable control for activity and inactivity detection</td>
</tr>
<tr>
<td>0x28</td>
<td>THRESH_FF</td>
<td>R/W</td>
<td>00000000</td>
<td>Free-fall threshold</td>
</tr>
<tr>
<td>0x29</td>
<td>TIME_FF</td>
<td>R/W</td>
<td>00000000</td>
<td>Free-fall time</td>
</tr>
<tr>
<td>0x2A</td>
<td>TAP_AXES</td>
<td>R/W</td>
<td>00000000</td>
<td>Axis control for single tap(double tap)</td>
</tr>
<tr>
<td>0x2B</td>
<td>ACT_TAP_STATUS</td>
<td>R</td>
<td>00000000</td>
<td>Source of single tap(double tap)</td>
</tr>
<tr>
<td>0x2C</td>
<td>BW_RATE</td>
<td>R/W</td>
<td>00001010</td>
<td>Data rate and power mode control</td>
</tr>
<tr>
<td>0x2D</td>
<td>POWER_CTL</td>
<td>R/W</td>
<td>00000000</td>
<td>Power-saving features control</td>
</tr>
<tr>
<td>0x2E</td>
<td>INT_ENABLE</td>
<td>R/W</td>
<td>00000000</td>
<td>Interrupt enable control</td>
</tr>
<tr>
<td>0x2F</td>
<td>INT_MAP</td>
<td>R/W</td>
<td>00000000</td>
<td>Interrupt mapping control</td>
</tr>
<tr>
<td>0x30</td>
<td>INT_SOURCE</td>
<td>R</td>
<td>00000000</td>
<td>Source of interrupts</td>
</tr>
<tr>
<td>0x31</td>
<td>DATA_FORMAT</td>
<td>R/W</td>
<td>00000000</td>
<td>Data format control</td>
</tr>
<tr>
<td>0x32</td>
<td>DATA0</td>
<td>R</td>
<td>00000000</td>
<td>X-Axis Data 0</td>
</tr>
<tr>
<td>0x33</td>
<td>DATA1</td>
<td>R</td>
<td>00000000</td>
<td>X-Axis Data 1</td>
</tr>
<tr>
<td>0x34</td>
<td>DATAY0</td>
<td>R</td>
<td>00000000</td>
<td>Y-Axis Data 0</td>
</tr>
<tr>
<td>0x35</td>
<td>DATAY1</td>
<td>R</td>
<td>00000000</td>
<td>Y-Axis Data 1</td>
</tr>
<tr>
<td>0x36</td>
<td>DATAZ0</td>
<td>R</td>
<td>00000000</td>
<td>Z-Axis Data 0</td>
</tr>
<tr>
<td>0x37</td>
<td>DATAZ1</td>
<td>R</td>
<td>00000000</td>
<td>Z-Axis Data 1</td>
</tr>
<tr>
<td>0x38</td>
<td>FIFO_CTL</td>
<td>R/W</td>
<td>00000000</td>
<td>FIFO control</td>
</tr>
<tr>
<td>0x39</td>
<td>FIFO_STATUS</td>
<td>R</td>
<td>00000000</td>
<td>FIFO status</td>
</tr>
</tbody>
</table>
ACT, an Enable Bits and INACT, an Enable Bits

A setting of 1 enables a bit. A setting of 0 disables the associated function. For activity detection, all enabling portions are logically ORed, causing the activity functions to trigger whenever any of the enabling portions are enabled for the specified period.

Register 0x2A—THRESH_ACT (Read/Write)

The THRESH_ACT register is eight bits and holds the threshold magnitude of the activity event. A setting of 1 corresponds to involvement in the event, whereas a setting of 0 prevents the corresponding bits from generating interrupts. The THRESH_ACT register is updated whenever a change is detected.

Register 0x2B—THRESH_INACT (Read/Write)

The THRESH_INACT register is eight bits and holds the threshold magnitude for detecting inactivity. The activity data is negated, so the magnitude of the inactivity event is compared with the value in the THRESH_INACT register. The scale factor is 62.5 m/s^2. A setting of 1 enables the associated bit for activity detection.

Register 0x32—THRESH_TAP (Read/Write)

The THRESH_TAP register is eight bits and holds the threshold magnitude of the tap event. A setting of 1 corresponds to involvement in the event, whereas a setting of 0 prevents the corresponding bit from triggering a tap event.

Suppress Bit

Register 0x2C—ACT_INACT_CTL (Read/Write)

A setting of 0 disables the associated function. In ac-coupled operation for activity detection, a setting of 1 enables the associated function.

ACT AC/DC and INACT AC/DC Bits

A setting of 0 selects dc-coupled operation, and a setting of 1 selects ac-coupled operation. In dc-coupled operation, the current acceleration magnitude is compared directly with the THRESH_AC/DC and THRESH_INACT_AC/DC to determine whether activity or inactivity is detected. In ac-coupled operation for activity detection, the acceleration value at the start of activity detection is taken as a reference value. New acceleration values are compared to this reference value, and if the magnitude of the difference exceeds the THRESH_AC/DC value, activity is indicated. Similarly, in ac-coupled operation for inactivity detection, a reference value is used for comparison and is updated whenever the scale factor is changed. In dc-coupled operation, the current acceleration magnitude is compared directly with the THRESH_INACT_AC/DC. If the difference is less than the value in the THRESH_INACT_AC/DC for the time-inactivity test, the device is considered inactive and the inactivity interrupt is triggered.

Adapt Bit

Register 0x2D—ACT_TAP_STATUS (Read/Only)

A setting of 0 in the adapt bit indicates that the part is asleep, and a setting of 1 indicates that the part is not asleep. This bit toggles only if the device is configured for auto sleep. See the AUTO_SLEEP Bit section for more information on auto-sleep mode.

LOW POWER Bit

Register 0x2E—INT_ENABLE (Read/Write)

A setting of 0 in the LOW POWER bit regulates selected interrupts, and a setting of 1 enables all selected interrupts. A value of 0 disables double tap function.

Table 20: Timing for the communication protocol

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART</td>
<td>USART</td>
<td>1 bit</td>
</tr>
<tr>
<td>SPI master</td>
<td>SPI slave</td>
<td>0.5 bit</td>
</tr>
<tr>
<td>SPI slave</td>
<td>SPI master</td>
<td>0.5 bit</td>
</tr>
</tbody>
</table>

Link Bit

Register 0x2F—FREE_FALL (Read/Write)

A setting of 1 in the link bit is required by the activity and inactivity functions before the start of the activity function and inactivity functions. Before activity detection, the link bit is required by the activity function. The link bit is cleared by the inactivity function. After activity detection, the link bit is required by the activity function and inactivity functions. The DATA_READY, ACT_TAP, and INACT_TAP functions are enabled by the corresponding interrupt enable bits. The link bit is set by the corresponding interrupt enable bit

Adapt Bit

Register 0x30—OFSX, Register 0x31—OFSY, Register 0x32—OFSZ—Offset Registers

These data registers are 16-bit, 2’s complement registers storing the bias values for each axis and are used to set the origins of the output data axes. The 16-bit values can be selected that are appropriate for the communication protocol and desired operating mode.

Threshold Bit

Register 0x32—THRESH_TAP (Read/Write)

A setting of 1 in the threshold bit indicates that the tap event corresponds to the time value representing the minimum time that the value of all axes must be above the THRESH_TAP threshold to qualify as a tap event. The scale factor is 62.5 m/s^2. A setting of 0 disables the double tap function.

Adapt Bit

Register 0x32—OFSX, Register 0x31—OFSY, Register 0x32—OFSZ—Offset Registers

The data registers are 16-bit, 2’s complement registers storing the bias values for each axis and are used to set the origins of the output data axes. The 16-bit values can be selected that are appropriate for the communication protocol and desired operating mode.

Setting bits in this register to a value of 0 enables interrupts, whereas a value of 1 prevents the corresponding bit from triggering an interrupt.

Adapt Bit

Register 0x33—INT_ENABLE (Read/Write)

A setting of 0 in the FREE_FALL bit is required by the activity and inactivity functions. After activity and inactivity functions, the link bit is required by the activity and inactivity functions. Again, the DATA_READY, ACT_TAP, and INACT_TAP functions are enabled by the corresponding interrupt enable bits. The link bit is set by the corresponding interrupt enable bit.
Analog Devices ADV7180 Video Decoder

- **ANALOG VIDEO INPUTS**
  - A_IN1, A_IN2, A_IN3, A_IN4, A_IN5, A_IN6
- **XTAL1, XTAL**
- **CLOCK PROCESSING BLOCK**
  - PLL
  - ADLLT PROCESSING
- **10-BIT, 86MHz ADC**
- **DIGITAL PROCESSING BLOCK**
  - 2D COMB
  - VBI SLICER
  - COLOR DEMOD
- **MUX BLOCK**
  - AA FILTER
- **REFERENCE**
- **I^2C/CONTROL**
  - P7 TO P0
- **OUTPUT BLOCK**
  - LLC
  - 8-BIT/16-BIT PIXEL DATA
  - VS
  - HS
  - FIELD^2
  - GPO^1
  - SFL
  - INTRQ
- **FIFO**
- **INPUT BLOCK**
  - SCLK, SDATA, ALSB, RESET, PWRDWN
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address</th>
<th>RW</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
</table>
Fixed-function: The 7400 series

7400
Quad NAND Gate

74374
Octal D Flip-Flop
The 74181 4-bit ALU
The 74181 4-bit ALU