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
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>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eax</td>
<td>Mostly General-Purpose Registers</td>
</tr>
<tr>
<td>ebx</td>
<td>Source index</td>
</tr>
<tr>
<td>ecx</td>
<td>Destination index</td>
</tr>
<tr>
<td>edx</td>
<td>Base pointer</td>
</tr>
<tr>
<td>esi</td>
<td>Stack pointer</td>
</tr>
<tr>
<td>edi</td>
<td></td>
</tr>
<tr>
<td>ebp</td>
<td></td>
</tr>
<tr>
<td>esp</td>
<td></td>
</tr>
<tr>
<td>eflags</td>
<td>Status word</td>
</tr>
<tr>
<td>eip</td>
<td>Instruction Pointer</td>
</tr>
</tbody>
</table>

#### Segments

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>Code segment</td>
</tr>
<tr>
<td>ds</td>
<td>Data segment</td>
</tr>
<tr>
<td>ss</td>
<td>Stack segment</td>
</tr>
<tr>
<td>es</td>
<td>Extra segment</td>
</tr>
<tr>
<td>fs</td>
<td>Data segment</td>
</tr>
<tr>
<td>gs</td>
<td>Data segment</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>Register</th>
<th>Description</th>
<th>31</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td>Always 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1</td>
<td>Global Registers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r7</td>
<td>Output Registers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r8/o0</td>
<td>Stack Pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r14/o6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r15/o7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r16/l0</td>
<td>Local Registers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r17/l1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r23/l7</td>
<td>Input Registers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r24/i0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r30/i6</td>
<td>Frame Pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r31/i7</td>
<td>Return Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSR</td>
<td>Status Register</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>Program Counter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nPC</td>
<td>Next PC</td>
<td></td>
<td></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.
```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

<table>
<thead>
<tr>
<th>55 48 47</th>
<th>24 23</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>x0</td>
<td></td>
</tr>
<tr>
<td>y1</td>
<td>y0</td>
<td></td>
</tr>
</tbody>
</table>

Accumulator

<table>
<thead>
<tr>
<th>a2</th>
<th>a1</th>
<th>a0</th>
</tr>
</thead>
<tbody>
<tr>
<td>b2</td>
<td>b1</td>
<td>b0</td>
</tr>
</tbody>
</table>

15 0

Program Counter

Status Register

Loop Address

Loop Count

PC Stack

SR Stack

Stack pointer
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

Do this on unit D1

FIRLOOP:

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

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

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

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

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

MPY .M1X A2, B2, A3 ; Sample \times 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
```

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

| [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

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

- 3-AXIS SENSOR
- SENSE ELECTRONICS
- DIGITAL FILTER
- ADC
- POWER MANAGEMENT
- CONTROL AND INTERRUPT LOGIC
- SERIAL I/O
- INT1
- INT2
- SDA/SDI/SDIO
- SDO/ALT ADDRESS
- SCL/SCLK
- VS
- VDD I/O
- GND
- CS
- 32 LEVEL FIFO

ADXL345
14 pins, 3mm by 5mm
DE1-SoC Connections to the ADXL345 Accelerometer
I²C Bus Protocol
<table>
<thead>
<tr>
<th>Address</th>
<th>Hex</th>
<th>Dec</th>
<th>Name</th>
<th>Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0</td>
<td>0</td>
<td>DEVID</td>
<td>R</td>
<td>11100101</td>
<td>Device ID</td>
</tr>
<tr>
<td>0x01 to 0x1C</td>
<td>1 to 28</td>
<td>1 to 28</td>
<td>Reserved</td>
<td></td>
<td></td>
<td>Reserved; do not access</td>
</tr>
<tr>
<td>0x1D</td>
<td>29</td>
<td>29</td>
<td>THRESH_TAP</td>
<td>R/W</td>
<td>00000000</td>
<td>Tap threshold</td>
</tr>
<tr>
<td>0x1E</td>
<td>30</td>
<td>30</td>
<td>OFSX</td>
<td>R/W</td>
<td>00000000</td>
<td>X-axis offset</td>
</tr>
<tr>
<td>0x1F</td>
<td>31</td>
<td>31</td>
<td>OFSY</td>
<td>R/W</td>
<td>00000000</td>
<td>Y-axis offset</td>
</tr>
<tr>
<td>0x20</td>
<td>32</td>
<td>32</td>
<td>OFSZ</td>
<td>R/W</td>
<td>00000000</td>
<td>Z-axis offset</td>
</tr>
<tr>
<td>0x21</td>
<td>33</td>
<td>33</td>
<td>DUR</td>
<td>R/W</td>
<td>00000000</td>
<td>Tap duration</td>
</tr>
<tr>
<td>0x22</td>
<td>34</td>
<td>34</td>
<td>Latent</td>
<td>R/W</td>
<td>00000000</td>
<td>Tap latency</td>
</tr>
<tr>
<td>0x23</td>
<td>35</td>
<td>35</td>
<td>Window</td>
<td>R/W</td>
<td>00000000</td>
<td>Tap window</td>
</tr>
<tr>
<td>0x24</td>
<td>36</td>
<td>36</td>
<td>THRESH_ACT</td>
<td>R/W</td>
<td>00000000</td>
<td>Activity threshold</td>
</tr>
<tr>
<td>0x25</td>
<td>37</td>
<td>37</td>
<td>THRESH_INACT</td>
<td>R/W</td>
<td>00000000</td>
<td>Inactivity threshold</td>
</tr>
<tr>
<td>0x26</td>
<td>38</td>
<td>38</td>
<td>TIME_INACT</td>
<td>R/W</td>
<td>00000000</td>
<td>Inactivity time</td>
</tr>
<tr>
<td>0x27</td>
<td>39</td>
<td>39</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>40</td>
<td>40</td>
<td>THRESH_FF</td>
<td>R/W</td>
<td>00000000</td>
<td>Free-fall threshold</td>
</tr>
<tr>
<td>0x29</td>
<td>41</td>
<td>41</td>
<td>TIME_FF</td>
<td>R/W</td>
<td>00000000</td>
<td>Free-fall time</td>
</tr>
<tr>
<td>0x2A</td>
<td>42</td>
<td>42</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>43</td>
<td>43</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>44</td>
<td>44</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>45</td>
<td>45</td>
<td>POWER_CTL</td>
<td>R/W</td>
<td>00000000</td>
<td>Power-saving features control</td>
</tr>
<tr>
<td>0x2E</td>
<td>46</td>
<td>46</td>
<td>INT_ENABLE</td>
<td>R/W</td>
<td>00000000</td>
<td>Interrupt enable control</td>
</tr>
<tr>
<td>0x2F</td>
<td>47</td>
<td>47</td>
<td>INT_MAP</td>
<td>R/W</td>
<td>00000000</td>
<td>Interrupt mapping control</td>
</tr>
<tr>
<td>0x30</td>
<td>48</td>
<td>48</td>
<td>INT_SOURCE</td>
<td>R</td>
<td>00000010</td>
<td>Source of interrupts</td>
</tr>
<tr>
<td>0x31</td>
<td>49</td>
<td>49</td>
<td>DATA_FORMAT</td>
<td>R/W</td>
<td>00000000</td>
<td>Data format control</td>
</tr>
<tr>
<td>0x32</td>
<td>50</td>
<td>50</td>
<td>DATA0</td>
<td>R</td>
<td>00000000</td>
<td>X-Axis Data 0</td>
</tr>
<tr>
<td>0x33</td>
<td>51</td>
<td>51</td>
<td>DATA1</td>
<td>R</td>
<td>00000000</td>
<td>X-Axis Data 1</td>
</tr>
<tr>
<td>0x34</td>
<td>52</td>
<td>52</td>
<td>DATAY0</td>
<td>R</td>
<td>00000000</td>
<td>Y-Axis Data 0</td>
</tr>
<tr>
<td>0x35</td>
<td>53</td>
<td>53</td>
<td>DATAY1</td>
<td>R</td>
<td>00000000</td>
<td>Y-Axis Data 1</td>
</tr>
<tr>
<td>0x36</td>
<td>54</td>
<td>54</td>
<td>DATAZ0</td>
<td>R</td>
<td>00000000</td>
<td>Z-Axis Data 0</td>
</tr>
<tr>
<td>0x37</td>
<td>55</td>
<td>55</td>
<td>DATAZ1</td>
<td>R</td>
<td>00000000</td>
<td>Z-Axis Data 1</td>
</tr>
<tr>
<td>0x38</td>
<td>56</td>
<td>56</td>
<td>FIFO_CTL</td>
<td>R/W</td>
<td>00000000</td>
<td>FIFO control</td>
</tr>
<tr>
<td>0x39</td>
<td>57</td>
<td>57</td>
<td>FIFO_STATUS</td>
<td>R</td>
<td>00000000</td>
<td>FIFO status</td>
</tr>
</tbody>
</table>
REGISTER DEFINITIONS

Register 0x02—THRESH_ACT (Read/Write)

The THRESH_ACT register is eight bits and contains the threshold value for detecting activity. The data format is unsigned, so the magnitude of the activity event compared with the value in the THRESH_ACT register. The scale factor is 62.5 m/s²/LSB. A value of 0 results in undesirable behavior if single tap/double tap interrupts are enabled.

Register 0x03—THRESH_INACT (Read/Write)

The THRESH_INACT register is eight bits and sets the threshold value for detecting inactivity. The data format is signed, so the magnitude of the inactivity event compared with the value in the THRESH_INACT register. The scale factor is 62.5 m/s²/LSB. A value of 0 may result in undesirable behavior if single tap/double tap interrupts are enabled.

Register 0x22—OFSX, OFSY, OFSZ (Read/Write)

The OFSX, OFSY, and OFSZ registers are each eight bits and offset adjustments in the rate bits. Each register contains a scale factor of 0.5 µg/LSB (that is, 0x2 = 2). The values stored in the offset registers is automatically added to the acceleration data, and the resulting value is stored in the output registers. For additional information regarding offset calibration and the use of the offset registers, refer to the Offset Calibration section.

Register 0x23—DUR (Read/Write)

The DUR register is eight bits and contains an unsigned time value representing the maximum time that an event must be above the THRESH_TAP threshold to qualify as a tap event. The scale factor is 25 µs/LSB. A value of 0 allows the single tap/ double tap function.

Register 0x24—THRESH_FF (Read/Write)

The THRESH_FF register is eight bits and sets the threshold value for detecting free-fall. The data format is signed, so the magnitude of the free-fall event compared with the value in the THRESH_FF register is set to a value of 0 as the time constant of the output data rate. A value of 0 results in an interrupt when the output data is less than the value in the THRESH_FF register.

ACT and INACT ACT/DC and INACT ACT/DC Bits

A setting of 0 selects dc-coupled operation, and a setting of 1 enables ac-coupled operation. In dc-coupled operation, the current acceleration magnitude is compared directly with THRESH_ACT and THRESH_INACT to determine whether activity or inactivity is detected.

Ac-coupled operation for activity detection, the acceleration value at the start of activity detection is taken as a reference value. New samples of acceleration are then compared to this reference value, and if the magnitude of the difference exceeds the THRESH_INACT value, the activity function is triggered. Similarly, in ac-coupled operation for inactivity detection, a reference value is used for comparison and is updated whenever the next magnitude exceeds the current reference. The reference value is selected, the device compares the magnitude of the difference between the reference value and the current acceleration with THRESH_INACT. If the difference is less than the value in THRESH_INACT, the device sets the inactivity function.

Register 0x25—THRESH_TAP (Read/Write)

The THRESH_TAP register is eight bits and contains the threshold value for detecting taps. The data format is unsigned, so the magnitude of the tap event compared with the value in the THRESH_TAP register. The scale factor is 6.25 g/LSB. A value of 0 results in undesirable behavior if simple tap/double tap interrupts are enabled.

Adapt Bit

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

Adapt Mode

If the list box is not set, the AUTO_SLEEP feature is enabled and setting the AUTO_SLEEP bit does not have an effect on device operation. Refer to the Link Bit section on the Link Mode section for more information on enabling the link function. When changing the AUTO_SLEEP bit, it is recommended that the part be placed into standby mode and then back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if the deep sleep mode is manually disabled. Otherwise, the first few samples of data after the deep sleep bit is cleared may have additional noise, even if the device was asleep when the bit was cleared.

Table 23: Running on Sleep Mode

The DATA_FORMAT register contains the presentation data rate to Register 0x32 through Register 0x37. All data except for the first 1.0 g range, must be clipped to avoid resampling.

Table 24: Setting of Readings in Sleep Mode

The DATA_FORMAT register contains the presentation data rate to Register 0x32 through Register 0x37. All data except for the first 1.0 g range, must be clipped to avoid resampling.
<table>
<thead>
<tr>
<th>Address</th>
<th>Register Name</th>
<th>RW</th>
<th>Value</th>
<th>Hex</th>
<th>RW</th>
<th>Value</th>
<th>Hex</th>
</tr>
</thead>
</table>

**Notes:**
- **Address:** Hexadecimal address of the register.
- **Register Name:** Name of the register.
- **RW:** Read/Write permission.
- **Value:** Hexadecimal value of the register.
- **Hex:** Hexadecimal representation of the value.
Fixed-function: The 7400 series

7400
Quad NAND Gate

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