Scalable and Low Power LDPC Decoder Design Using High Level Algorithmic Synthesis

Yang Sun, and Joseph R. Cavallaro
Depart. of Electrical and Computer Engineering
Rice University
6100 Main, Houston, TX 77005
{ysun,cavallar}@rice.edu

Tai Ly
Synfora Inc.
201 San Antonio Circle Suite #172,
Mountain View, CA 94040
Tai.Ly@synfora.com

Abstract— This paper presents a scalable and low power low-density parity-check (LDPC) decoder design for the next generation wireless handset SoC. The methodology is based on high level synthesis: PICO (program-in chip-out) tool was used to produce efficient RTL directly from a sequential un-timed C algorithm. We propose two parallel LDPC decoder architectures: (1) per-layer decoding architecture with scalable parallelism, and (2) multi-layer pipelined decoding architecture to achieve higher throughput. Based on the PICO technology, we have implemented a two-layer pipelined decoder on a TSMC 65nm 0.9V 8-metal layer CMOS technology with a core area of 1.2 mm^2. The maximum achievable throughput is 415 Mbps when operating at 400 MHz clock frequency and the estimated peak power consumption is 180 mW.

I. INTRODUCTION

Wireless communication is experiencing rapid growth to provide ubiquitous mobile wireless connectivity. The continuously changing and evolving wireless specifications pose a major challenge to designers to implement these highly complex algorithms in hardware as rapidly as possible, while still maintaining area and power efficiency. As a very competitive coding scheme for next generation wireless systems, low-density parity-check (LDPC) codes [1] are being considered for many 4G systems because of their excellent error correction performance and highly parallel decoding scheme. To meet the data rate and power consumption constraints in wireless handsets, it is very challenging to design a high performance LDPC decoder at low area cost with reduced development time.

Because different standards employ different LDPC codes, it is very important to design a flexible LDPC decoder that can be tailored to different applications. To address this challenge, some efficient ASIC architectures have been proposed in the literature [2], [3], [4], [5], [6] to achieve flexible decoding of several LDPC codes. However, one limitation of these partial-parallel architectures is that the level of parallelism has to be at the sub-circulant level. In this paper, we will explore the design space of parallel realizations of LDPC decoders using a high level synthesis (HLS) methodology. Under the guidance of the designers, HLS can effectively exploit the parallelism of a given algorithm. In this work, we propose two parallel LDPC decoding algorithms that can be exploited by the HLS to produce area and power efficient hardware.

II. PICO PLATFORM

The high level synthesis used in this paper is the PICO Algorithmic Synthesis, which creates application accelerators from un-timed C for complex processing hardware in video, audio, imaging, wireless and encryption domains [7]. Figure 1 shows the overall design flow for creating application accelerators using PICO. The user provides a C description of their algorithm along with performance requirements and functional test inputs. The PICO system automatically generates the synthesizable RTL, customized test benches, SystemC models at various levels of accuracy as well as synthesis and simulation scripts. PICO is based on an advanced parallelizing compiler that finds and exploits parallelism at all levels in the C code. PICO provides multi-level hierarchical design capability for complex designs such as LDPC decoders and block-level clock gating to minimize power at the architecture level. The quality of the generated RTL is competitive with manual design, and the RTL is guaranteed to be functionally equivalent to the algorithmic C input description [7]. The generated RTL can then be taken through standard simulation, synthesis, place and route tools and integrated into the SoC through automatically configured scripts.

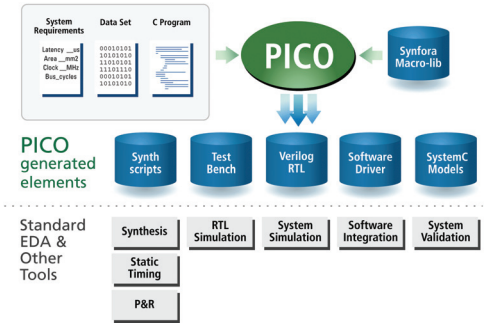

Fig. 1. PICO algorithmic synthesis

978-1-4244-4941-5/09/$25.00 ©2009 IEEE

III. DECODING ALGORITHM

A binary LDPC code is a linear block code specified by a very sparse binary $M \times N$ parity check matrix: $\mathbf{H} \cdot \mathbf{x}^T = 0$, where $\mathbf{x}$ is a codeword and $\mathbf{H}$ can be viewed as a bipartite graph where each column and row in $\mathbf{H}$ represent a variable node and a check node, respectively. In this paper, we will focus on a special class of LDPC codes called block-structured LDPC codes which are adopted by many new wireless standards. As shown in Figure 2, a block structured parity check matrix can be viewed as a two-dimensional array of square sub-matrices. Each sub-matrix is either a zero matrix or a z by z weight-1 circulant matrix. The size of the $\mathbf{H}$ matrix differs from standard to standard: e.g. the maximum block length is 2304 for IEEE 802.16e and 1944 for IEEE 802.11n. To describe the decoding algorithm, we define $\mathcal{N}_m$ as the set of variable nodes connected to check node m, and $\mathcal{N}_m \backslash n$ as the set $\mathcal{N}_m$ with variable node n excluded. R_{mn} and Q_{mn} are check and variable messages, respectively. P_n is the *a posteriori* probability log-likelihood ratio of variable node n. $\mathbf{H}^l$ is the l-th layer of $\mathbf{H}$. y_n is the received channel data. σ^2 is the noise variance. The layered scaled-minsum decoding algorithm is then summarized in Algorithm 1.

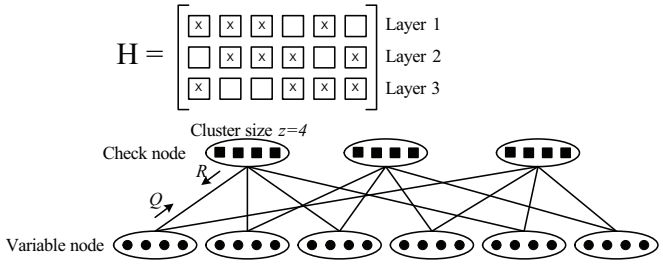

Fig. 2. An example of 3 × 6 block structured parity check matrix.

IV. ARCHITECTURE DESIGN

A. Per-layer decoding architecture

In Algorithm 1, one full iteration is divided into L sub-iterations where each sub-iteration corresponds to one layer's data processing. In the traditional partial-parallel decoder architecture, each $z \times z$ sub-matrix is treated as a block within which all the involved parity checks are processed in parallel using z number of decoding cores. Each core is independent from all others since there is no data dependence between adjacent check rows. The parallelism is only at the sub-circulant level because it is easier to treat each circulant sub-matrix as a whole processing block. However, different systems demand different levels of parallelism. To the best of our knowledge, the VLSI design of LDPC decoders with scalable parallelism is largely missing in the literature. Because the parity check matrices defined in different wireless standards can be very different, this poses a challenge for hand-coded RTL designers.

Algorithm 1 Layered scaled-minsum algorithm

Initialization:
$\forall(m,n)$ with $\mathbf{H}(m,n)=1$, set $R_{mn}=0$, $P_n=\frac{2y_n}{\sigma^2}$
Iteration:
for iteration $i=1$ to I **do**
  for layer $l=1$ to L **do**
    Stage 1) Read and pre-process:
    $\forall(m,n)$ with $\mathbf{H}^l(m,n)=1$:
      Read P_n and R_{mn} from memory
      Calculate $Q_{mn}=P_n-R_{mn}$
    Stage 2) Decode and write back:
    $\forall(m,n)$ with $\mathbf{H}^l(m,n)=1$:
      $R^{'}_{mn}=0.75\times\prod_{j\in\mathcal{N}_m\backslash n}\mathrm{sign}(Q_{mj})\times\left(\min_{j\in\mathcal{N}_m\backslash n}|Q_{mj}|\right)$
      $P^{'}_{n}=Q_{mn}+R^{'}_{mn}$
      Write $P^{'}_{n}$ and $R^{'}_{mn}$ back to memory
  end for
end for
Decision making: $\hat{x_n}=\mathrm{sign}(P_n)$

In this work, we leverage PICO high level synthesis to automatically create scalable decoder architectures. Take WiMax LDPC decoder as an example, where z is between 24 and 96. The structure in Figure 3(a) corresponds to the traditional WiMax LDPC decoder architecture where $z=96$ cores are generated to provide the maximum parallelism. The "#pragma unroll" directive in the C program is used to tell the PICO compiler to unroll this loop. Figure 3(b) shows a way to partially unroll the loop: the inner loop is unrolled by 48, with an outer loop of 2 iterations that pipeline the data over the 48 cores, sequentially, for 2 times. In this way, multiple parallelism levels can be realized to tailor the throughput according to the application requirement.

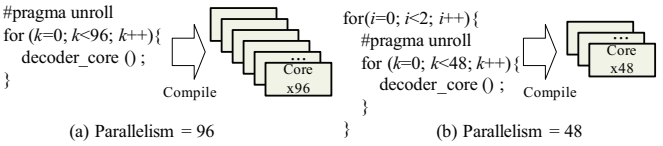

Fig. 3. Scalable data path generation by PICO

To implement Algorithm 1 in hardware, we propose a per-layer two-stage scheduling algorithm which is illustrated in Figure 4 (without loss of generality, we assume the parallelism level is z). It is a block-serial scheduling algorithm: data in each layer is processed block-column by block-column. Core 1 first reads a vector (size of 1-by-z) of P_n and R_{mn} messages from memory, calculates Q_{mn}, and then finds the minimum and the second minimum values among array Q_{mn} for each row m over all column n. Core 2 computes the new $R^{'}_{mn}$ and $P^{'}_{n}$ values based on the two minimum values produced by core 1, and writes the new $R^{'}_{mn}$ and $P^{'}_{n}$ values back to memory. Q_{mn} values, calculated in core 1, are stored in an array

so that they can be re-used by core 2.

As a case study, we describe a length 2304 rate 1/2 WiMax LDPC decoder in a sequential un-timed C code. Figure 5 shows the corresponding PICO generated hardware architecture block diagram. The top level LDPC_decoder() will loop over I iterations. In each iteration, it loops over L layers of the parity check matrix, and calls decoder_core1() and decoder_core2(). The for loops in decoder_core1() and decoder_core2() are unrolled. The for loop in barrel_shifter() is also unrolled to shift P_n messages as an array of 8-bit, left by n from 0 to 95. The top level function can return early if all the parity checks are satisfied or the maximum number of iterations is reached. Both P and R messages are represented with 8-bit fixed-point numbers. All arrays shown in the block diagram are declared as global C arrays which are synthesized as register files implemented in flip-flops. Large memories such as P and R memory are treated as user-supplied macros (SRAMs).

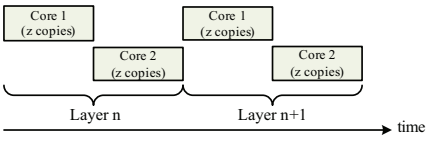

Fig. 4. Per-layer decoding algorithm

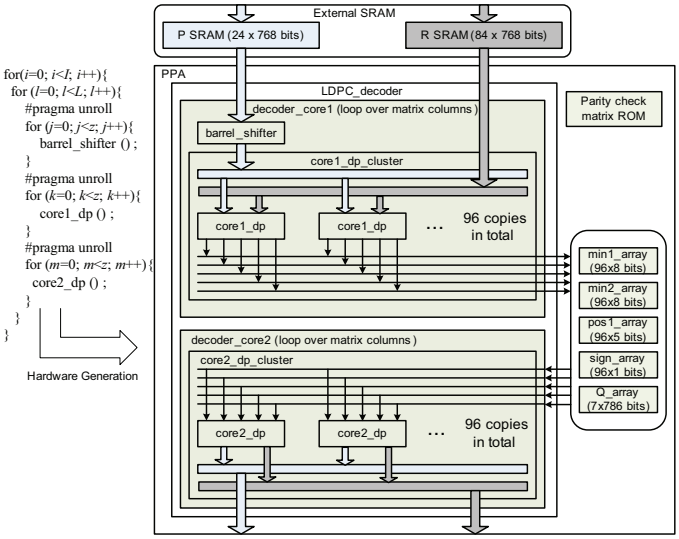

Fig. 5. PICO hardware architecture block diagram for per-layer decoding of a (2304, 1/2) WiMax LDPC code

B. Multi-layer pipelined decoder architecture

In the per-layer architecture, the core utilization is low (about 50% as shown in Figure 4). However, it is possible to do pipelined processing between layers so that while core 2 is operating on the current layer, core 1 could start working on the next layer of the matrix as shown in Figure 6. In order to pipeline core 1 and core 2, additional conflict detection logic is needed to insert stall cycles to avoid pipeline hazards where core 1 reads P_n messages from a location before core 2 has written to that location. To do this, we add a "scoreboard" variable, which is set to contain a 1 in bit n if and only if a write to P_n is pending by core 2 for row m. As core 2 writes to each P_n, it will clear bit n of the scoreboard. As core 1 processes a column n, it will check if scoreboard has a 1 in bit n. If it does, then core 1 will do nothing for that iteration, thus waiting for core 2 to write to P_n before reading from P_n. Figure 7 shows a two-layer pipelined PICO hardware architecture block diagram. This architecture is similar to the per-layer architecture except that each core now has its own copies of arrays, and Q array has been replaced by a Q FIFO.

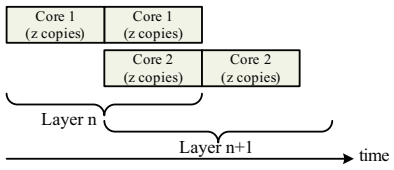

Fig. 6. Multi-layer pipelined decoding algorithm

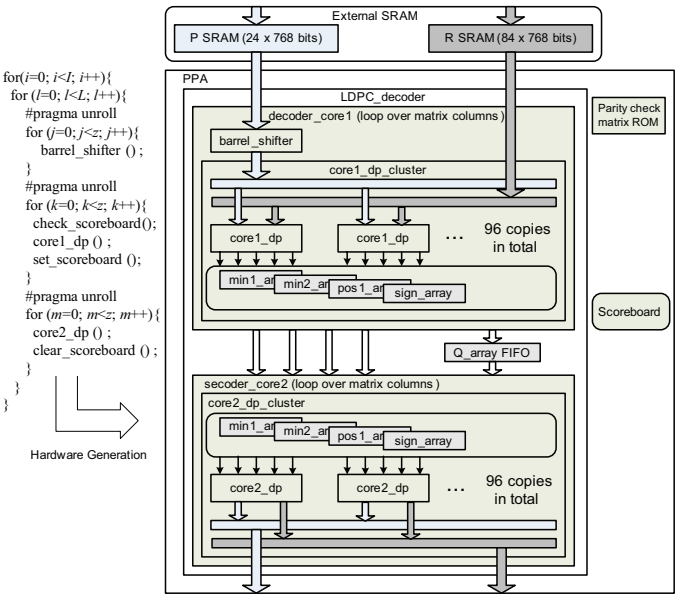

Fig. 7. PICO hardware architecture block diagram for two-layer pipelined decoding of a (2304, 1/2) WiMax LDPC code

Figure 8 compares the latency and area of these two architectures. In the analysis, RTLs are generated by PICO and are synthesized using Synopsys Design Compiler on a TSMC 65nm technology. Note that the area value shown in Figure 8 is for the total standard cells. This will give a fair comparison because two architectures would require the same amount of external SRAMs. In Figure 8, we can see both latency and area increase as clock frequency increases. This is expected because PICO will adjust the design and find the best solution for a given target clock frequency.

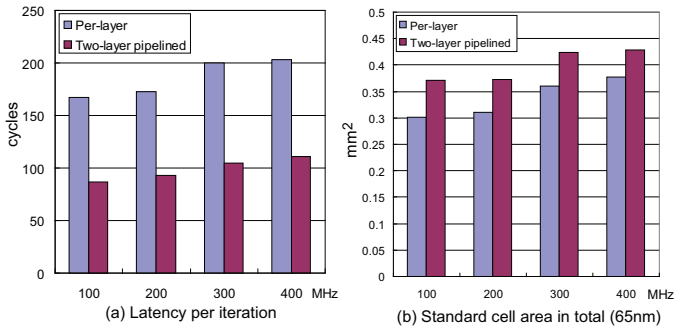

Fig. 8. Latency and area (65nm) comparisons of two PICO hardware architectures synthesized for different target clock frequency goals

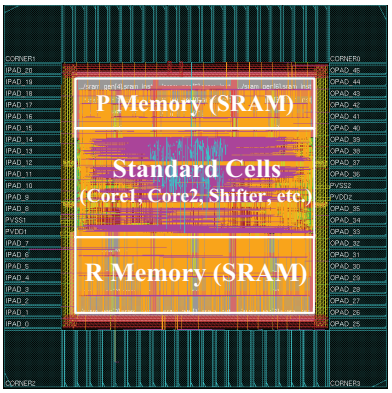

Fig. 9. VLSI layout view of the LDPC decoder

C. Low power implementation via clock-gating

PICO scheduler can analyze the underlying data flow graph, and set those idle registers' "enable" signals to "0" when the module has no activity. PICO also provides block-level clock gating which shuts off entire processing blocks to minimize power at an architectural level. Table I compares the power consumption of a (2304, 1/2) pipelined LDPC decoder with and without clock-gating. SpyGlass [8] was used to conduct the gate-level power estimation (not including external SRAMs). From Table I, we can see a 29% reduction of the "sequential internal power" via clock-gating.

TABLE I

SPYGLASS POWER ESTIMATES WITH AND WITHOUT CLOCK GATING

Power	Leakage	Internal	Switching	Total
W/ clock-gating	3.43mW	46.1mW	22.5mW	72.0mW
W/O clock-gating	3.43mW	64.5mW	22.5mW	90.4mW

V. IMPLEMENTATION RESULTS AND COMPARISONS

A flexible LDPC decoder which fully supports the IEEE 802.16e WiMax standard has been described in "PICO-rized" C. Then the PICO tool was used to produce Verilog RTL for the proposed two-layer pipelined LDPC decoder architecture. The generated RTL was synthesized using Synopsys Design Compiler, and placed & routed using Cadence SoC Encounter on a TSMC 65nm 0.9V 8-metal layer CMOS technology. The VLSI layout view of this decoder with a core area of 1.2 mm^2 (standard cells + SRAMs) is shown in Fig. 9. Table II compares our decoder with the state-of-the-art LDPC decoders of [2] and [3]. A fair comparison is difficult to make because of different design parameters. However, it can be roughly inferred that the PICO-generated decoder can achieve comparable performance with the hand designed decoders in terms of throughput, area, and power.

TABLE II

COMPARISON WITH EXISTING LDPC DECODERS

	This Work	[2]	[3]
Core Area	1.2 mm^2	0.74 mm^2	1.337 mm^2
Max Frequency	400 MHz	240 MHz	400 MHz
Max Power	180 mW	235 mW	NA
Technology	65 nm	65 nm	65 nm
Quantization	6	5	6
Number of Iterations	10	13	25-20
Max Code Length	2304	1944	2304
Memory (SRAM)	82,944 bit	68,256 bit	0.551 mm^2
Max Throughput @ R=1/2	415 Mbps	178 Mbps	333 Mbps
Max Latency @ R=1/2	2.8 μs	5.75 μs	6.0 μs

VI. CONCLUSION

A high performance low power LDPC decoder has been implemented based on PICO high level synthesis flow. Two parallel decoder architectures are introduced and compared. The decoder, which was implemented on a 65nm CMOS technology, delivers comparable results to the hand-coded designs while significantly reducing the development time.

VII. Acknowledgement

The first and second authors at Rice University were supported in part by Nokia, NSN, Xilinx, and NSF under grants CCF-0541363, CNS-0551692, and CNS-0619767.

References

[1] R. Gallager, "Low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 8, pp. 21–28, Jan. 1962.
[2] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, "A Scalable Decoder Architecture for IEEE 802.11n LDPC Codes," in *GLOBECOM*, 2007, pp. 3270–3274.
[3] T. Brack, et al., "Low complexity ldpc code decoders for next generation standards." in *Design, Automation, and Test in Europe*. ACM, 2007, pp. 331–336.
[4] Y. Sun, M. Karkooti, and J. R. Cavallaro, "VLSI Decoder Architecture for High Throughput, Variable Block-size and Multi-rate LDPC Codes," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May. 2007, pp. 2104–2107.
[5] Y. Sun and J. R. Cavallaro, "A low-power 1-Gbps reconfigurable LDPC decoder design for multiple 4G wireless standards," in *IEEE International SOC Conference (SoCC)*, Sept. 2008, pp. 367–370.
[6] K. Gunnam, G. S. Choi, M. B. Yeary, and M. Atiquzzaman, "VLSI Architectures for Layered Decoding for Irregular LDPC Codes of WiMax," in *Int. Conf. Commun. (ICC)*, June 2007.
[7] S. Aditya and V. Kathail, *Algorithmic Synthesis Using PICO*. Springer Netherlands, 2008, pp. 53–74.
[8] *SpyGlass datasheet*, http://www.atrenta.com.