Spectrum of IC choices

<table>
<thead>
<tr>
<th>Flexibility</th>
<th>You choose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Custom</td>
<td>polygons (Intel)</td>
</tr>
<tr>
<td>ASIC</td>
<td>circuit (Sony)</td>
</tr>
<tr>
<td>Gate Array</td>
<td>wires</td>
</tr>
<tr>
<td>FPGA</td>
<td>logic network</td>
</tr>
<tr>
<td>PLD</td>
<td>logic function</td>
</tr>
<tr>
<td>GP Processor</td>
<td>program (e.g., Pentium)</td>
</tr>
<tr>
<td>SP Processor</td>
<td>program (e.g., DSP)</td>
</tr>
<tr>
<td>Multifunction</td>
<td>settings (e.g., Ethernet)</td>
</tr>
<tr>
<td>Fixed-function</td>
<td>part number (e.g., 74LS00)</td>
</tr>
</tbody>
</table>

NMOS Transistor Cross Section

NMOS Transistor Cross Section

Inverter Transistors and Layout

NAND Gate Transistors and Layout

Full-custom ICs

Standard Cell ASICs

Channeled Gate Arrays
Architectural Description

Values stored in static memory cells control all the configurable logic elements and interconnect resources. These values load into the memory cells on power-up, and can reload if necessary to change the function of the device.

The Spartan-IIE user-programmable gate array, shown in Figure 1, is composed of five major configurable elements:

- Versatile multi-level interconnect structure
- Input/Output Block
- CLBs
- Source index
- Data segment
- Product Specification
- Stack pointer
- Extra segment
- Destination index
- CLBs
- Data segment
- Base pointer
- Stack pointer
- Status word
- Instruction Pointer

As can be seen in Figure 1, the CLBs form the central logic structure with easy access to all support and routing structures.

In Figure 4, the CLB is shown as a basic slice with two Look-Up Tables (LUTs) and registration stages. Each slice contains a set of carry and control logic, a function generator, and a multiplexer.

Example: Euclid’s Algorithm

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

Example: Euclid’s Algorithm

```c
int gcd(int m, int n)
{
    int r;
    while ((r = m % n) != 0) {
        m = n;
        n = r;
    }
    return n;
}
```
### Euclid on the SPARC

**gcd:**
- `save %sp, -112, %sp`
- `mov %10, %10`
- `b .LL3`
- `mov %11, %10`
- `mov %11, %10`
- `.LL2:`
  - `mov %0, %10`
  - `call .rem, 0`
  - `mov %10, %10`
  - `cmp %0, 0`
  - `bne .LL5`
  - `mov %10, %10`
  - `ret`
  - `restore`

### DSP56000 Programmer's Model

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.

### Motorola DSP56301

The only difference between them is that the carry propagates in opposite directions.

Test shows a block diagram of the AGU.

Motorola DSP56301 Block Diagram

**Figure 1-1.**

### Motorola DSP56301 ALU

Motorola DSP56301 AGU

### TI TMS320C6000 VLIW DSP

### FIR Filter in 56000

- `move #samples, r0`
- `move #coeffs, r4`
- `move #n-1, m0`
- `move m0, m4`
- `move p: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**

- **FIR**:
  - **LDH .D1 *A1++, A2**: Fetch next sample
  - **LDH .D2 *B1++, B2**: Fetch next coeff.
  - **[B0] SUB .L2 B0, 1, B0**: Decrement count
  - **[B0] B .S2 FIRLOOP**: Branch if non-zero
  - **ADD .L1 A4, A3, A4**: Accumulate result

  Use the cross path

  Predicated instruction (only if B0 non-zero)

  Run these instructions in parallel

---

**AX88796 Ethernet Controller**

**5.0 Registers Operation**

**5.1 MAC Core Registers**

- All registers of MAC Core are 8-bit wide and mapped into pages which are selected by PS (Page Select) in
- **01H Page Start Register**
- **02H Page Stop Register**
- **03H Boundary Pointer**
- **04H Transmit Status Register**
- **05H Number of Collisions Register**
- **06H Receive Status Register**
- **07H Interrupt Status Register**
- **08H Current Remote DMA Address 0**
- **09H Current Remote DMA Address 1**
- **0AH Reserved Remote Byte Count 0**
- **0BH Reserved Remote Byte Count 1**
- **0CH Command Register**
- **0DH Frame Alignment Errors**
- **10H, 11H Data Port Data Port**
- **12H, 13H Transmit Page Start Address**
- **14H, 15H Transmit Page Stop Address**
- **16H, 17H Transmit Remote DMA Address**
- **18H, 19H Transmit Byte Count 0**
- **1AH, 1BH Transmit Byte Count 1**
- **1CH, 1DH Transmit Page Start Address**
- **1EH, 1FH Transmit Page Stop Address**
- **20H, 21H Transmit Remote DMA Address**
- **22H, 23H Transmit Byte Count 0**
- **24H, 25H Transmit Page Start Address**
- **26H, 27H Transmit Page Stop Address**
- **28H, 29H Transmit Remote DMA Address**
- **2AH, 2BH Transmit Byte Count 0**
- **2CH, 2DH Transmit Page Start Address**
- **2EH, 2FH Transmit Page Stop Address**
- **30H, 31H Transmit Remote DMA Address**
- **32H, 33H Transmit Byte Count 0**
- **34H, 35H Transmit Page Start Address**
- **36H, 37H Transmit Page Stop Address**
- **38H, 39H Transmit Remote DMA Address**
- **3AH, 3BH Transmit Byte Count 0**
- **3CH, 3DH Transmit Page Start Address**
- **3EH, 3FH Transmit Page Stop Address**
- **40H, 41H Transmit Remote DMA Address**
- **42H, 43H Transmit Byte Count 0**
- **44H, 45H Transmit Page Start Address**
- **46H, 47H Transmit Page Stop Address**
- **48H, 49H Transmit Remote DMA Address**
- **4AH, 4BH Transmit Byte Count 0**
- **4CH, 4DH Transmit Page Start Address**
- **4EH, 4FH Transmit Page Stop Address**
- **50H, 51H Transmit Remote DMA Address**
- **52H, 53H Transmit Byte Count 0**
- **54H, 55H Transmit Page Start Address**
- **56H, 57H Transmit Page Stop Address**
- **58H, 59H Transmit Remote DMA Address**
- **5AH, 5BH Transmit Byte Count 0**
- **5CH, 5DH Transmit Page Start Address**
- **5EH, 5FH Transmit Page Stop Address**

---

**SAA7114H Registers, page 1 of 7 (!)**

**Fixed-function: The 7400 series**

- **7400 Quad NAND Gate**
- **74374 Octal D Flip-Flop**