# A variable-rate Viterbi decoder in 130-nm CMOS: design, measurements, and cost of flexibility

Kamuf, Matthias; Öwall, Viktor; Rodrigues, Joachim; Anderson, John B

*Published in:*
Proceedings, Norchip Conference

2008

# A Variable-Rate Viterbi Decoder in 130-nm CMOS

Matthias Kamuf, Viktor Öwall, Joachim Neves Rodrigues, and John B. Anderson
Department of Electrical and Information Technology, Lund University, SE-221 00 Lund, Sweden

***Abstract*— This paper discusses design and measurements of a flexible Viterbi decoder fabricated in 130-nm digital CMOS. Flexibility was incorporated by providing various code rates and modulation schemes to adjust to varying channel conditions. Based on previous trade-off studies, flexible building blocks were carefully designed to cause as little area penalty as possible. The chip runs down to a minimal core supply of 0.8 V. It turns out that striving for more modulation schemes is beneficial in terms of power consumption once the price is paid for accepting different code rates viz. radices in the trellis and survivor path units.**

## I. Introduction

Varying channel conditions in a mobile environment require variable-rate transmission, i.e., code rate and modulation scheme have to be adjustable. Consider high-rate wireless personal area networks (WPANs) [1], which provide short-range ad-hoc connectivity for mobile communication devices. A flexible channel decoding platform in such an environment is required to provide at least two decoding modes, one when good error-correcting capability is needed at low SNR, and one supporting high data throughput if the channel is good.

The Viterbi algorithm (VA) serves as demonstration vehicle, and the transmission uses binary convolutional as well as trellis-coded modulation (TCM) codes, which in [1] enables transmitting information at high rates per Hertz of bandwidth. The subset selectors of the TCM codes have 8 states (i.e., $m = 3$ memory elements) and are rate $R_c = 1/2$ for QPSK and rate $R_c = 2/3$ for 16-QAM and 64-QAM. The two code rates imply butterflies of different radices and puncturing is not applicable if code performance is to be fully maintained. Considerations in [2] suggest that a flexible channel decoding platform has to efficiently process both radix-2 (R2) and radix-4 (R4) butterflies. Furthermore, both decoding modes use the same computational kernel to limit overhead in area.

The resulting architecture was presented in [2], whereas this paper focuses on silicon implementation, measurements and comparisons to other flexible trellis decoders. An important question is addressed when one wants to incorporate flexibility, namely, what is its overall hardware and performance cost. We start on an application level and evaluate competing design alternatives to provide flexibility in a trellis decoder. Furthermore, on an architecture level, an R2/R4 trellis kernel and a cycle-matched, folded survivor path (SP) unit enable the efficient incorporation of the required flexibility. Finally, a real measure of flexibility is extracted by silicon implementation.

## II. Classification of Flexible Trellis Decoders

Other flexible trellis decoders are studied to point out where in the application and performance space our design is located. These attempts are divided into the following categories: the first two ($m$- and algorithm-flexible) are expected to operate in the low energy region and therefore employ small constellations such as BPSK or QPSK. Coding is based on rate $1/c$ convolutional codes, including punctured or concatenated versions thereof. The last category (bandwidth-flexible) inherently supports larger constellations and different coding schemes, thus facing other design challenges.

### A. $m$-flexible Solutions

These approaches use one decoding algorithm and provide flexible error correction by varying encoder memory $m$. For example, Chadha [3] and Hocevar [4] designed flexible VA-based architectures.

Chadha's implementation provides a fully parallel solution ($m_{\max} = 6$) and shuts down unnecessary parts when processing trellises with fewer states. The extra hardware spent in the flexible designs is compared to a fixed design with the same $m_{\max}$. This overhead is at most 2.9% since it mainly accounts for shut-down logic and routing resources. However, no evaluation is given of the provided flexibility compared to fixed designs with $m < m_{\max}$. In this case, an increasing relative overhead should be encountered as $m$ decreases. Supported code rates are $1/2$ and $1/3$. Code rate is not a critical design parameter when used with antipodal constellations such as BPSK and QPSK since the calculation of distances to the $2^c$ code sequences is very simple. However, the design does neither provide puncturing resources nor a demapper for higher constellations to enable high-rate transmission.

Hocevar's design is a DSP coprocessor, which supports a variety of code rates below 1 that are achieved by puncturing the basic $1/2$, $1/3$, and $1/4$ convolutional codes. Although being more flexible than a tailored design such as Chadha's, the price for such flexibility is paid by severe throughput degradation [5].

### B. Algorithm-flexible Solutions

An straightforward combination is the VA together with the (max-)log-MAP algorithm since they share the main processing engine, the add-compare-select (ACS) operation. For example, Bickerstaff *et al.* [6] provide 256-state Viterbi and 8-state log-MAP decoding. The trellis block processes 8 states in parallel, i.e., the Viterbi decoding is carried out time-multiplexed. Since this design is to be (commercially) used in third generation mobile services, it includes several communication interfaces and an evaluation of the cost of flexibility was not of main interest.

In Cavalloro's work [7] the combination of VA with an augmented soft-output unit is investigated. Encoder memory is variable up to $m_{\max} = 8$. The approach is based on the

1-4244-2493-1/08/$20.00 ©2008 IEEE

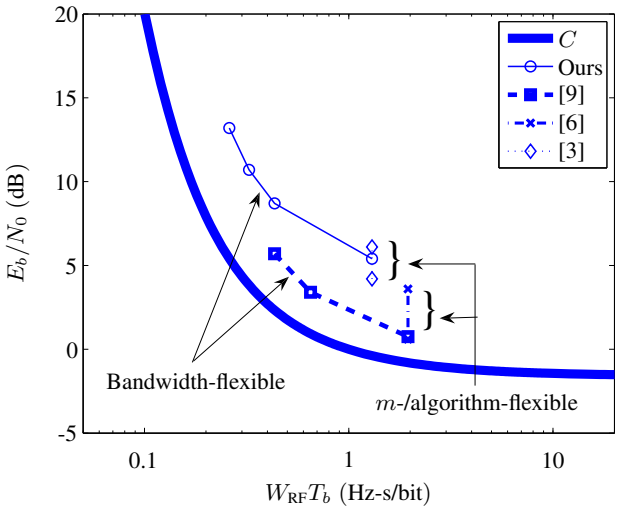


Fig. 1. Energy–bandwidth performance of some implemented flexible trellis decoders for BER of $10^{-5}$. Shannon AWGN capacity $C$ is drawn bold for comparison. Designs become more complex to implement the closer to $C$ they are, i.e., [9] is more complex than ours. This is reflected in the line thickness.

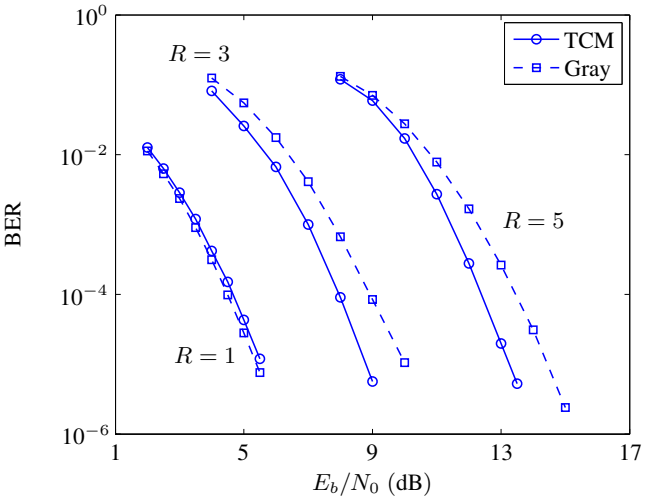


Fig. 2. Performance comparison of rate-$R$ transmission schemes using TCM or convolutional coding with Gray-mapped constellations.

work of Chadha [3], i.e., the contribution of flexibility is due to shut-down logic and routing resources.

### C. Bandwidth-flexible Solutions

A Viterbi processor for TCM codes is presented in [8]. It is widely programmable and executes the main decoding operations sequentially, thus limiting the achievable throughput as in the case of [4].

Miyauchi *et al.* [9] describe a fully integrated dedicated soft-input soft-output processor focused on iterative decoding. It has many features such as arbitrary coding polynomials, interleaving patterns, and constellation configurations. Different classes of coding approaches are supported, parallel/serial concatenated convolutional codes, turbo TCM, and serial concatenated TCM. The highest constellation considered is 8-PSK. Design challenges in this approach are mainly concerned with the incorporation of R4-processing to log-MAP decoding.

Our flexible Viterbi decoder [2] also belongs to this class providing the advantage of a wider range of transmission rates to adapt to low- and high-energy scenarios.

### D. Performance Evaluation

The different approaches are compared in an energy–bandwidth sense. Note that energy here is the required received energy per bit $E_b$ at the input to the decoder to achieve a certain bit error rate (BER), not the energy consumed by these implementations. Fig. 1 shows some implemented flexible trellis decoders and their energy–bandwidth performance to achieve a BER of $10^{-5}$ in the AWGN channel. It is assumed that an AWGN channel use with two independent dimensions occurs every symbol time $T_s = T_b \cdot R$, where $T_b$ is time per data bit and the number of data bits per channel use (transmission rate) is

$$R = \begin{cases} \log_2 \mathcal{M} \cdot R_c & \text{for convolutional codes} \\ \log_2 \mathcal{M} - 1 & \text{for TCM codes,} \end{cases}$$

where $\mathcal{M}$ is the number of constellation symbols. RF bandwidth $W_{\mathrm{RF}} = 1.3/T_s$ is normalized to $T_b$ and includes excess bandwidth introduced by 30% root-raised-cosine pulses. The Shannon AWGN capacity $C$ is shown for comparison.

From Fig. 1, note that designs solely based on antipodal modulations ($m$- and algorithm-flexible) [3], [6] provide no increased throughput as the channel SNR improves. They just trade required energy for bandwidth, which is indicated by vertical lines. One solution is to employ higher constellations together with (punctured) convolutional codes to gradually increase data rates. However, compared to TCM, which was intended for higher constellations, these systems are not as energy-efficient for the same BER, throughput, and complexity. This is shown in Fig. 2, where $R = 3$ and $R = 5$ systems are simulated. The transmission rates for the Gray-mapped constellations are derived by a rate $1/2$ code punctured to $R_c = 3/4$ (16-QAM) and $R_c = 5/6$ (64-QAM). For TCM, the code rate of the subset selector is $2/3$ for both 16-QAM and 64-QAM. It is seen that for $R > 1$, the setups with the Gray-mapped constellation require about 1–1.3 dB more $E_b/N_0$ at the target BER of $10^{-5}$.

It is noteworthy that systems become more complex to implement the closer to capacity they get. Consider Miyauchi's design [9], which apparently provides a good energy–bandwidth trade-off with help of iterative decoding. However, that iterative decoding schemes run multiple times over a trellis, increasing latency and raw computational cost per decoded bit. That is, for low-power low-cost applications as in WPAN, such a solution is over-designed. The flexible Viterbi decoder described in our work is a lower-complexity solution that adjusts to varying channel conditions by providing several transmission rates using different constellations.

Programmable trellis processors, such as [4], [8], provide highest flexibility. In Fig. 1, these systems would realize design points that are bound by a sphere. However, this flexibility degrades processing speed and power consumption by several orders of magnitude compared to more dedicated solutions [5].

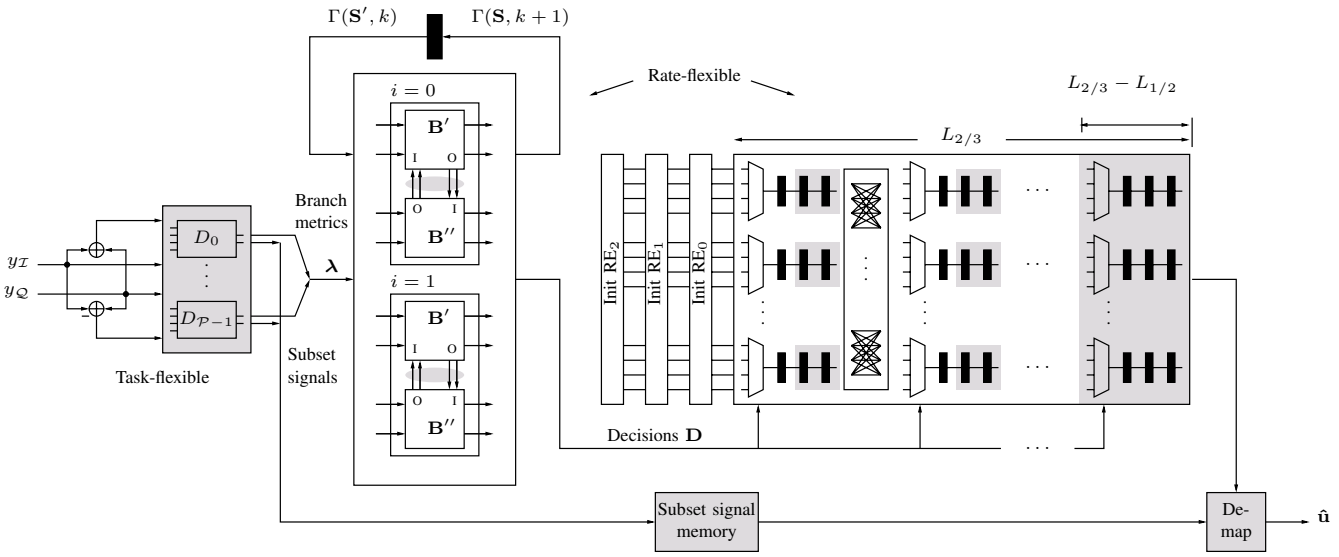


Fig. 3. Architecture of the designed flexible Viterbi decoder. Additonal parts needed for decoding of TCM codes are shaded in gray. In the BM unit (on the left), the increased complexity is mainly due to the use of higher constellations in TCM (task-flexible). For trellis and SP unit (center and right part, respectively), rate-flexibility is the source of the added complexity.

## III. The Flexible Architecture

This section briefly presents the architecture of our flexible Viterbi decoder (Fig. 3). An in-depth coverage of the design considerations for the different building blocks is found in [2].

### A. Branch Metric Unit

The branch metric (BM) unit shown on the left in Fig. 3 provides measures of likelihood $\boldsymbol{\lambda}$ for transitions in a trellis stage. In an AWGN channel, the optimal distance measure is the squared Euclidean distance between received channel symbol $\mathbf{y} = (y_{\mathcal{I}}, y_{\mathcal{Q}})$ and constellation symbol $\mathbf{s}$, i.e., $|\mathbf{y}-\mathbf{s}|^2$. For antipodal signaling this expression is reformulated to additions and subtractions of the channel symbols. Channel symbols are quantized with $q = 3$ bits without causing much performance degradation [2]. Higher order modulations, however, need to employ the squaring operation if optimality is to be maintained. A low-complexity workaround is to use an absolute distance measure, i.e., $|\mathbf{y} - \mathbf{s}|$. Besides, more bits per channel symbol are needed to minimize BER performance degradation. It was found in [2] that $q = 5$ and $q = 7$ bits for 16-QAM and 64-QAM, respectively, give negligible degradation (ca. 0.1 dB) compared to the performance using unquantized Euclidean distance.

Ontop of the distance calculations, TCM codes require an additional subset decoder $D$. The calculations needed for subset decoding of the $\mathcal{P}$ subsets, $y_{\mathcal{Q}} - y_{\mathcal{I}}$ and $y_{\mathcal{Q}} + y_{\mathcal{I}}$, can be reused in case of rate $1/2$ convolutional coding. These results are equivalent to the BMs for code symbols $\{+1 - 1\}$ and $\{-1 - 1\}$, respectively. The remaining two metrics are derived from these by negation. A single subset decoder $D$ consists of comparators, a demapper, and the actual distance calculation. The complexity in such a subset decoder stems from the use of higher order modulations, resulting in more slicing operations to find the most likely subset signal. This signal has to be stored for all subsets in a subset signal memory. Together with the reconstructed subset sequence from the SP unit, the final decoded data sequence $\hat{\mathbf{u}}$ can be established.

### B. Trellis Unit

The architecture of a trellis unit (center part in Fig. 3) depends on the code rate $R_c$ and number of states $N$. The ACS operations in this unit discard unlikely branches in a trellis diagram and produce decision symbols $\mathbf{D}$ about the surviving branches. Since trellis diagrams of $1/2$ and $2/3$ encoders consist of R2 and R4 butterflies, i.e., we need a rate-flexible trellis unit.

Whereas R2 processing is done in one clock cycle, R4 processing is time-multiplexed in the rate-flexible trellis unit. All partial survivors are calculated during two cycles, and in the third cycle the final update takes place. The partial survivors needed for the final compare-select (CS) are calculated in different butterfly units and have to be stored temporarily. Appropriate routing for the final CS is according to the required ordering of the updated SMs $\Gamma(\mathbf{S}, k+1)$, where $\mathbf{S}$ is the set of states in the trellis diagram. Here, the partial survivors are brought together by means of I/O channels between adjacent butterfly units that belong to the same pair $i$ (here, both code trellises have 8 states and thus there are two such pairs $i = 0, 1$ processing 4 states each).

The arithmetic components in the rate-flexible butterfly unit are identical to the ones in a conventional R2 butterfly unit. To cope with a decomposed (time-multiplexed) R4 butterfly, routing resources are provided to distribute the partial survivors as dictated by the BM distribution (reflected in $\mathbf{B}'$ and $\mathbf{B}''$) and the state transitions. In total the rate-flexible butterfly unit only adds six $2:1$ multiplexers (MUXes) and two registers on top of an R2 butterfly unit, and there is no arithmetic overhead.

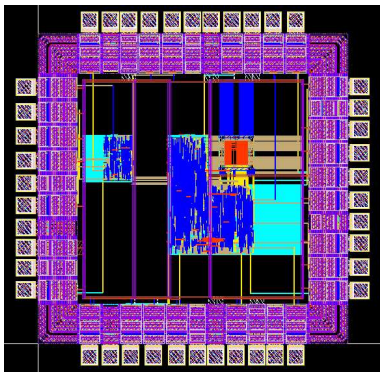

Fig. 4. Layout of the routed chip. Designs ONE and FIVE are shown on the left and right side, respectively. Row utilization is 80% in both implementations.

### C. Survivor Path Unit

The survivor bits from the trellis unit are processed by the SP unit (right part of Fig. 3) to reconstruct the transmitted data bits. The register-exchange (RE) approach is chosen since the number of states is low. Since an additional subset signal memory is needed for TCM, the least overhead is introduced since the decoding latency is the lowest compared to trace-back architectures. Additionally, for TCM a demapper needs to be employed that delivers the most likely subset signal at a certain time. This is a MUX which chooses a subset signal depending on the decoded subset number from the SP unit. For convolutional decoding, $b$ information bits are decoded every cycle. In case of TCM, however, $b+1$ must be decoded per trellis stage since the subset number consists of $b+1$ bits. Hence, the RE algorithm must store in total $(b+1)NL$ bits, where $L$ is the decoding depth. Note that rate $1/2$ and $2/3$ codes require different $L$, hence the distinction in Fig. 3, where the gray parts can be disabled during rate $1/2$ processing.

The complexity of the RE network is lowered by matching it to the throughput of the trellis unit. Recall that R4 processing takes 3 clock cycles and thus the RE update can also be carried out sequentially; that is, the registers are placed in series such that three cycles are needed to update the complete survivor sequence. The hardware requirement is drastically lowered compared to a straightforward parallel approach since 66% of the MUXes and interconnections become obsolete, and utilization for both modes is effectively increased.

## IV. Implementation and Measurements

The chip was modeled in VHDL at register-transfer level (RTL) and taken through a flow using Synopsys Design Compiler for synthesis and Cadence Encounter for routing. A high-speed standard cell library from Faraday is used for an 8 metal layer, 130-nm digital CMOS process from UMC. The RTL and gate level netlists are verified against test vectors generated from a MATLAB fixed-point model. Post-layout timing is verified using Synopsys Prime Time with net and cell delays back-annotated in standard delay format.

Fig. 4 shows the layout of the routed chip. It consists of two designs. The design on the left is called ONE, which provides a transmission rate of $R = 1$ using QPSK and an 8-state rate $1/2$ systematic convolutional code. This is a fixed design tailored for one purpose and serves as reference to which the flexible design is compared to. The flexible design, named FIVE, additionally provides $R = 3, 5$ transmission rates using TCM with 16-QAM and 64-QAM modulation schemes. Compared to a fixed design of $R = 5$, there is no hardware overhead, i.e., a higher rate design always includes the lower rate ones. This is due to the sharing of the BM calculations.

TABLE I

Fabricated designs and their modulation schemes. Power consumption in mW was extrapolated from $f_{\mathrm{clk}} = 160\,\mathrm{MHz}$ measurement in order to compare to an earlier power estimation in parentheses at $V_{\mathrm{dd}} = 1.2\,\mathrm{V}$ and $f_{\mathrm{clk}} = 250\,\mathrm{MHz}$.

| | ONE | FIVE |
|---|---|---|
| QPSK, $R = 1$ | 5.78 (4.9) | 13.35 (10.3) |
| 16-QAM, $R = 3$ | — | 19.0 (14.7) |
| 64-QAM, $R = 5$ | — | 19.71 (15.2) |

The chip is pad-limited due to test purposes and measures 1.44 mm$^2$. Designs ONE and FIVE are placed on the same die with separate $V_{\mathrm{dd}}$ to measure their power consumption independently. The cell area of ONE is 4.1 kGates (NAND2-equivalent), and for FIVE it is 19.2 kGates (15.7 kGates logic, 3.5 kGates memory). These numbers apply to synthesized blocks at gate level. Design constraints are chosen such that the implementation is in the flat part of the area–delay curve. The critical path for the designs lies in the trellis unit (1.65 ns and 1.98 ns, respectively).

In TCM mode, design FIVE achieves a symbol rate of 168 Mbaud/s, a throughput of 504 Mbit/s and 840 Mbit/s using $R = 3$ and $R = 5$ configurations. Design ONE achieves a throughput of 606 Mbit/s; flexibility causes a speed penalty in that FIVE provides 504 Mbit/s in $R = 1$ mode. If WPANs are the application, all these throughputs are higher than specified in [1]. Thus, the supply voltage can be lowered to save energy. Note that the throughput numbers apply to the circuit level. When considering data rates, one also has to keep in mind the underlying modulation. As an example, from a transmission perspective, for ONE to achieve the highest throughput requires a transmission bandwidth of 606 MHz, whereas 64-QAM with FIVE only needs $(606/840) \cdot 168 = 121\,\mathrm{MHz}$ to achieve the same data throughput.

The fabricated chip is verified up to a clock frequency of 160 MHz. This limit is due to the used pattern generator and logic analyzer. For the measurements, the core supply voltage $V_{\mathrm{dd}}$ was varied between the nominal value of 1.2 V down to 0.8 V, below which the voltage swings became too low for the pads to work properly. At the lower supply limit and highest clock frequency, the circuits still work properly such that more advanced investigations on energy-speed trade-offs could not be carried out.

The presented designs' estimated and measured power consumption are shown in Table I. Power estimation (numbers in parentheses) is carried out with Synopsys Power Compiler on the synthesized netlists, back-annotated with state- and path-dependent toggle information from a simulation run. Consider

the following comparison scenario: convolutional decoding using either a fixed (ONE) or the flexible design FIVE to find out how much power may be sacrifice for a certain flexibility. This comparison provides a measure of the initial cost of flexibility.

One may determine the error from the pre-layout power estimation figures compared to silicon measurements. One needs to evaluate the impact of static power consumption $P_{\text{stat}}$ to be able to extrapolate the measured values according to $P_{\text{dyn}} \propto f_{\text{clk}}$. The measured leakage current at the nominal core supply of 1.2 V was 50 $\mu$A, which gives $P_{\text{stat}} = 60\,\mu$W. However, this source of power consumption can be ignored as is seen from the measurements listed in Table I. The worst case contribution appears at the lowest measured power consumption of 5.78 mW and accounts for about 1%. At 0.8 V, the leakage current dropped to 14 $\mu$A (equiv. $P_{\text{stat}} = 11\,\mu$W). Advanced leakage minimization techniques such as power gating were not supported in the used technology. However, if one could use a low-leakage library, the measurements could still be conducted at the highest clock frequency of 160 MHz, while $P_{\text{stat}}$ is expected to drop by another factor of 10. Since $P_{\text{stat}}$ in all cases is at least two–three orders of magnitude lower than $P_{\text{dyn}}$, the extrapolation of the values in Table I results in estimation errors of 16% and 23% for designs ONE and FIVE.

Power consumption is sacrificed for flexibility. For design FIVE, 2.3 times more power is consumed for $R = 1$ processing. From an earlier power estimation, comparing a design with 16-QAM as highest modulation and FIVE, the latter consumes 4% and 9.7% more power in $R = 1$ and $R = 3$ modes, respectively. Furthermore, $R = 5$ in design FIVE requires an extra 3.4% power compared to $R = 3$, a low number considering the additional rate provided.

Fig. 5 shows plots of $P = f(V_{\text{dd}})$ for the two designs run at different $R$. It confirms the initial gap between fixed and flexible design, apparently when looking at the curves $R = 1$. However, once this penalty is paid, the incremental power to be spent for higher transmission rates is small.

If QPSK is often used and power consumption is a critical factor for the application, it makes sense to accept the additional fixed design. Otherwise, one flexible design that covers all transmission rates is sufficient.

In order to conduct an overall fair comparison, one further needs to introduce larger modulation schemes also for the R2-design ONE, which to date is solely based on a rate $1/2$ convolutional code using QPSK. Then the R2- and R2/R4-design can be compared based on equal transmission rate, and the contribution of task flexibility (different modulations and symbol mapping) in the BM unit related to (code) rate flexibility of trellis and SP units can be evaluated. From cell area of the different building block in the current design and their evolution using different modulations, task flexibility has a larger impact on an implementation than rate flexibility.

Comparisons to designs presented in Section II are difficult due to the mentioned different application areas and technologies. Chadha's [3] implementation for $m = 4$ is comparable to our design in terms of trellis complexity viz. number of branches per trellis stage. However, his implementation is for an FPGA, thus only gate count (23.5 kGates) serves as comparison measure.

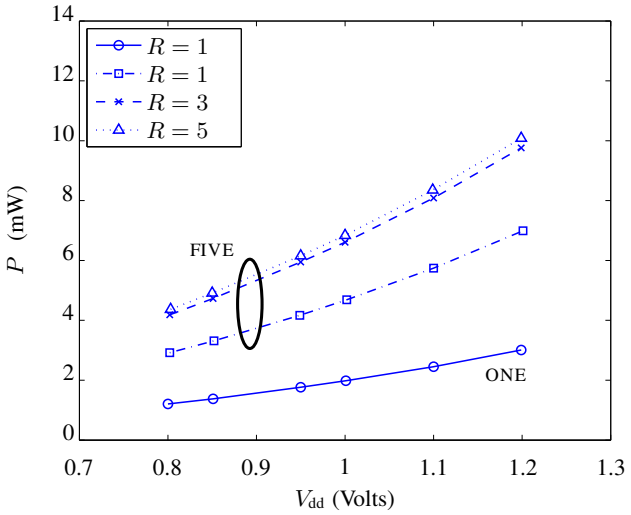


Fig. 5. Power consumption of designs ONE and FIVE at $f_{\text{clk}} = 160$ MHz.

## V. Conclusions

We presented a design of a flexible Viterbi decoder fabricated in 130-nm digital CMOS. To adapt to varying channel conditions, the flexibility applies to both code rate and modulation schemes. The cost of this flexibility is characterized on application and architectural level, which initially steered the design choices, as well as by measurements on the fabricated chip. Having accepted the initial impact of rate flexibility when comparing a fixed and a flexible design at their lowest transmission rate, one should incorporate more modulations since the additional cost in terms of power consumption becomes smaller.

## References

[1] "*Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)*," IEEE Standard 802.15.3, 2003.

[2] M. Kamuf, V. Öwall, and J. B. Anderson, "Optimization and implementation of a Viterbi decoder under flexibility constraints," *IEEE Trans. Circuits Syst. I, Reg. Papers*, available online at IEEE Explore.

[3] K. Chadha and J. R. Cavallaro, "A reconfigurable Viterbi decoder architecture," in *Proc. Asilomar Conf. Signals, Syst., and Comp.*, Pacific Grove, CA, Nov. 2001, pp. 66–71.

[4] D. E. Hocevar and A. Gatherer, "Achieving flexibility in a Viterbi decoder DSP coprocessor," in *Proc. IEEE Veh. Technol. Conf.*, Boston, Sept. 2000, pp. 2257–2264.

[5] H. T. Feldkämper, H. Blume, and T. G. Noll, "Study of heterogeneous and reconfigurable architectures in the communication domain," *Adv. Radio Science—Kleinheub. Berichte*, vol. 1, pp. 165–169, May 2003.

[6] M. A. Bickerstaff, *et al.*, "A unified turbo/Viterbi channel decoder for 3GPP mobile wireless in 0.18-$\mu$m CMOS," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1555–1564, Nov. 2002.

[7] J. R. Cavallaro and M. Vaya, "Viturbo: A reconfigurable architecture for Viterbi and turbo decoding," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, Hong Kong, Apr. 2003, pp. 497–500.

[8] H.-L. Lou, P. Tong, and J. M. Cioffi, "A programmable codec design for trellis coded modulation," in *Proc. IEEE Global Telecommun. Conf.*, Phoenix, Nov. 1997, pp. 944–947.

[9] T. Miyauchi, *et al.*, "High-performance programmable SISO decoder VLSI implementation for decoding turbo codes," in *Proc. IEEE Global Telecommun. Conf.*, San Antonio, TX, Nov. 2001, pp. 305–309.