September 1987

Microarchitecture specification for the Analog Processor Node, APN

John Mates

Steve Owens

Follow this and additional works at: http://digitalcommons.ohsu.edu/csetech

Recommended Citation
http://digitalcommons.ohsu.edu/csetech/169

This Article is brought to you for free and open access by OHSU Digital Commons. It has been accepted for inclusion in CSETech by an authorized administrator of OHSU Digital Commons. For more information, please contact champieu@ohsu.edu.
Microarchitecture Specification for
the Analog Processor Node, APN

John Mates and Steve Owens

Oregon Graduate Center
Department of Computer Science
and Engineering
19600 N.W. von Neumann Drive
Beaverton, OR 97006-1999 USA

Technical Report No. CS/E 88-014

Abstract

This technical report contains the Microarchitecture Specification for the Analog Processor Node or APN, which was designed in the 1987 OGC Advanced VLSI class, and fabricated by MOSIS. In addition, the results of chip testing are included. The APN takes digital input, converts that input and internally stored digital weights into analog signals, performs a sum of products operation in the analog domain and then converts the result back to digital for external transmission. The entire process takes one clock. There are four such processors on one APN chip.
1. Introduction

This document specifies the microarchitecture of an analog/digital integrated circuit. The design and testing of this circuit and companion circuits is both a class exercise and a first attempt to create VLSI circuitry which can efficiently emulate a connection net, the major component of neurocomputers. The APN chip is part of the OGC Cognitive Architecture project.

Connection net nodes compute a simple function of their inputs. Our circuit is intended to multiplex the computation of this function over many abstract connection net nodes in a single processor node (PN). The circuit will conform to 3u MOSIS scaleable rules for CMOS. We call it an "analog processor node" (APN) to distinguish it from all-digital processor nodes (DPNs). The APN chip contains four analog processor nodes, henceforth referred to as APN(i) where i is an integer, [1 - 4].

Our design uses four bit buses throughout. There are two buses for addressing the internal registers and the APNs, respectively. There is a bus for I/O to and from the internal registers, a bus to carry each APN's computed output value, and a bus to place each APN address on the output pins. We expect our bus design (and associated control logic) to minimize the "glue" required to interface multiple PNs with the other chips in a connection network.

Our design adheres to several conventions: (1) external signal names start with lowercase "x"; (2) all external signals are used and are valid during during phase 1 of the non overlapping two phase clock used for chip timing implying the naming convention that external signal names terminate with the suffix ".-1"; and (3), every chip input signal is phase one trapped by the internal module which handles that signal.

The following text specifies the architectures of each of the five modules from which we have built our APN chip, namely, the external pins module (XPINSMOD), and four each of register file modules (REGMOD), the APN processor module (APMOD), the APN output module (OUTMOD), and the bus interface module (BiMOD).

1.1. XPINSMOD: the module which controls the external pins.

The XPINSMOD module organizes the external interface of our APN chip. It interfaces all external signals on the pins of the chip to the internal chip signals used by the other modules. XPINSMOD delivers incoming signals directly to the relevant APN modules along wires. XPINSMOD also contains tristate drivers which put outgoing chip signals onto the external pins.

The outgoing signals of the bidirectional bus, xinout_B, are gated by external signal xrd_1 (buffered by input pad circuitry). The four APNs share this bus.

The outgoing bus signals xvout_B.i and xaout_B.i (representing APN(i)'s computed output value and "address" or ID) are gated by the local outenab.i_1 signal, generated in the BiMOD module of APN(i).

Other outgoing signals, placed on the external pins by XPINSMOD, include signals used for resolving bus contention, xreqout.i_1 and xbusy.i_1. When data are placed on the output bus, XPINSMOD asserts xwstrb.i_1. All these signals are sourced by tristate drivers which reside in XPINSMOD.
XPINSMOD uses thirty seven pins to interface each APN on the chip to the external environment. Some pins are common to all four APNs; others are not common.

(1) Four pins to address one of nine internal registers (xraddr_B).
(2) Four pins to address (select) the APN chip (xcaddr_i_B).
(3) Four bidirectional data in/out pins (xinout_B).
(4) Four value output pins (xvout_i_B).
(5) Four chip address ("ID") output pins (xaout_i_B).
(6) Two pins (xreqout_i_1 and xreqin_1) to handle output bus contention.
(7) One pin to signal a busy output bus (xbusy_i_II).
(8) A pin for the chip reset signal (xreset_1).
(9) A chip select pin for enabling I/O to or from the master (xcs_i_1).
(10) A pin which the master asserts to read an APN register (xrd_1).
(11) A pin which the master asserts to write an APN register (xwr_1).
(12) A pin which requests a value load into an APN register (xvstrb_1).
(13) A pin which requests a load into any APN register (xcstrb_i_1).
(14) A pin which indicates good output to other chips (xwstrb_i_1).
(15) A pin which can override BiMOD's bus control logic (xcenab_i_1).
(16) Two non overlapping clock pins (ph1 and ph2).
(17) Four power supply pins, two and GND and two at Vdd.

Note that the four APNs use the common buses, xraddr_B, xcaddr_B, and xinout_B, and the common signals, ph1, ph2, GND, Vdd, xreset_1, xrd_1, xwr_1, xreqin_1 and xvstrb_1.

XPINSMOD transfers the following external (from off chip) input signals directly into the following internal output signals for use by the other modules of the chip:

(1) To the RIOMOD of each of the four REGMODs:
   xraddr_B -----> raddr_B
   xcaddr_B -----> caddr_B
   xvstrb_1 -----> vstrb_1
   xcstrb_i_1 -----> cstrb_1
   xcs_i_1 -----> cs_1
   xwr_i_1 -----> wr_1
   xrd_i_1 -----> rd_1

(2) To the RFILEMOD of each of the four REGMODs:
   xinout_B -----> in_B

(3) To the BiMOD module in each of the four APNs:
   xcenab_i_1 -----> cenab_1
   xreset_1 -----> reset_1
   xreqin_1 -----> reqin_1
   xbusy_i_II -----> busy_II

XPINSMOD receives the following signals from the indicated modules and generates the indicated signals:

(1) From the RFILEMOD of the ith REGMOD:
   out_B -----> xinout_B (gated by xrd_1).
   r8_B -----> xaout_B (gated by outenab_1 from BiMOD).

(2) From the XPINSMOD, itself:
xrd_1 ----> used to gate out_B.

(3) From the ith OUTMOD module:
  vout_B ----> xvout.i_B (gated by outenab_1 from BiMOD).

(4) From the ith BiMOD:
  wstrb_1 ----> xwstrb.i_1 (gated by outenab_1 from BiMOD).
  outenab_1 ----> used to gate xaout.i_B, xvout.i_B and xwstrb.i_1.
  reqout_1 ----> xreqout.i_1.

The padframe by which XPINSMOD interfaces the APNs to external signals is MOSIS standard frame 84p79x92. It has a die size of 7900u x 9200u and an interior project size of 6620u x 7920u. (The four APNs occupy roughly half the project area.) The padframe package is an 84 pin PGA (pin grid array) with pads numbered from 1 to 84, starting with the #1 pad on the upper right edge of the die, and counting counterclockwise around the die. Pad #10 must be blank.

Seven pad types interface external to internal signals. They are called, "PadIn," "PadOUT," "PadIO," "PadAnalog," "PadGND," "PadVdd," and "PadBlank." (These are the names of standard pad cells put in the public domain by the University of Washington.)

PadBlank is a blank pad, intended for substrate bonding. PadGND and PadVdd are used to route power onto the chip. PadAnalog allows analog signals to move on and off the chip (with protective circuitry on the pad). PadIn provides buffered inputs. PadOUT provides buffered output. PadIO provides a tristate output and an unbuffered input connected to the same pad.

The following lines describe the mapping of pads (and corresponding PGA pins) by number to the signals they carry and to the types of pads used:

Signals common to all APNs:
  10 : blank pad (for substrate bonding); PadBlank (blank pad).
  11, 53: GND; PadGND (special pad for ground line).
  32, 74: Vdd; PadVdd (special pad for power).
  9 : ph2; PadIn (buffered input pad).
  13 : ph1; PadIn (buffered input pad).
  12 : xreset_1; PadIn (buffered input pad).
  51 : xrd_1; PadIn (buffered input pad).
  52 : xreqin_1; PadIn (buffered input pad).
  54 : xvstrb_1; PadIn (buffered input pad).
  55 : xwr_1; PadIn (buffered input pad).
28-31: xinout.B3 - xinout.B0; PadIO (tristate pads), enabled with xrd_1.
33-36: xraddr.B3 - xraddr.B0; PadIn (buffered input pads).
  70-73: xcaddr.B0 - xcaddr.B3; PadIn (buffered input pads).
75-78: xanalog.B0 - xanalog.B3; PadAnalog (analog pads).

Signals local to each APN:
  1-4: xaout.4.B3 - xaout.4.B0; PadIO (tristate pads) enabled by outenab.4_1
 18-21: xaout.3.B0 - xaout.3.B3; PadIO (tristate pads) enabled by outenab.3_1
 43-46: xaout.1.B3 - xaout.1.B0; PadIO (tristate pads) enabled by outenab.1_1
 60-63: xaout.2.B0 - xaout.2.B3; PadIO (tristate pads) enabled by outenab.2_1
  5-8: xvout.4.B3 - xvout.4.B0; PadIO (tristate pads) enabled by outenab.4_1
 14-17: xvout.3.B0 - xvout.3.B3; PadIO (tristate pads) enabled by outenab.3_1
47-50: xvout.1.B3 - xvout.1.B0; PadIO (tristate pads) enabled by outenab.1_1
56-59: xvout.2.B0 - xvout.2.B3; PadIO (tristate pads) enabled by outenab.2_1
22-27: xbusy.3.II, xwstrb.3.I, xreqout.3.I, xcs.3.I, xcstrb.3.I, xcenab.3.I.
64-69: xbusy.2.I, xwstrb.2.I, xreqout.2.I, xcs.2.I, xcsstrb.2.I, xcenab.2.I.

The last four groups of signals have pad types as follows: PadIN (buffered inputs) for xcs.i.I, xcsstrb.i.I, and xcenab.i.I; PadOUT (buffered outputs) for xwstrb.i.I and xreqout.i.I; PadIO (tristate outputs) for xbusy.i.I.

The xbusy.i.I signals pass through the unbuffered input paths of their PadIO pads to their respective BiMOD(i)s. The tristated signals tied to these PadIO pads are GNDs which are gated by xreqout.i.I signals. The xbusy.i.I signals are daisy chained so that an xbusy.i.I signal is asserted (grounded) if the local xreqout.i.I is asserted or if the external signal is low (brought down by some device on its "daisy chain") or both.

1.2. REGMOD: a module for register storage and I/O.

REGMOD contains a file of nine four bit registers and the logic required to control the reading and writing of these registers. Two submodules, RFILE and RIO, implement the register file and the logic, respectively. REGMOD specifies no external (i.e., off chip) interface signals. This specification describes, in order, REGMOD interface signals, the RFILE submodule, the RIO submodule, and REGMOD signal timing.

1.2.1. REGMOD signal specifications.

1. in.B -- Four bit bus used to write a selected register.
3. out.B -- Four bit bus used to read a selected register.
4. rreg.2 -- A signal which enables a register read.
5. wreg.2 -- A signal which enables a register write.
6. r0.B -- A four bit bus tied to the output bits of register 0.
7. r1.B -- A four bit bus tied to the output bits of register 1.
8. r2.B -- A four bit bus tied to the output bits of register 2.
11. r5.B -- A four bit bus tied to the output bits of register 5.
15. cs.1 -- Chip select signal (used by system master).
16. tcs.2 -- A ph1 trapped copy of cs.1, above.
17. wr.1 -- A register read request signal (used by system master).
18. twr.2 -- A ph1 trapped copy of wr.1, above.
19. rd.1 -- A register write request signal (used by system master).
20. trd.2 -- A ph1 trapped copy of rd.1, above.
21. vsstrb.1 -- A "new data value on bus" signal.
22. tvstrb.2 -- A ph1 trapped copy of vsstrb.1, above.
23. cstrb.1 -- A "new weight on bus" signal.
24. tcstrb.2 -- A ph1 trapped copy of cstrb.1, above.
25. raddr.B -- A four bit register address bus.
27. caddr.B -- A four bit chip address bus.
28. teaddr_B - A ph1 trapped copy of caddr_B, above.
29. rs0_2  -- Select line for register 0.
30. rs1_2  -- Select line for register 1.
31. rs2_2  -- Select line for register 2.
32. rs3_2  -- Select line for register 3.
33. rs4_2  -- Select line for register 4.
34. rs5_2  -- Select line for register 5.
35. rs6_2  -- Select line for register 6.
36. rs7_2  -- Select line for register 7.
37. rs8_2  -- Select line for register 8.
38. ph1     -- Signal from a two phase non overlapping clock.
39. ph2     -- Signal from a two phase non overlapping clock.
40. Vdd     -- Power.
41. GND     -- Ground.

NOTE: The register select signals (rs0_2, rs1_2, ... rs8_2) and the rreg_2 and wreg_2 signals are internal to REGMOD. REGMOD also uses the interface bus signal, r8_B, internally for register I/O control. Together, these signals control the interface between the RFILE and RIO submodules.

1.2.2. RFILE submodule.

This submodule is a file of nine registers. Each has a select line by which it can be selected for read or write operations. Each register drives its own output bus. They also share common input and output buses to which access is controlled as described below.

1.2.2.1. Individual register design.

Each register consists of four dlatches (data latches) made from two clocked inverters and one simple inverter. The clocked inverters use exactly opposite signals on their control inputs so that, when the input clocked inverter is open, the second (whose input signal is the inverted output of the first) is closed. Both clocked inverter outputs are tied together. Thus, when the first clocked inverter is open, it charges the gate of the second (which is off); when the first goes off, the second regeneratively feeds its output through the simple inverter to its input, maintaining the data. An array of four latches, with Enable inputs tied to a single register select line, comprises an individual four bit register.

1.2.2.2. Register file design.

Corresponding "Set" inputs of the data latches of each of the nine registers are wired together with the corresponding line of the four bit input bus, tin_B. This input bus is taken from ph1 traps on the bus, in_B (which was passed in from the xinout_B pins by XPINSMOD). Joint assertion of a register's select line, the write signal, wreg_2, and ph2, allows the register's latches to capture tin_B during ph2. Therefore, register contents are ph1 valid.

We gate corresponding bits of the nine register outputs through clocked inverters to the output bus, out_B, so that when a register select signal, the read signal, rreg_2, and ph2 are asserted, the selected register's value is placed on out_B (valid during ph1).

Each register also directly drives its own output bus (each signal buffered by a pair of inverters) (one of: r0_B, r1_B, ... , r8_B). These bus signals are valid during ph1 and can be accessed in parallel by the other modules on the chip.
1.2.3. RIO submodule.

This module contains static control logic performed on ph1 trapped copies of the signals, cs_1, wr_1, rd_1, vstrb_1, cstrb_1, raddr_B and caddr_B, provided to this module by XPINSMOD. These ph1 trapped signals are prefixed by "t" and suffixed by "_2", thusly: tcs_2, twr_2 ... The ph1 traps belong to the RIO submodule.

The register file can be externally accessed (via the buses, in_B and out_B) in a "normal" mode or in a "slave" mode. The normal mode is the usual operation in the context of network activity. The slave mode is a mode where I/O is controlled by an external "master."

1.2.3.1. Normal mode.

If the "load value" strobe signal, tvstrb_2, is high and the chip address bus value, tcaddr_B, matches the chip's ID (see the section on chip address loading, below), and a local chip "run" signal (see section below), cs_I, is high, then (and only then) the data on tin_B will be gated into the register addressed by traddr_B during ph2. (In the network context, if RIO gets a high strobe, tvstrb_2, some other chip is asserting new data (on one or more of tin_B, traddr_B and tcaddr_B) to trigger a register update on the particular chip whose ID = tcaddr_B.)

If the "load weight" strobe signal, tcstrb_2, is high and the chip "run" signal is high, then (and only then) the data on tin_B will be gated into the register addressed by traddr_B during ph2. (In the network context, if RIO gets a high strobe, tcstrb_2, a weight update is being requested by a state machine hardwired to this chip. Thus, no ID matching is required and the caddr_B value is ignored.)

(Note that in both cases, above, any register can be loaded, even though we expect that "weights" and "values" will, by convention, occupy every other register.)

1.2.3.2. Slave mode.

If the "master" asserts the chip select (tcs_2) and write enable (twr_2), RIO asserts wreg_1 so that the register selected by traddr_B loads data from tin_B just as in the normal mode.

If the "master" asserts the chip select (tcs_2) and the read enable (trd_2), RIO asserts rreg_1 so that the contents of the register selected by traddr_B is placed on the out_B bus.

If the "master" asserts the chip select and neither enable signal is high, RIO ignores tcaddr_B, tvstrb_1 and tcstrb_1, and asserts no output signal, thus halting all I/O with RFILE. A local "run" signal, cs_I, is low in just this case; it is otherwise high, indicating that the master is not forcing a pause.

1.2.3.3. Chip Address handling.

In the normal mode, RIO may allow RFILE I/O to happen if it finds a value on the tcaddr_B bus which matches its chip "ID", and tvstrb_2 is asserted. By convention, the ninth register in RFILE (selected by rs8_2) contains this four bit ID. Therefore the corresponding register bus, r8_B, extends from RFILE to RIO.

The ID can be loaded by the master (see the section on slave mode I/O, above). The ID register's latched output will always be available for testing against input from tcaddr_B (by ANDing the XNOR of the respective bits of tcaddr_B and r8_B).

1.2.3.4. Register address decoding.

Each register in the register file is accessed using a select line which is asserted if the register's address is found on traddr_B. The select signal is generated by a four input NAND gate which is tied to four of eight signals: the traddr_B signals and their inverted signals (which four determines the
address responded to). A register cannot be loaded or read unless its address is selected and wreg_2 or rreg_2 is asserted. Select signals are stable during ph2.

1.2.4. Bus and signal timing. (See figure titled: REGMOD Signal Timing.)

All input signals to the RIO submodule (cs_1, wr_1, rd_1, vstrb_1, cstrb_1, raddr_B, and caddr_B) are assumed ph1 valid. REGMOD traps them on the trailing edge of ph1 so that they are internally valid during ph2. During ph2 RIO's output signals (rreg_2 and wreg_2 and the nine select signals rs0_1, rs0_2, rs1_2 ... rs8_2) arrive as input to the RFILE submodule.

Since RFILE's control signals are valid in ph2, the registers load their values during ph2. The register outputs are therefore stable during ph1. The input data bus (in_B), however, must be trapped for use during ph2.

(The above mentioned ph1 traps are implemented as a pair of pass transistors or complementary gates, gated by ph1 and phII.)

---

1) All input signals to REGMOD must be valid during ph1.
2) ph1 traps the input signals. Trap outputs are valid during ph2.
3) Input dependant REGMOD control signals created and used during ph2.
4) Register select and enable signals permit register loading during ph2.
5) Latched register output is available in ph1 (one cycle after requested).

REGMOD Signal Timing
2. APMOD: the APN analog processor module

The APMOD contains the circuits which convert the weights and values stored in the input register to an analog voltage/current representation, performs the sum of products computation and converts the output to a digital form which is passed to the output register.

APMOD has eight bus input signals r0.B through r7.B. These signals are valid during ph1. APMOD delivers one bus signal as output, value_B, valid during ph2. All bus signals are four bits wide. The APMOD has one additional digital input, APCLK which is connected to PH2, and two analog inputs APBIAS and APSCALE which control circuit operating conditions.

The APMOD circuits are wired to implement the APN function for four bits of numerical precision, however in the future it may be desireable to build APN's with greater precision. In order to test the practicality of using analog computation the design goal for the present circuits is 8 - 10 bits of accuracy. The APN has been designed to provide very high speed computation, circuit simulations indicate that the APN can compute the sum of products operation in 50 ns.

2.1. APN Processor Block Diagram.

The block diagram of the APN analog processor is shown in Fig. 2. Each value/weight input pair is multiplied by the DACs and the product is expressed as a current on the current summing node. The summed currents are converted to a digital representation by the "squishing ADC" and passed to the output bus by the encoder circuit. The AP_CONTROL circuit provides buffered clocks to the ADC and encoder. The IREF circuit provides scaled reference currents to the DACs and the ADC.

2.2. Multiplying DAC.

The multiplying DAC is comprised of two current steering DACs connected such that the output current of the value DAC sets the full scale current of the weight DAC. The schematic of the VALUE DAC is shown in Fig. 2.1. The current steering switches connect the outputs of the binary weighted current mirror array to the output or to VSS depending on the state of value bus V[0..3]. The full scale current of the VALUE DAC is set by IREF circuit which sinks a reference current from the IR node. The current mirror array uses the "Wilson" configuration for speed. The circuit topology of the WEIGHT DAC is shown in Fig 2.2.

2.3. Squishing ADC.

All of the WEIGHT DAC outputs are tied together at the current summing node ISUM and the combined currents are sensed as a voltage developed across the load resistor which is connected to the analog to digital converter input comparators. The ADC circuit is shown in Fig. 2.4. The 15 comparators quantize the input to one of 16 values which are encoded to straight binary form by the encoder logic. The squishing function is performed by adjusting the values of the reference resistor string for higher gain in the middle of the range. The reference string resistors and the load resistor are implemented in polysilicon.

2.3.1. ADC Comparator.

The schematic for the analog to digital converter comparator is shown in Fig 2.5. M1 and M2 form the differential input amplifier which is loaded by the cross coupled latch M6 and M7 which is clocked by M4. The first stage outputs are connected to the similar second stage differential amplifier M8, and M9 which is loaded by the M12 and M13 latch and clocked by M11.
2.3.2. ADC Encoder.

The ADC encoder is comprised of a nor plane followed by a trap and buffer which drives the output value bus.

2.3.3. AP Control

The AP control circuit buffers and delays the single phase clock that times the operation of the ADC and ENCODER circuits. It is shown in Fig. 2.7.

2.3.4. IREF Current Reference

The reference current source circuit shown in Fig. 2.3 sets provides the reference for the four DACS and the establishes the voltage developed across the reference resistor string in the ADC.

2.4. OUTMOD: the APN output module.

This module takes input from APMOD, REGMOD, and BiMOD (which handles output bus contention) and outputs a result when the bus is available and the APN’s computation differs from the previous one(s).

2.4.1. OUTMOD signal specifications.

1. value_B -- The four bit computation of APMOD (input to OUTMOD).
2. tval_B -- A ph1 trapped copy of value_B (valid in ph2).
3. vout_B -- The new four bit output value (output from OUTMOD).
4. tvout_B -- A ph1 trapped copy of vout_B (valid in ph2).
5. outreq_1 -- A signal which OUTMOD sends to BiMOD requesting the bus.
6. wstrb_1 -- A signal from BiMOD which OUTMOD uses to reset outreq_1.
7. newd_2 -- A temporary result of the comparison of tval_B and tvout_B.
8. clocks -- A ph1 and ph2; self explanatory.

2.4.2. OUTMOD operation.

APMOD sends OUTMOD a four bit bus signal which corresponds to a newly computed value. This input (value_B) is ph1 trapped into tval_B on every ph1. OUTMOD generates a signal, newd_2, by ANDing the pairwise XNOR of corresponding bits of tval_B and tvout_B (ph1 trapped vout_B signals). This signal goes high whenever APMOD computes a signal different from the last one to leave the chip.

If newd_2 is high during ph2, tval_B is loaded into a four bit data latch, providing BiMOD’s outenab_1 signal is low. A local signal, toutenab_2, is ph1 trapped to test during the ph2 generation of the latch enable signal(s). (The latch contents should remain stable while outenab_1 is gating the data bits onto the external bus.) At the same time, a '1' is loaded into a one bit data latch. These latches are named vout_B and outreq_1, respectively.

The outreq_1 signal is sent to BiMOD and requests that BiMOD handle bus contention and determine when to gate the vout_B signal through XPINSMOD’s tristate drivers onto the external pins. The outreq_1 signal can be reset to low only with a high wstrb_1 signal coming into OUTMOD from BiMOD (indicating that the new data are now on the bus).
The global reset signal (from XPINSMOD), reset_1, is used to reset the data latches and the outreq_1 signal during ph1. (Note that the data latches are constructed like those in REGMOD's RFILE submodule, out of one inverter and two clocked inverters, except the inverter is replaced by a two input NOR device, where one input is tied to a reset signal (AND of ph1 and reset_1) and the other is the output of the input clocked inverter... so the device behaves like a simple inverter unless a reset signal is asserted, when the latches are reset low.)
Appendix A

APN Chip Layout Data
September 30, 1987

3. Tabulation of floor plan area, number of placements and gate counts.

The following table (Table 1) describes the gate counts, layout area, number of cell placements required and number of required connections for each module or submodule of the APN chip. The number of transistors is exact. The floor area is exact, but does not include area taken by the large buses which surround the four APNs and interface their signals with XPINSMOD and the padframe. We have made rough estimates of the number of cell placements required, as computed in a "recursive" fashion. (Since we used composite cells already created wherever possible, those count as one placement.) The number of connections is the approximate count of the number of signal paths which were be placed between interacting modules.

<table>
<thead>
<tr>
<th>Module</th>
<th>Gates</th>
<th>Area (x10^3 μm^2)</th>
<th>Placements</th>
<th>Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPINSMOD</td>
<td>1720</td>
<td>19300</td>
<td>38</td>
<td>54</td>
</tr>
<tr>
<td>RFILEMOD</td>
<td>846</td>
<td>7601</td>
<td>12</td>
<td>104</td>
</tr>
<tr>
<td>RIOMOD</td>
<td>136</td>
<td>1787</td>
<td>11</td>
<td>38</td>
</tr>
<tr>
<td>APMOD</td>
<td>683</td>
<td>6353</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>OUTMOD</td>
<td>112</td>
<td>1817</td>
<td>24</td>
<td>100</td>
</tr>
<tr>
<td>BiMOD</td>
<td>161</td>
<td>978</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Totals*</td>
<td>3658</td>
<td>37836</td>
<td>155</td>
<td>416</td>
</tr>
</tbody>
</table>

NOTE: The APN chip has one XPINSMOD and four of each of the remaining modules (RFILEMOD, RIOMOD, APMOD, OUTMOD, and BiMOD). Therefore the totals reflect the layout design effort, NOT the chip total gates, etc.!

4. Register array sizes.

Recall that each register array (RFILEMOD) has nine four-bit registers. They are somewhat special in that every register output is both (a) buffered by an inverter pair, and (b) tristated onto an output bus Through a clocked inverter. Each register is made from a row of four data type latches made from two clocked inverters and a simple inverter. Four latches with associated clocked inverters take up 393(10)^3 μm^2. The decoding circuitry which creates the ctrl and INVT signals for the four latches requires a four-input NAND gate, two two-input NAND gates and two simple inverters. They take up 281(10)^3 μm^2. A single four bit register, with decoder and output buffers, takes up 844(10)^3 μm^2.

We are certain we can reduce the floor space required by these registers and decoders by more than fifty percent. For this reason, we believe we can put as many as sixty four APNs on a chip this size, using the identical technology.
5. Estimated hours required for mask design.

There were an estimated six hundred or so hand placed cells or hand placed connections in the mask design for the APN chip. We spend around three hundred man hours designing the mask. We conclude that we spent about thirty minutes per hand placement of a preexisting cell or drawing of a connection.
Figs 2.1 ANALOG PROCESSOR BLOCK DIAGRAM
A/D CONVERTER COMPARATOR CELL

FIG 2.5
OUTPUT 1 of 16 to BINARY ENCODER
FIG 2.6
AP CONTROL (CLOCK CIRCUIT)

FIG 2.7
APN Test Results

Of ten APN chips tested, eight showed some functionality at 5 MHz, 1 MHz and 0.5 MHz. This functionality was very consistent from chip to chip. Two chips appeared to be "stuck" in a fixed state. Despite being driven at five volts against ground, all chips showed a substantial voltage drop to ca. 2 volts against ground when inserted into the logic tester fixture. They do not, however, heat up under these conditions, suggesting that there is a high (not low) resistance path from Vdd to GND.

In view of our extensive prefabrication tests of the layout design against netlists and connectivity criteria, we suspect that the fault lies in the analog portion of the chip where power and grounds are intentionally connected together through polysilicon runs, precluding the simple but compelling connectivity tests we performed on the digital portion.

We believe that the problem resulted from a conflict with Magic placing wells automatically, which conflicted with the wells we placed explicitly for the analog section.

Bias voltages inside the analog section were tested on the four external pins reserved for this purpose. They were within 10% of the specified voltages.

We will be trying to fix the APN and resubmit for fabrication this June.