



DPS-4D Preprocessor Card

George Cheney

University of Massachusetts Lowell
Center for Atmospheric Research

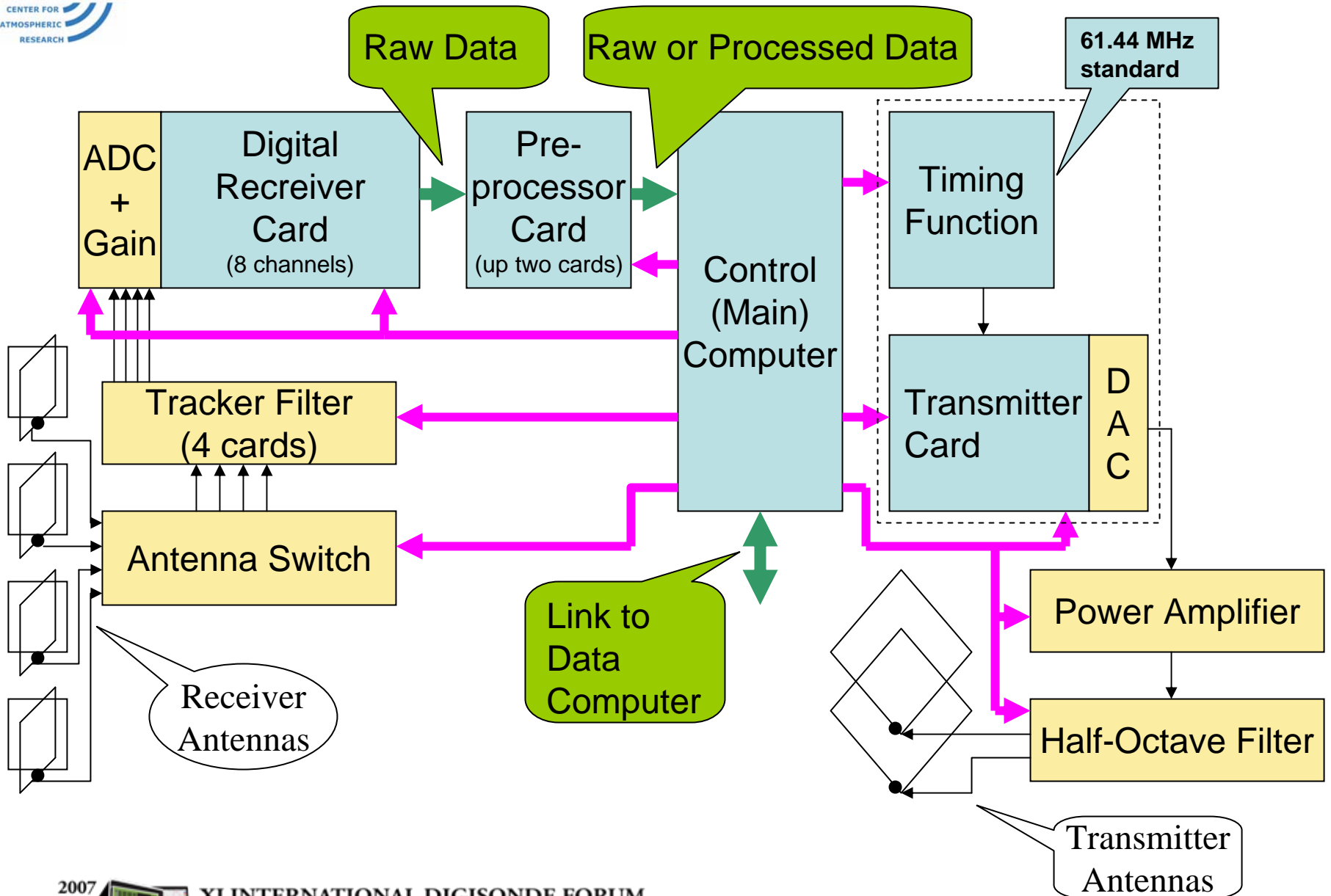
- and -

Department of Electrical and Computer Engineering



XI INTERNATIONAL DIGISONDE FORUM
30 APRIL TO 3 MAY 2007

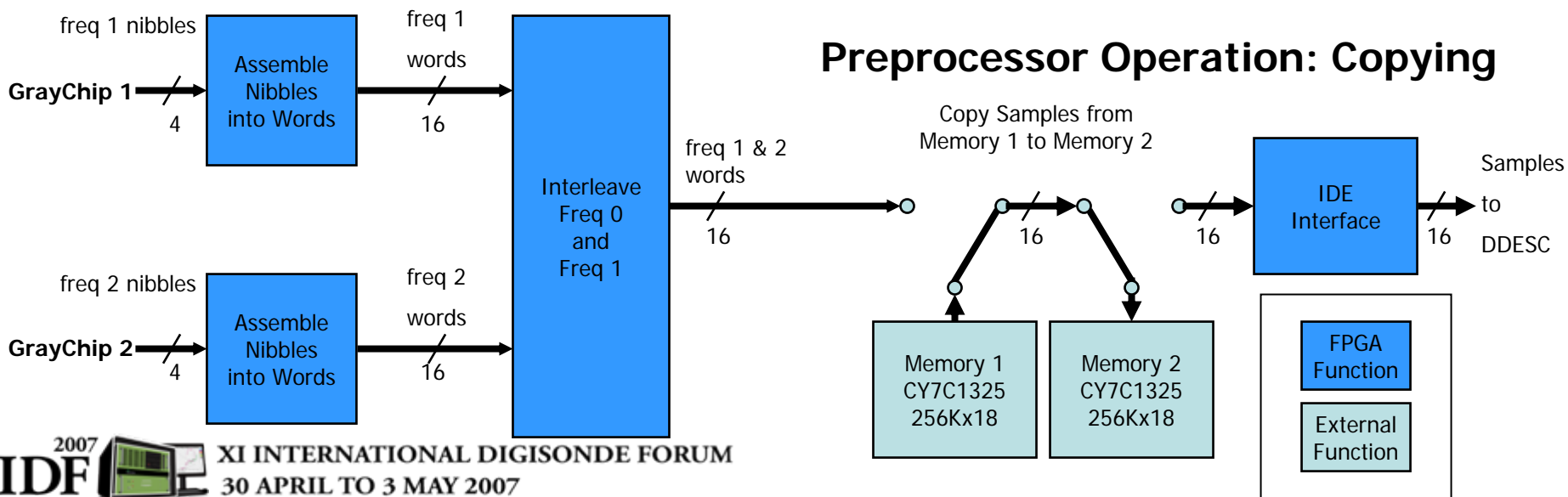
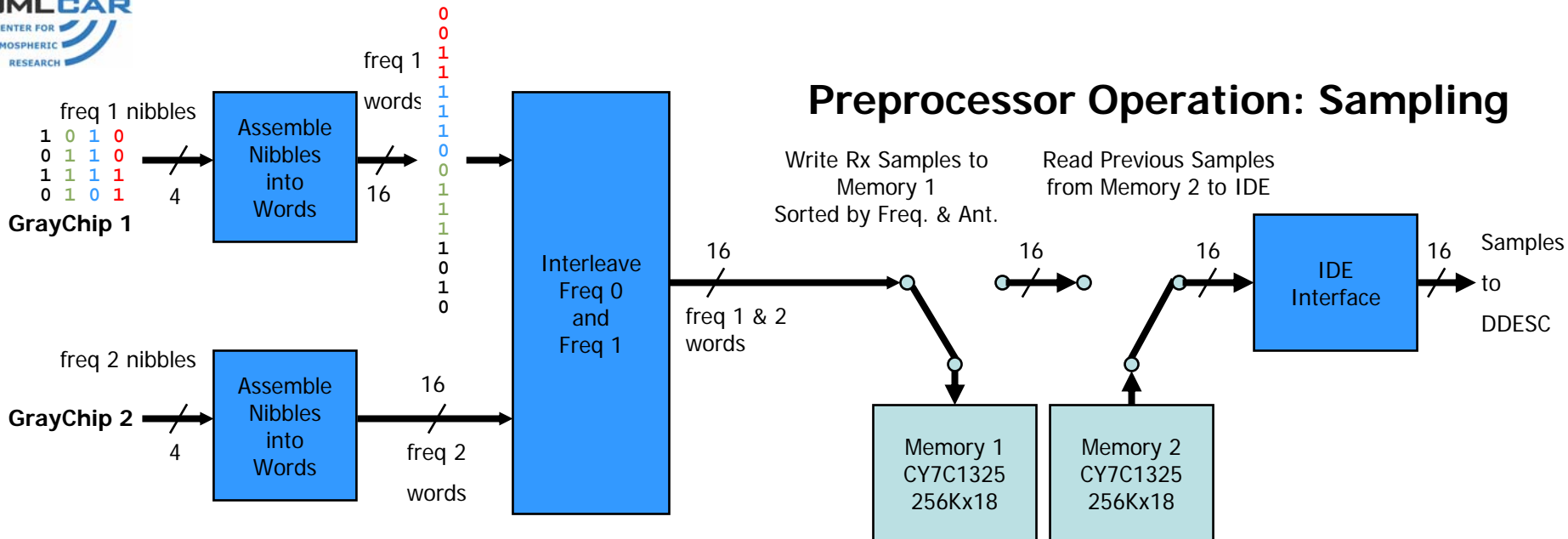
DPS-4D Digital Transceiver Block Diagram



Preprocessor Functions

- Gather Rx Samples from the Digital Receiver Card
 - Convert GrayChip Digital Down Converter Samples from 4-bit Nibbles to 16-bit words.
 - Interleave frequency 1 and frequency 2 samples.
 - Sort samples by frequency and antenna for delivery to DDESC.
 - IDE Interface to DDESC
- Future Processing Functions
 - High speed DMA interface for 1.25 km sampling
 - RFI Mitigation in hardware (Walter Jones - nearly completed)
 - Twin Frequency decoding in hardware

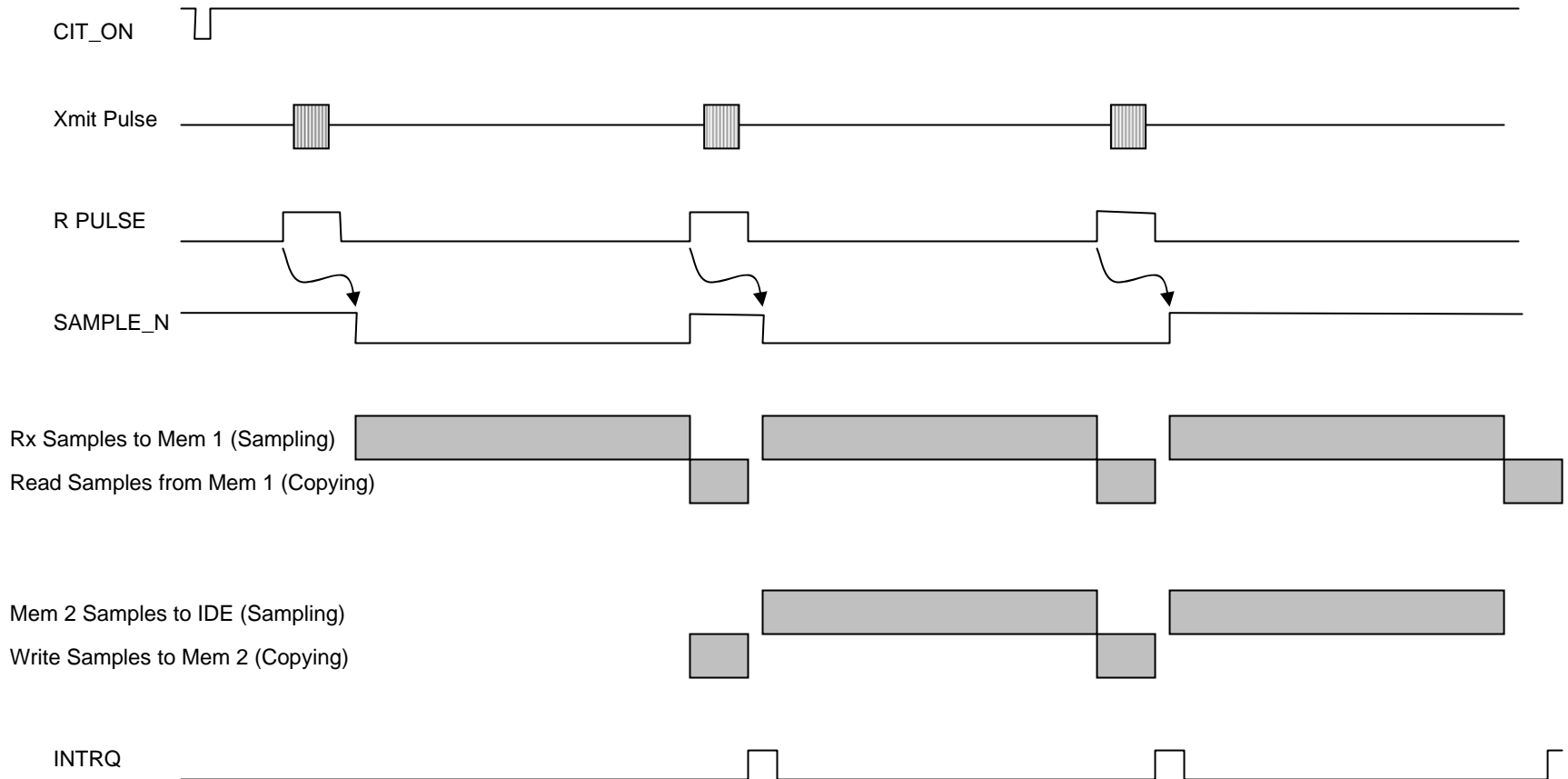
Pre-Processor Card Block Diagram



Sample Sorting

Preprocessor Input Sequence (one of two frequencies)			Preprocessor Output Sequence		
Height	Antenna		Frequency Number	Antenna Number (Memory Address)	Heights
	Frequency 1	Frequency 2			
1	1	1	1	1 (0x00000)	1 to 256
	2	2		2 (0x08000)	1 to 256
	3	3		3 (0x10000)	1 to 256
	4	4		4 (0x18000)	1 to 256
2	1	1	2	1 (0x20000)	1 to 256
	2	2		2 (0x28000)	1 to 256
	3	3		3 (0x30000)	1 to 256
	4	4		4 (0x38000)	1 to 256
⋮	⋮	⋮			
256	1	1			
	2	2			
	3	3			
	4	4			

Preprocessor Timing



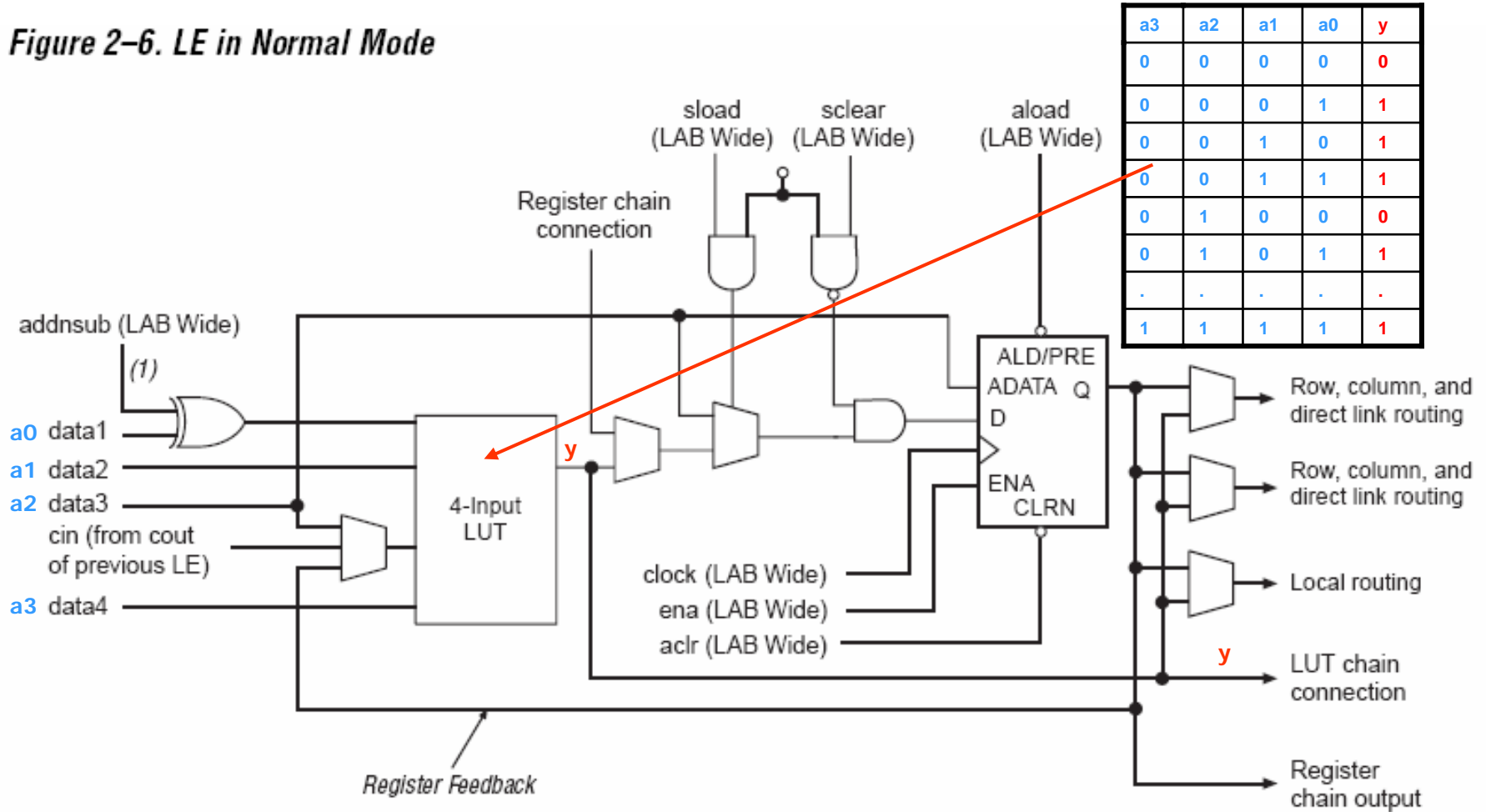
Preprocessor New Hardware Techniques

- Field Programmable Gate Array (FPGA) for Complex Digital Logic
- Verilog Hardware Description Language Replaces Schematic Design
 - Behavioral level to gate level
 - Simpler design than schematic capture
 - Portable (Altera to Actel or Aeroflex)
- New Design Tools
 - Altera's Quartus II & ModelSim
 - Gate synthesis
 - Chip place and route
 - Simulation (functional and timing)

Stratix FPGA Logic Element

$$y = a0 + a1 + a2 \& a0 + a0 \& a1 \& a2 \& a3$$

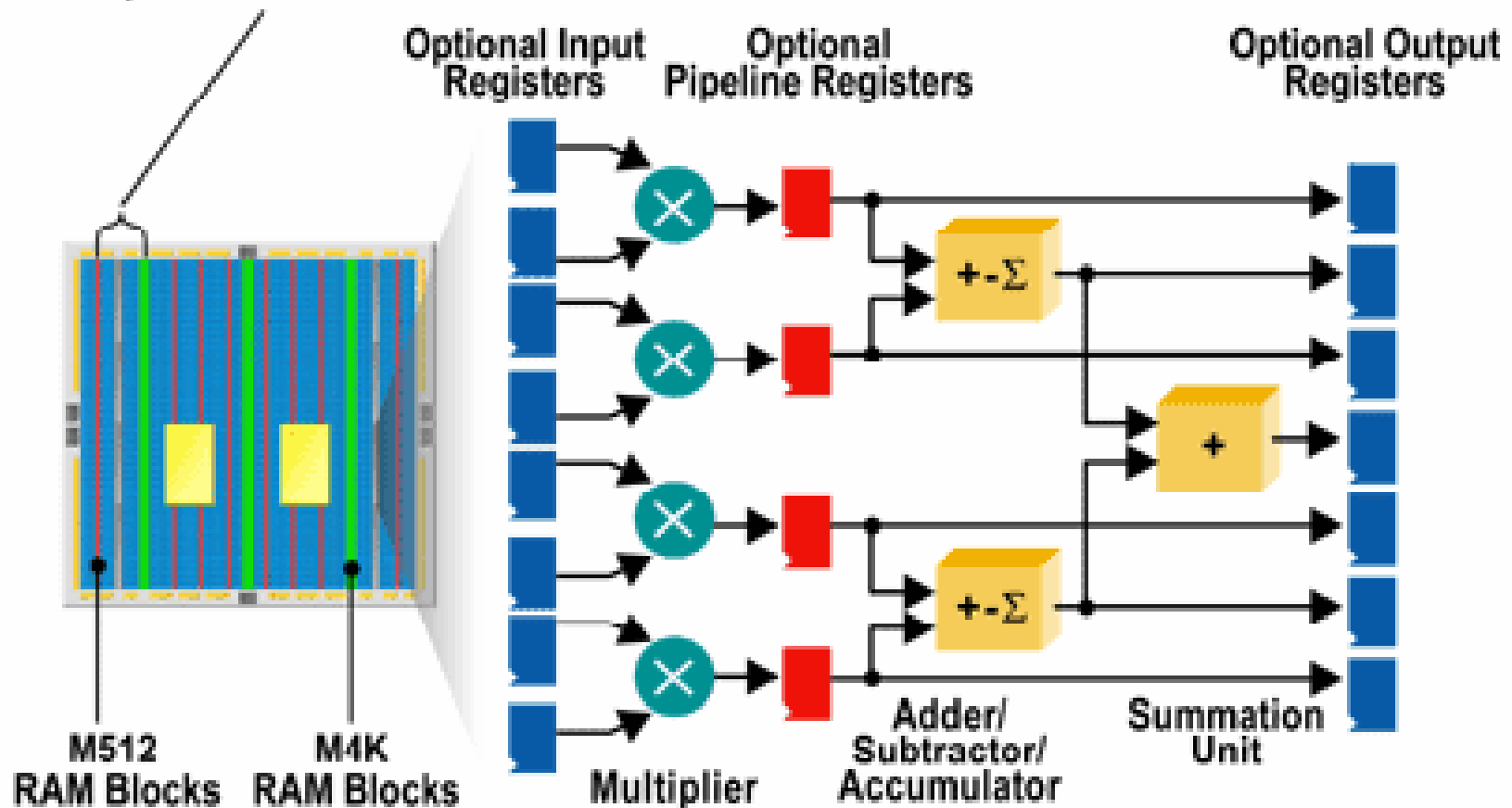
Figure 2-6. LE in Normal Mode



Stratix DSP Block (10 available)

Figure 2. DSP Block

Memory & DSP Blocks Placed for Optimum Data Transfer



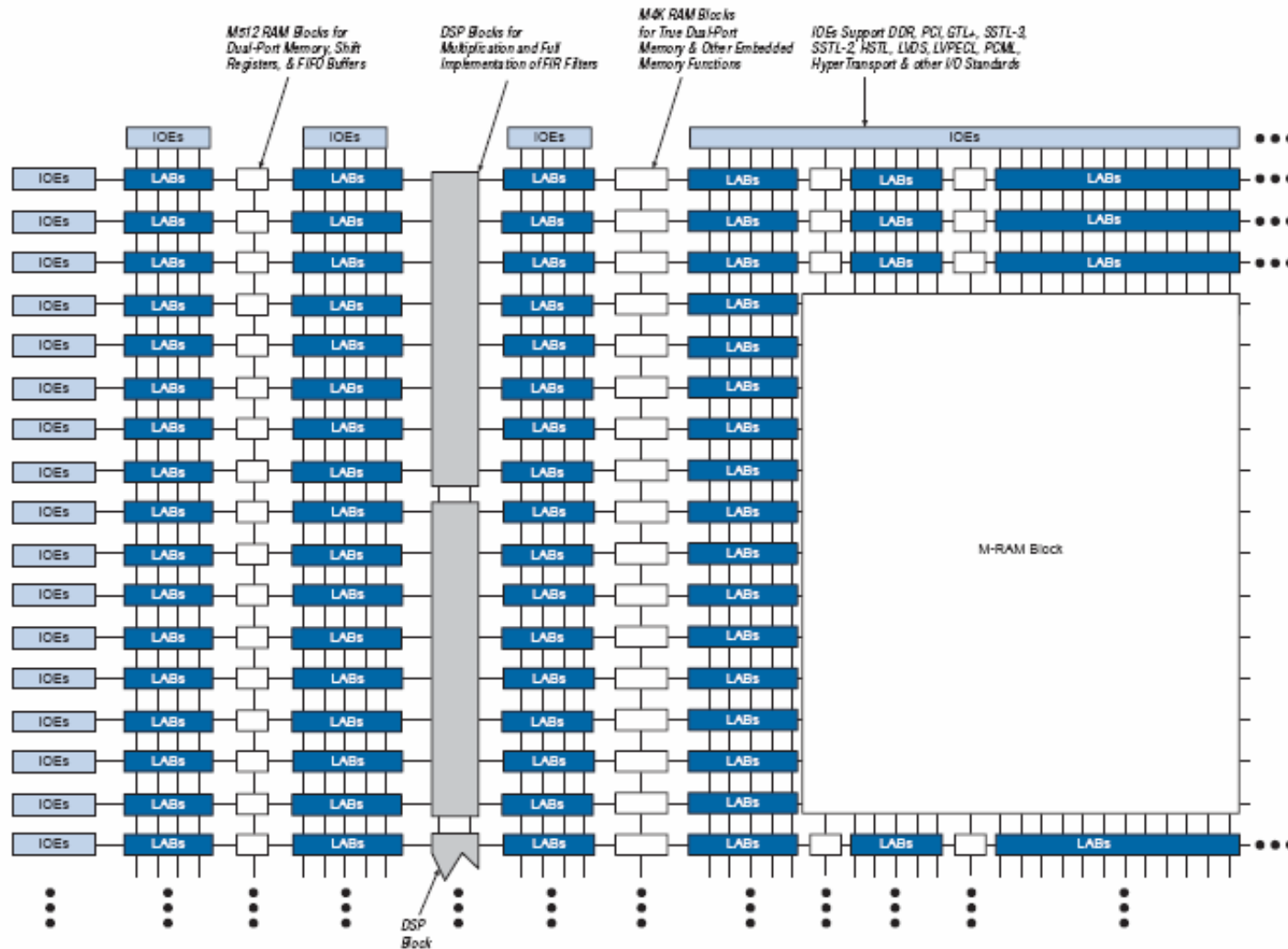
Stratix EP1S25 Resources

Table 1-1. Stratix Device Features — EP1S10, EP1S20, EP1S25, EP1S30

Feature	EP1S10	EP1S20	EP1S25	EP1S30
LEs	10,570	18,460	25,660	32,470
M512 RAM blocks (32 × 18 bits)	94	194	224	295
M4K RAM blocks (128 × 36 bits)	60	82	138	171
M-RAM blocks (4K × 144 bits)	1	2	2	4
Total RAM bits	920,448	1,669,248	1,944,576	3,317,184
DSP blocks	6	10	10	12
Embedded multipliers (1)	48	80	80	96
PLLs	6	6	6	10
Maximum user I/O pins	426	586	706	726

Stratix FPGA block Diagram

Figure 2-1. Stratix Block Diagram



Verilog Module

```

module InterlaceAandB(  addr,      // Address to write
                       wData,     // Interlaced write data
                       rdyOut,    // Interlaces data word ready
                       rdyIn,     // A and B inputs ready
                       dataA,     // A input data
                       dataB,     // B input data
                       clk,       // System clock
                       reset);    // Global reset

```

```
`include "globals.v"
```

```

output [`AW-1:0] addr;           // Address to write
output [`DW-1:0] wData;         // Data to write
output reg rdyOut;              // Output ready flag
input rdyIn;                    // Input ready status bit
input [`DW-1:0] dataA;          // A input data
input [`DW-1:0] dataB;          // B input data
input clk;                       // System clock
input reset;                      // Global reset

```

```
// State Machine Controller
```

```
// State assignments
```

```

parameter  S0   = 0,
           S1   = 1,
           S2   = 2,
           S3   = 3,
           S4   = 4,
           S5   = 5,
           S3a  = 6,
           S3b  = 7,
           S3c  = 8;

```

```
// Current state register
```

```

reg [3:0] state;

always @(posedge clk or posedge reset)
  if (reset)
    state <= S0;
  else
    state <= nextState;

```

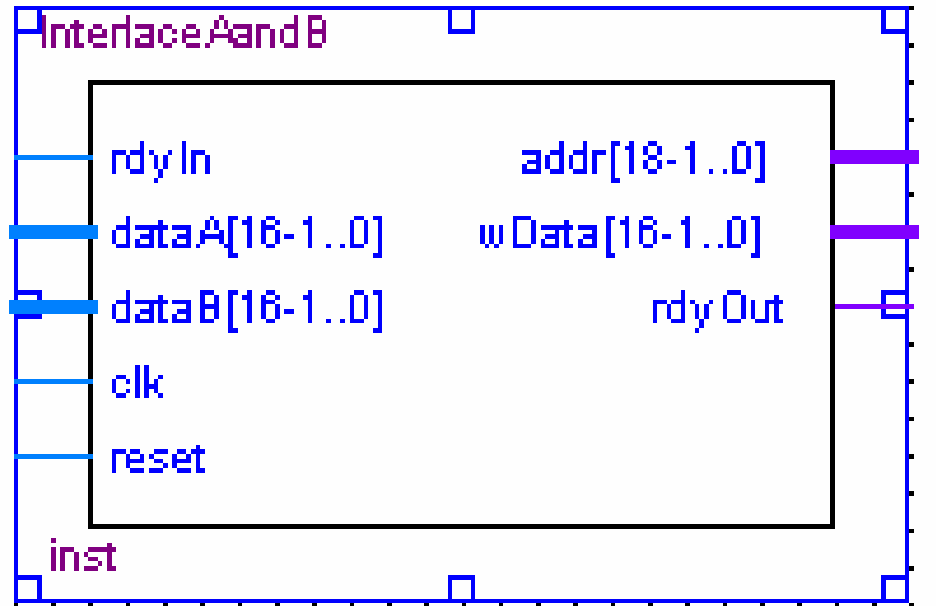
```

reg [3:0] nextState; // Next state
always @* // Next state generation logic
begin
  nextState = state;
  case (state)
    S0:  nextState = S1;
    S1:  if (rdyIn)
          nextState = S2;
    S2:  nextState = S3;
    S3:  nextState = S3a;
    S3a: nextState = S3b;
    S3b: nextState = S3c;
    S3c: nextState = S4;
    S4:  nextState = S5;
    S5:  if (rcvrNum)
          nextState = S2;
        else
          nextState = S1;
  endcase
end
assign wData = (rcvrNum) ? dataB : dataA;
wire rcvrNum = cntr[0] /* synthesis keep */;
wire [1:0] antNum = cntr[3:2] /* synthesis keep */;
wire q = cntr[1] /* synthesis keep */;
wire [`AW-9:0] smplNum = cntr[`AW-5:4] /* synthesis keep */;
assign addr = {rcvrNum, antNum, 4'b0, smplNum, q};
reg [`AW-5:0] cntr;

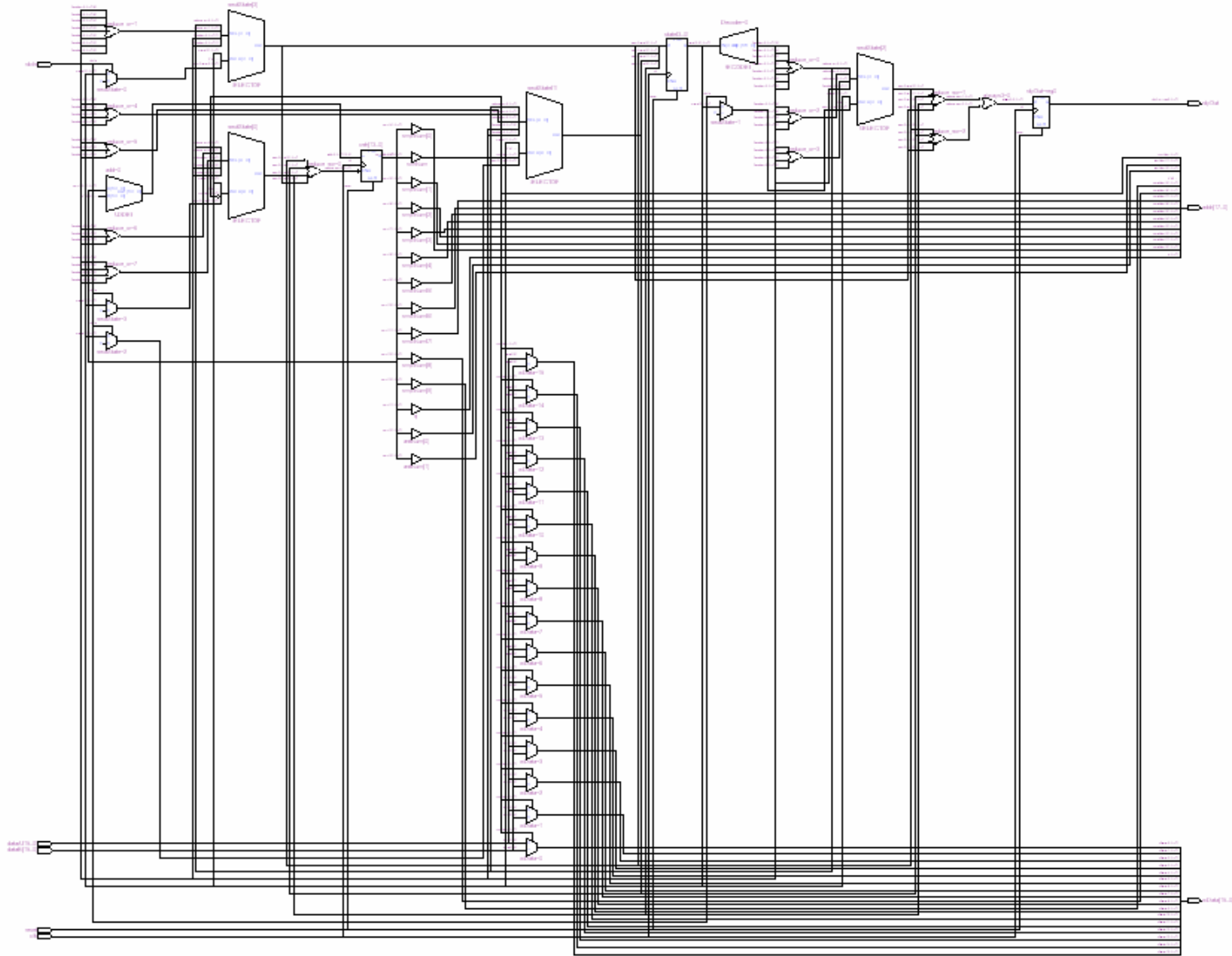
always @(posedge clk or posedge reset)
  if (reset)
    cntr <= 0;
  else if (nextState == S4)
    cntr <= cntr + 1'b1;
always @(posedge clk or posedge reset)
  if (reset)
    rdyOut <= 1'b0;
  else if (nextState == S2 || nextState == S3)
    rdyOut <= 1'b1;
  else
    rdyOut <= 1'b0;
endmodule

```

Module Symbol



Synthesized Logic – Quartus RTL



Preprocessor FPGA Resource Utilization

