ALTERA M9K
EMBEDDED MEMORY BLOCKS

M9K Overview

- M9K memories are Altera's embedded high-density memory arrays
  - Nearly all modern FPGAs include something similar of varying sizes
- 8192 bits per block (9216 including parity bits)
- Highly flexible port configurations
- In general, embedded array memories will perform much better than memories synthesized from LUTs
  - Higher clock rates / higher throughput / lower latency
  - Lower energy dissipation
  - Lower use of chip resources
  - Exception: very small memories

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyclone-iv/cyiv-51003.pdf

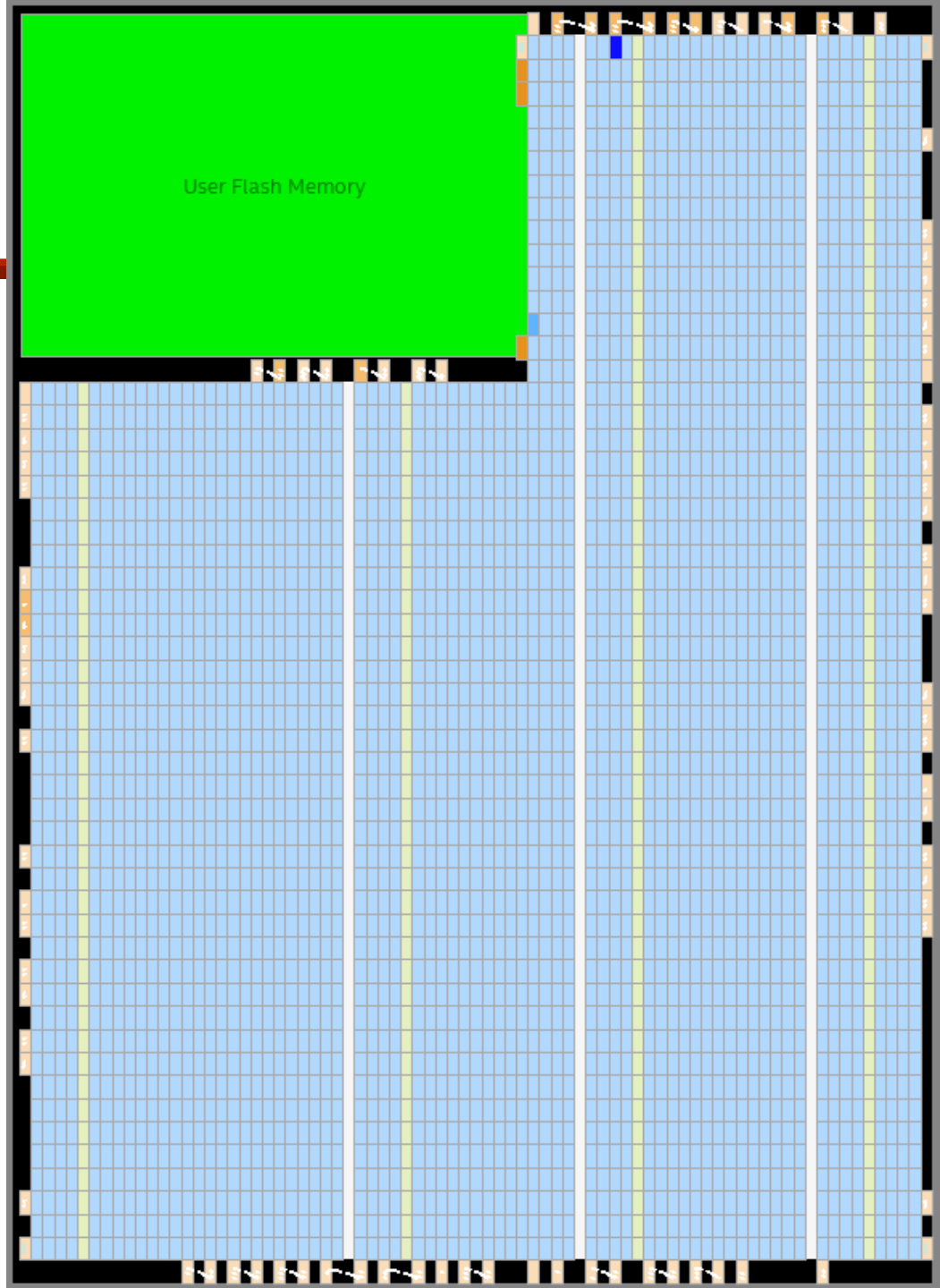
Data Initialization Capabilities

- ROMs
  - The embedded memory array is truly an SRAM acting like a ROM so its contents must be initialized
- SRAM
  - Unique to FPGAs, the contents of SRAMs may be initialized at configuration time
- Contents are specified in verilog in an **`initial`** block
  - This is the only time you may synthesize an **`initial`** block!
- Initialization data contents are specified with a .mif file by Quartus

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyclone-iv/cyiv-51003.pdf

FPGA Chip

- Max 10 10M50DAF484C7G chip
- **Yellow rectangles are M9K memory blocks**
  - **182 blocks on each chip**
  - **Total of 182 KBytes (204 KB)**
- Light-blue rectangles: Logic Array Blocks (LAB), each of which contains 16 logic elements (LE), each of which contains a 4-input LUT, a flip-flop, and routing muxes
- White rectangles: hardware 18x18 multipliers
- Green rectangle: on-board flash memory that can store the bit-stream that programs the FPGA when it is powered on
- Brown blocks on the border are I/O ports and drivers



M9K Size Configurations

- Supported configurations per memory block

Number of words (words)	Number of bits in words (bits)
8192	1
4096	2
2048	4
1024	8 or 9
512	16 or 18
256	32 or 36

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyclone-iv/cyiv-51003.pdf

M9K Interface Modes

- Single port
- Simple dual-port
  - Supports simultaneous read and write operations to different locations
- True dual-port
  - Supports any combination of two-port operations: two reads, two writes, or one read and one write, at two different clock frequencies
- Shift register
- ROM
  - 1 port or 2 port
- FIFO

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyclone-iv/cyiv-51003.pdf

M9K Details

- Independent read-enable and write-enable signals for each port
- Packed mode in which the M9K memory block is split into two 4.5 K single-port RAMs
- True dual-port (one read and one write, two reads, or two writes) operation
- Byte enables for data input masking during writes
- Two clock-enable control signals for each port (port A and port B)

https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyclone-iv/cyiv-51003.pdf

Four Main Methods to Specify an M9K

1. Let Quartus infer an M9K from appropriate verilog (generally the best approach)
2. Use the IP catalog tool (see an example in the PLL Tutorial)
3. Use Quartus QSYS (not recommended, #3 is better)
4. Use a Quartus "Language Template"
   - Edit > Insert Template > Verilog > Full Designs > RAMs and ROMs

- See the Compilation Report to find out if M9K blocks were really used during synthesis

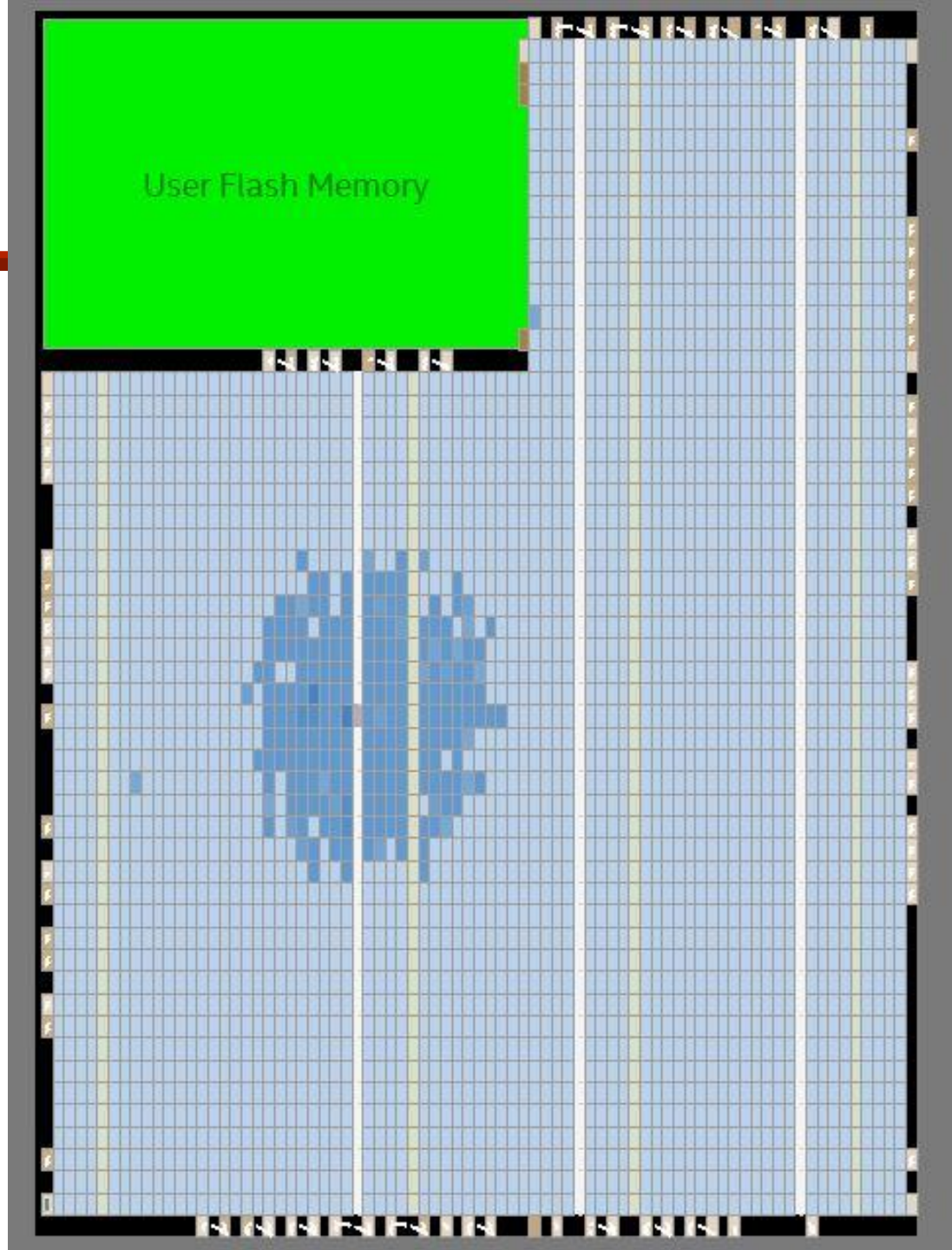
M9K Basic SRAM Template

- The "**`synthesis ramstyle`**" pragma comment is not necessary for Quartus to infer a M9K block but it is a helpful bit of documentation and explicitly states what the designer wants
- With this pragma, Quartus will either use an M9K or print a warning

```
module basic_ram(
   input clk,
   input wr_en,
   input  [7:0] data_in,
   output [7:0] data_out,
   input  [6:0] address
   );

   reg [7:0] mem [127:0]  /* synthesis ramstyle = M9K */;

   // To initialize the RAM, Quartus supports initialization
   // which normal RAMs and synthesis do not support.
   // initial begin
   //   mem[0] = 8'b0000_0000;
   //   mem[1] = 8'b0000_0001;
   //   mem[2] = 8'b1000_1000;
   //   ...
   //   mem[127] = 8'b1111_1111;
   // end

   always @(posedge clk) begin
      if (wr_en == 1'b1) begin
         mem[address] <= data_in;  // write
      end
      data_out <= mem[address];     // read
   end
endmodule
```

Example Design Utilizing LUT Memory

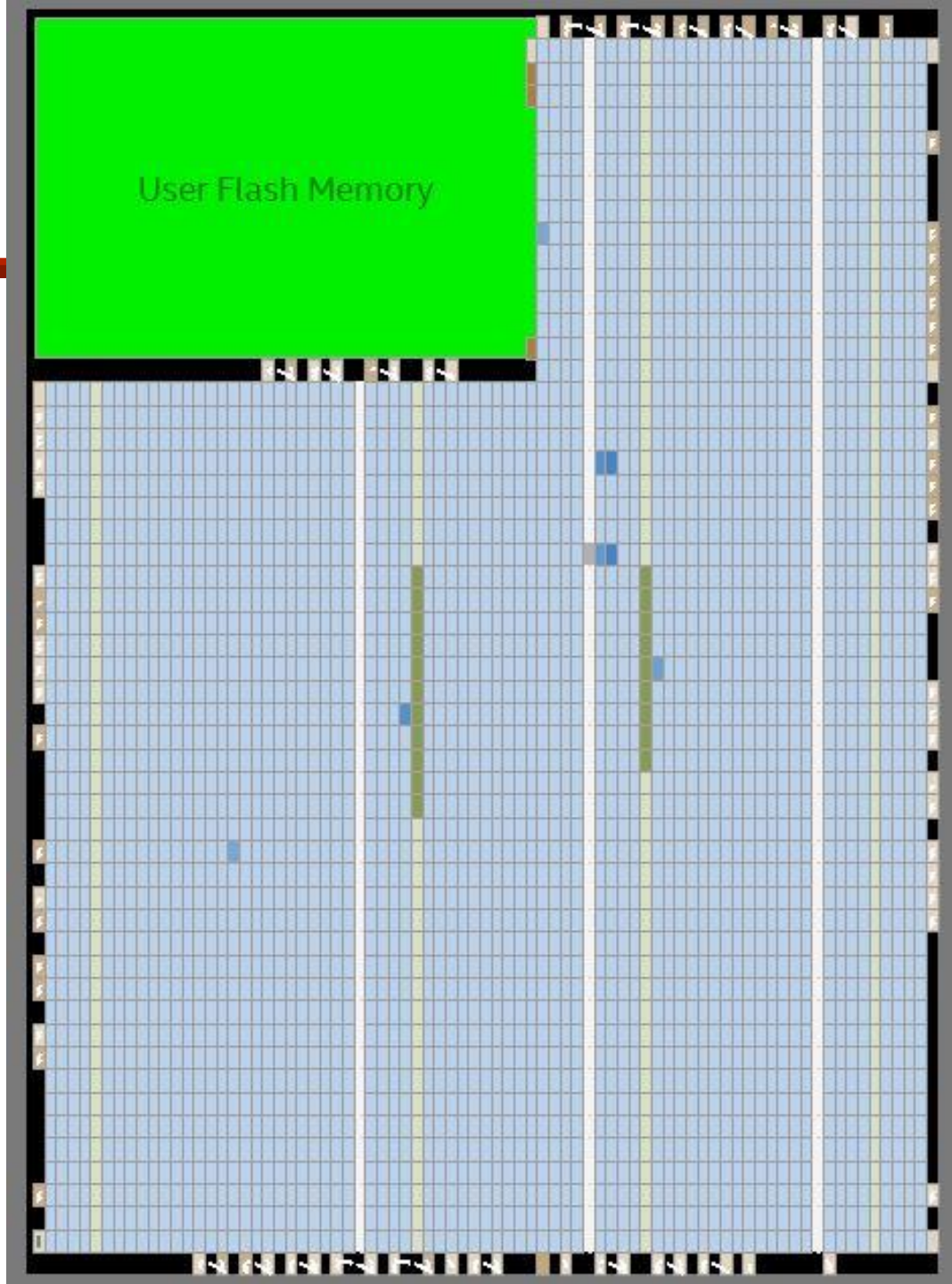
- In this example, the M9Ks are not enabled and the large ROM memories are implemented using individual Logic Elements



Diagrams courtesy Justin Salazar

Example Design Utilizing Block RAM Memory

- In this example, the M9Ks are enabled
- Many Logic Elements are freed for other uses
- Should have a higher maximum clock frequency
- Should dissipate lower power



Diagrams courtesy Justin Salazar