INTEL/ALTERA M9K & M10K EMBEDDED MEMORY BLOCKS

Updated January 19, 2022

M9K & M10K "Block RAM" Overview

- M9K and M10K memories are Intel/Altera's embedded high-density memory arrays
  - Nearly all modern FPGAs include similar "block memories"
- Each block contains approximately 9000 or 10,000 bits of memory per block respectively
- They have 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 other chip resources (e.g., LUTs)
  - Exception: very small memories

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

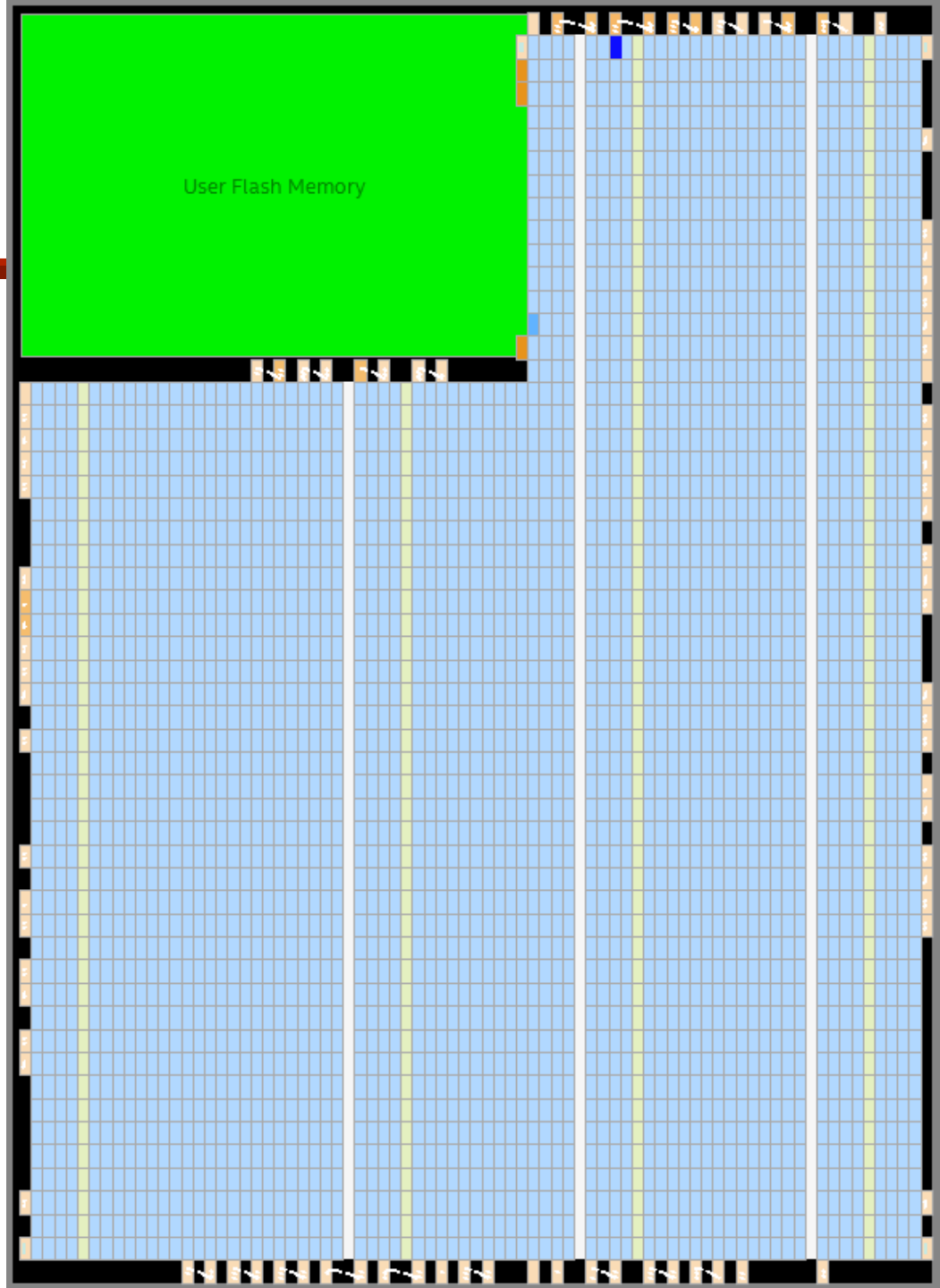
Data Initialization Capabilities

- 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

M10K Size Configurations

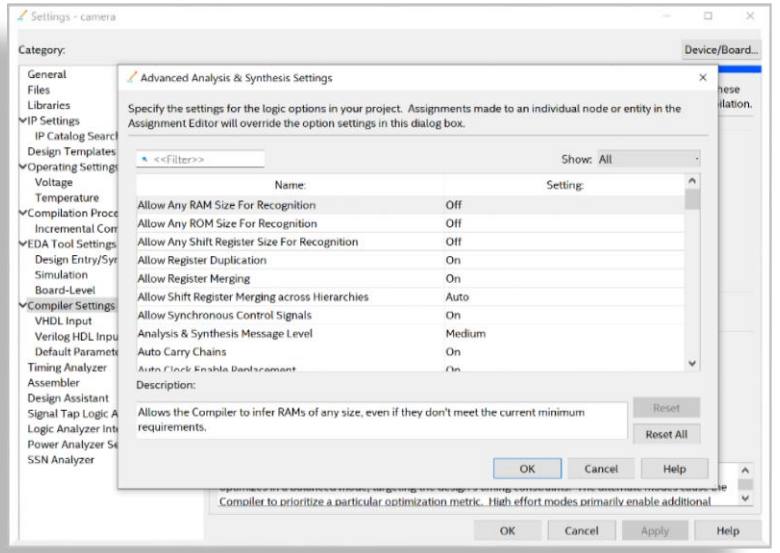
- Supported configurations per memory block

Number of words (words)	Number of bits in words (bits)
4096	2 or 1
2048	4 or 5
1024	8 or 10
512	16 or 20
256	32 or 40

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

M9K & M10K Memory Sizes

- For the Quartus compiler to select an M9K or M10K memory block, the number of words in a declared verilog memory is constrained as follows:
  1. The number of words in a memory must be a power-of-2
  2. The number of words in a memory may be any value if the following option is set first in Quartus Prime (it is apparently not available in Quartus II):
     - Assignments
       --> Settings
       --> Compiler Settings
       --> Advanced Settings (Synthesis)
       --> Allow Any RAW Size for Recognition (Turn On)



text-align:left;

Table 1. Advanced Synthesis Settings (1 of 13)

Option	Description
Allow Any RAM Size for Recognition	Allows the Compiler to infer RAMs of any size, even if the RAMs do not meet the current minimum requirements.
Allow Any ROM Size for Recognition	Allows the Compiler to infer ROMs of any size even if the ROMs do not meet the design's current minimum size requirements.

https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/mapIdTopics/mwh1465495270874.htm

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)

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] addr_wr,
   input  [6:0] addr_rd
   );

   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[addr_wr] <= data_in;    // write mem
      end
      data_out <= mem[addr_rd];      // read mem
   end
endmodule
```

M10K Basic SRAM Template

- In the general case, it does not appear to be helpful to use any pragmas for M10K memory declarations
- This example instantiates a 24-bit × 1024-word memory which should be built from *three* M10K memory block arrays—because one M10K can have 8 or 10 bits per word when configured for 1024 words.

```
module basic_ram (
   input         clk,
   input         wr_en,
   input  [23:0] data_in,
   output [23:0] data_out,
   input  [9:0]  addr_wr,
   input  [9:0]  addr_rd
   );

   reg [23:0] mem [1023:0];

   // To initialize the RAM, Quartus supports initialization
   // which normal RAMs and synthesis do not support.
   // initial begin
   //   mem[0]    = 24'h03B03F;
   //   mem[1]    = 24'h000FFF;
   //   mem[2]    = 24'hBEBEBE;
   //   ...
   //   mem[1023] = 24'h726384;
   // end

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

M9K and M10K Verilog

- As stated in the Verilog Single-Bit Memories handout, there should normally not be any logic in register declarations:

  ```
  always @(posedge clock) begin
  ```

  This rule is even more important when coding block RAMs. In many cases the compiler will not use a block RAM (and use many LUTs instead) if this rule is violated.

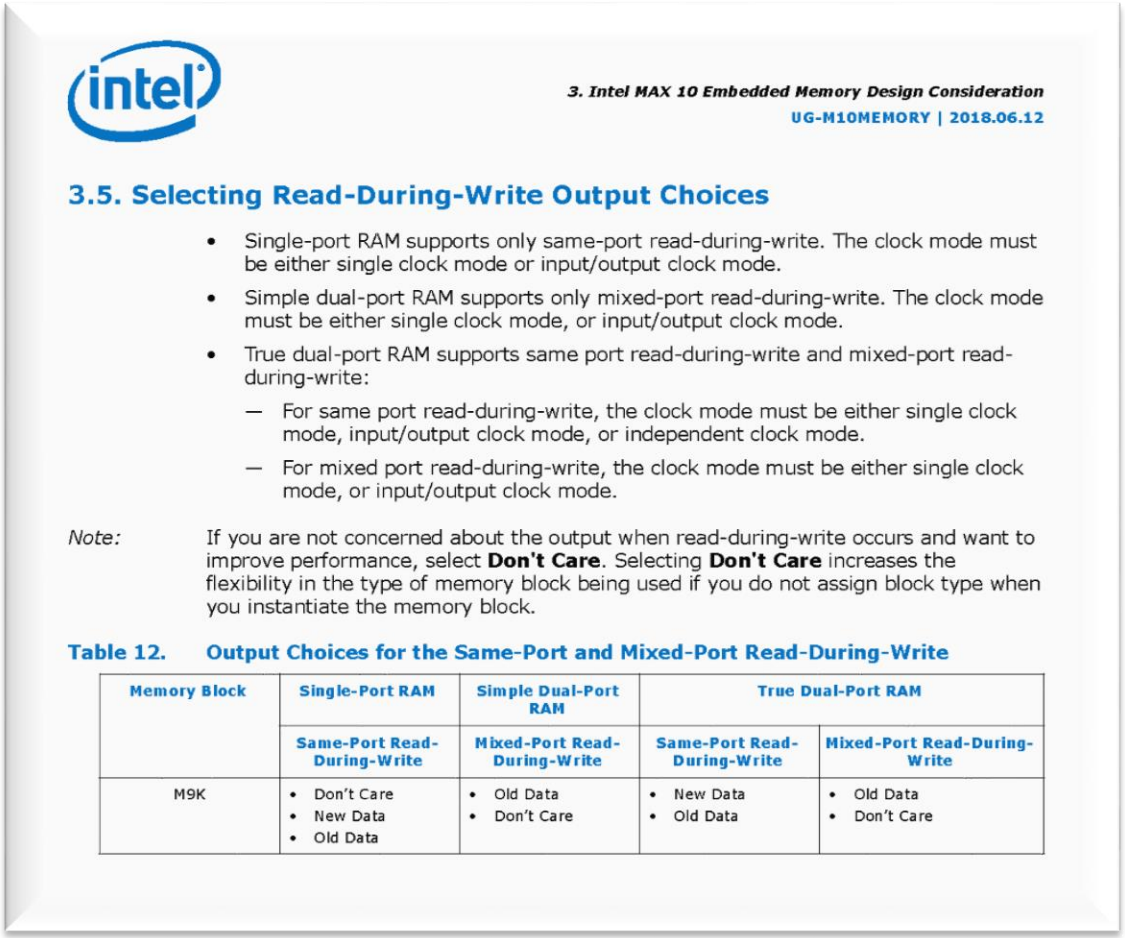
M9K Basic SRAM Template

- “In addition to specifying the type of memory block for the RAM implementation, by setting the value to "no_rw_check", you can use the ramstyle attribute to indicate that you do not care about the output of the inferred RAM when there are simultaneous reads and writes to the same address. By default, Quartus Prime tries to create an inferred RAM with the same read-during-write behavior as your HDL source. In some cases, a RAM must be mapped into logic because it has a read-during-write behavior that is not supported by the memory blocks in your target device. In other cases, Quartus must insert extra logic to mimic your read-during-write behavior, which can increase the resource requirements or reduce the performance of your design. Setting the "no_rw_check" value directs the Quartus Prime Compiler that the read-during-write behavior of the HDL source does not need to be preserved.
- You can specify both a block-type and "no_rw_check" in a single attribute by separating the values with a comma, for example "no_rw_check, M9K", or you can specify only a block-type or "no_rw_check".

https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/hdl/vlog/vlog_file_dir_ram.htm

M9K Simultaneous Write and Read Operations

- M9K block RAMs behave differently with various port configurations and various selectable simultaneous write/read characteristics
- Selectable "old" or "new" from a single port is likely implemented by the order the RAM performs the two operations



3. Intel MAX 10 Embedded Memory Design Consideration
UG-M10MEMORY | 2018.06.12

3.5. Selecting Read-During-Write Output Choices

- Single-port RAM supports only same-port read-during-write. The clock mode must be either single clock mode or input/output clock mode.
- Simple dual-port RAM supports only mixed-port read-during-write. The clock mode must be either single clock mode, or input/output clock mode.
- True dual-port RAM supports same port read-during-write and mixed-port read-during-write:
  - For same port read-during-write, the clock mode must be either single clock mode, input/output clock mode, or independent clock mode.
  - For mixed port read-during-write, the clock mode must be either single clock mode, or input/output clock mode.

Note: If you are not concerned about the output when read-during-write occurs and want to improve performance, select **Don't Care**. Selecting **Don't Care** increases the flexibility in the type of memory block being used if you do not assign block type when you instantiate the memory block.

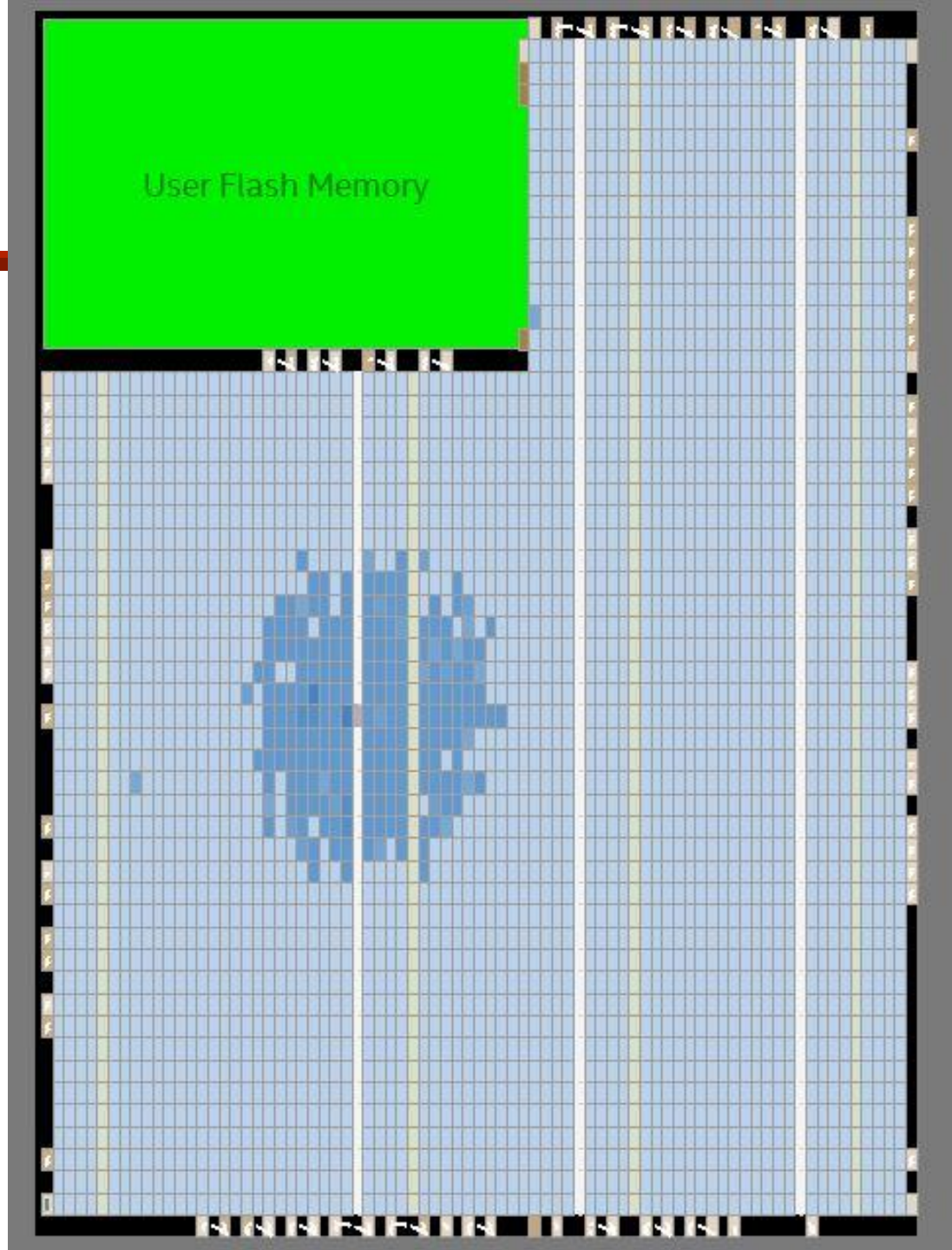
Table 12. Output Choices for the Same-Port and Mixed-Port Read-During-Write

Memory Block	Single-Port RAM	Simple Dual-Port RAM	True Dual-Port RAM	
	Same-Port Read-During-Write	Mixed-Port Read-During-Write	Same-Port Read-During-Write	Mixed-Port Read-During-Write
M9K	• Don't Care • New Data • Old Data	• Old Data • Don't Care	• New Data • Old Data	• Old Data • Don't Care

Intel MAX 10 Embedded Memory User Guide, page 20
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/ug_m10_memory.pdf

Example Design Utilizing LUT Memory

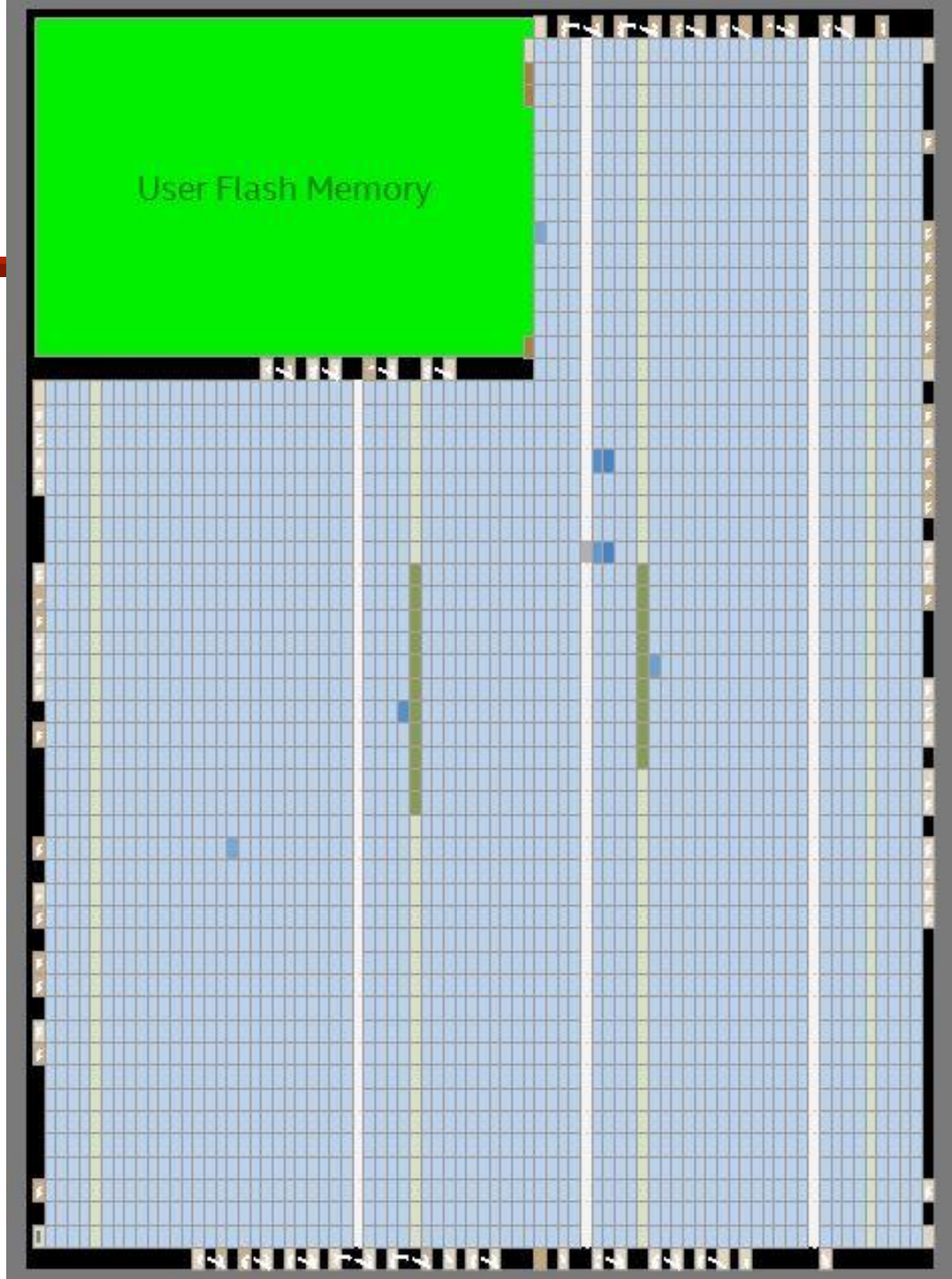
- In this example, the M9Ks are not enabled and the large ROM memories are implemented using individual Logic Elements



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