Research Article

Power-Aware DVB-H Mobile TV System on Heterogeneous Multicore Platform

Yu-Sheng Lu,[1,2] Chin-Feng Lai,[1] Chia-Cheng Hu,[3] Han-Chieh Chao,[4] and Yueh-Min Huang[1]

[1] *Department of Engineering Science, National Cheng Kung University, No.1, University Rd., Tainan 701, Taiwan*
[2] *Business Customer Solutions Laboratory, Chunghwa Telecom Laboratories, No. 12, Lane 551,*
  Min-Tsu Rd. Sec.5 Yang-Mei, Taoyuan 326, Taiwan
[3] *Department of Information Management, Naval Academy, No. 669, Junxiao Rd., Zuoying District, Kaohsiung 813, Taiwan*
[4] *College of Electrical Engineering & Computer Science, National ILan University, No. 1, Sec. 1, Shen-Lung Rd., I-Lan 260, Taiwan*

Correspondence should be addressed to Yueh-Min Huang, huang@mail.ncku.edu.tw

In mobile communication network, the mobile device integrated with TV player is a novel technology that provides TV program services to end users. As TV program is a real-time video service, it has greater technical difficulties to overcome than a traditional video file download or online streaming, especially when TV programs are played on handheld devices. A challenge is how to save power in order to provide users with longer TV program services. To address this issue, this study proposes a mobile TV system on a heterogeneous multicore platform, which utilizes a Digital Video Broadcasting-Handheld (DVB-H) wireless network to receive the TV program signal, thus, saving power according to the features of DVB-H TV signal and heterogeneous multi-core.

1. Introduction

Along with the progressive digital TV broadcasting technology, TV viewing is no longer restricted by time or space; the new trend is to watch digital TV programs through wireless mobile devices. At present, watching TV on a mobile phone device can be performed in two ways. Service providers can transmit TV program data to mobile phone users by 3G network, or a base station can transmit TV programs through a Digital Video Broadcasting network [1–3]. The main difference between 3G and DVB network is that 3G network transmits data through on demand wireless network communication, hence, there will be transmission rates and bandwidth limits if too many users access this network at the same time. As to DVB-H, it transmits TV programs through the broadcasting transmission of TV base station; hence, there will be no transmission network congestion. Three issues have been studied regarding the mobile TV system: (1) TV signal transmission technology and how to enhance TV signal fault-tolerance or increase signal transport efficiency in order to improve display quality of TV programs; (2) mobile TV application developments and

provision of personal context aware services, recommending suitable TV programs according to user habits and preferences of watching TV; (3) how to enhance display quality, provide smooth TV programming if delays occur, and reduce power consumption in mobile TV players [4–7]. Concerning power-saving issues, two parts are discussed: (1) components of receiving TV signals, how to design receiver startup schedule while receiving a TV program signal to save receiver power; (2) design a power-saving play mechanism according to TV program signal features, after received TV signal is converted to digital data by the demodulator (Figure 1) [8–11]. Therefore, this study proposes a power-aware DVB-H mobile TV system on a heterogeneous multicore platform. This system is implemented in two major parts: a front-end buffer control mechanism and a parallel DVB-H TV signal decoding model.

When receiving a DVB-H TV program signal from a base station, signal is demodulated to generate video and audio data. As video bit rate, quality, and resolution are directly related to content complexity, running too many buffers will consume power, while too few buffers will cause the program to fail to be played successfully. Hence, this
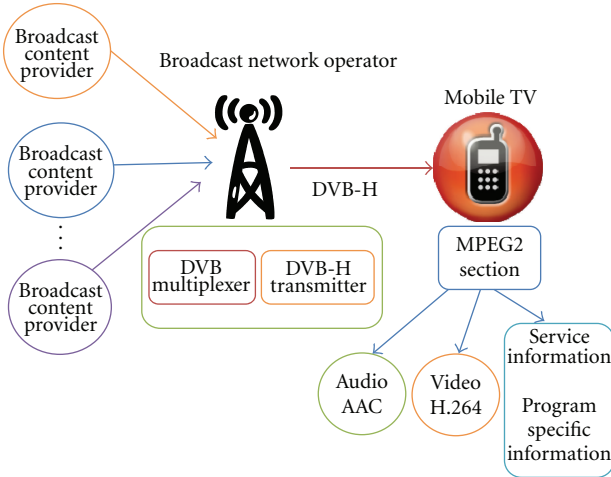
Figure 1: DVB-H Mobile TV Workflow.

Figure 2: DVB-H/T System Archietcture.

paper proposes a front-end buffer control mechanism to configure appropriate buffers according to the TV program video features, in order to utilize buffers and save power.

The parallel DVB-H TV signal decoding model uses a data partition processing method to run parallel DSP decoding of DVB-H videos on a heterogeneous multicore platform. It also schedules videos according to the DVB-H video features, in order to reduce data dependency among the frames on a multicore platform.

The remainder of this paper is organized as follows. Section 2 introduces DVB-H specification and the parallel decoding technique; Section 3 presents the overall architecture of the DVB-H mobile TV system, the front-end buffer control mechanism, and the processes and methodology of the parallel DVB-H TV signal decoding model; Section 4 discusses implementation and result, and Section 5 gives conclusions.

2. Related Work

2.1. Digital Video Broadcasting-Handheld. Digital Video Broadcasting-Handheld (DVB-H) is based on Digital Video Broadcasting-Terrestrial (DVB-T) specification and provides a solution to lower receiver power consumption and improves mobile receiving performance [12–18]. Figure 2 shows the outline of the DVB-H/T system specifications for common TV broadcasting programs using the DVB-T signal transfer mode. Senders can use an A/D converter to convert the analog video and audio signals to a digital signal, respectively, and use a Moving Picture Experts Group 2 (MPEG-2) codec technique to convert TV program data into MPEG-2 format. DVB-H service data are compressed and encapsulated into an IP packet then encapsulated into the transmission stream through a Multiprotocol Encapsulation (MPE) mechanism. Meanwhile, the time slicing data stream is added. Along with other DVB-T TV services, the multiplexer multiplexes it into a larger transmission stream (or multiple program transmission stream) before sending the data in a DVB wireless network. At the receiver,
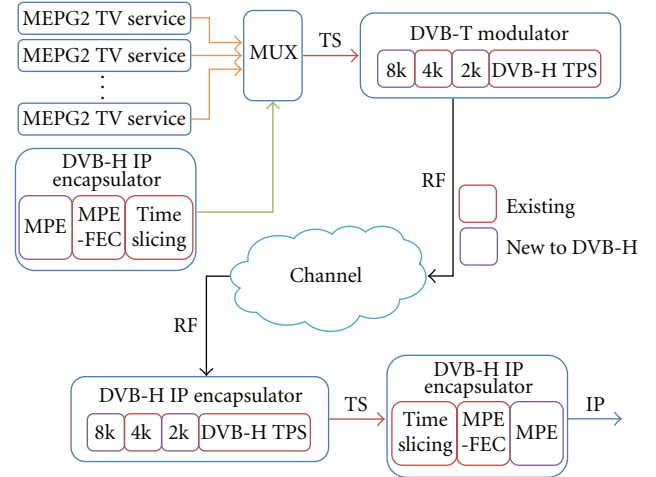
if a client wants to receive certain services, the receiver front-end circuit must run continuously in order to obtain the complete transmission stream. Then, the demultiplexer extracts the video, audio, and data information streams of the selected programs and delivers this information to the video decoder, audio decoder, and other applications for processing. The sender Multi-Protocol Encapsulation-Forward Error Correction (MPE-FEC) and time slicing mechanisms are collectively called the DVB-H IP-Encapsulator, while the receiver reverse recovery portion is called the DVB-H IP-Decapsulator. The overall DVB-H container format is shown in Figure 3. The IP data container format for each layer of DVB-H is shown in Figure 3, as an IP packet in the MPE section and redundant data in the FEC section. After Section format encapsulation, the MPE and FEC sections are connected end to end according to the encapsulating sequence to form a section data string. Then, it begins to slice the first and all of the other 184 bytes of each section data string. A 4-byte transmission stream header is added to the front of the 184-byte data length in order to complete a transmission stream encapsulation or MPEG2 transmission stream packet. Its data length is 188 bytes, with two major parts. The first is a data front-end header that occupies a 4-byte length with the available information, including a Sync. Byte = 47 hex for synchronizing the emitter and receiver, error indications, and stream packet recognition. The second part is the data transfer payload, which length is 184 bytes. In Figure 3, above the IP packet is the User Datagram Protocol (UDP) and the Real-time Transport Protocol (RTP). The top layer is compressed video data, where IP and UDP packets add their packet headers. The RTP packet is encapsulated and used to bear the H.264 images and AAC compressed voice, as RFC3984 specification.

2.2. Parallel Decoding. One ideal parallel process could double the system processing efficiency; however, when coding/decoding a picture, there exists a data dependency problem [19–25]. As the video image format contained in the DVB-H TV signal is an H.264 baseline format, this section
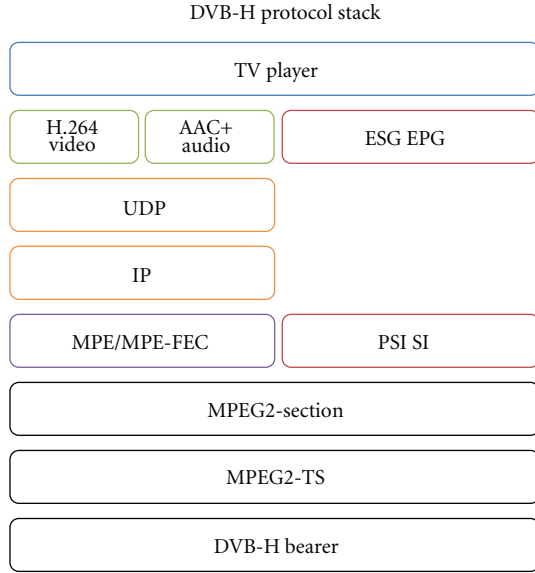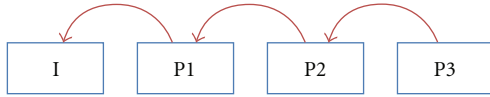
DVB-H protocol stack

FIGURE 3: DVB-H Protocol.

Sequence decode

Parallel decode
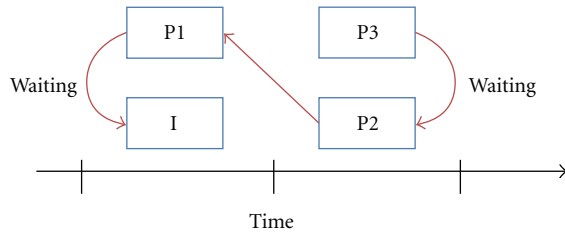
FIGURE 4: Sequence/Parallel Video Decode.

FIGURE 5: Functional Partition Decode Workflow.

introduces the H.264 image feature. In H.264 decoding, pictures are divided into I frame, P frame, and B frame, where P frame is decoded according to the I frame picture data, and the B frame refers to the picture data of the I frame and the P frame. Unless there is good parallel processing, data collision will occur, as shown in Figure 4. When decoding two interdependent pictures, even when both pictures are simultaneously processed, the other picture must wait for a decoded reference before decoding. Therefore, how to utilize parallel processing to shorten the operation waiting time is the focus of many studies. Parallel decoding is divided into two orientations, a function partition, and a data partition, detailed as follows.

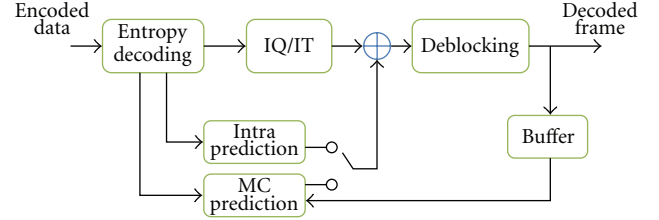*2.2.1. Function Partition.* The H.264 decoding process can be roughly divided into entropy decoding (ED), inverse quantization and inverse transform (IQ/IT), intra- or inter-prediction (PPC), and deblocking filter (DF). As shown in Figure 5, the function partition divides the entire H.264 decoding process into independent tasks. The main purpose is to use a balanced processing concept to configure the tasks required by each processor so that each processor can share the load, and processing can be accelerated. The advantage of this mechanism is that it can easily and extensively eliminate data dependency, while its disadvantage is that its task division is subject to a number of processors.

2.2.2. Data Partition. Data partitioning divides decoding data into partitions, which are computed by different processors. Each processor performs the same data operations, but process different data units. Previous literatures have studied how to divide data while avoiding data dependency; the partition includes groups of picture (GOP) levels, frame levels, slice levels, and macroblock levels. Their features are detailed below.

GOP level [21, 22]: it divides the video segments in GOP, allocates each GOP to each processor to decode, as each section of the GOP can independently run decoding. Decoding in this manner can increase processing quantities at linear speed [23]; however, applying this technique requires large memory space to save the decoded GOP fragments.

Frame level: this parallel decoding method allocates each single picture to a respective processor to operate, where preanalysis sorting operations or a tournament algorithm is adopted in order to process picture allocation. Primarily, two pictures without data dependency are found and simultaneously operated. Flierl and Girod [24] proposed a B Frame parallel decoding method, which is suitable for traditional coding methods, and because the B frame is not referred to by other picture, no data dependency will occur. B frame is analyzed first and allocated to different processors for decoding. However, regarding H.264 coding, B frame can be a reference for other pictures, therefore, is not suitable here.

Slice level [25, 26]: in H.264, the Slice is the smallest independent decoding unit, meaning that a single Slice can independently run decoding. Therefore, similar to GOP level partitioning, various Slices are allocated to various processors for decoding. As compared with the GOP Level, this parallel method is more favorable to memory utilization without extra analysis sorting. However, its main disadvantage is that, slice divisions can range from a macroblock to one entire frame, where memory use, scalability, and balance
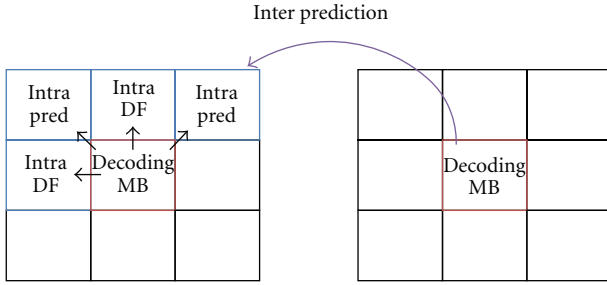
FIGURE 6: Macroblock Decoding References.

FIGURE 7: System Architecture.

remain unsatisfied. Roitzsch [27] proposed a Slice-Balancing algorithm, which can improve its scalability; however, it is for the coder end and cannot be applied in the decoder end.

Macroblock level: a frame consists of many picture macroblocks, and each macroblock is allocated to a variable processor to operate. This scheme has the best scalability and balance; however, it requires the most directions for solutions, such as data dependency. As shown in Figure 6, in H.264 decoding, each macroblock must refer to its neighboring macroblock to operate. The main concept of macroblock level parallel decoding is to locate two macroblocks, which are without data dependency in order to speed the rate decode of decoding. Van der Tol et al. [28] proposed an echelon sorting process to solve the data dependency problem. Although this sorting method can improve the speed of decoding, it is subject to a number of processors. In cases of high resolution pictures, this algorithm requires complex operations and several processors. Chong et al. [29] proposed scheduling with parse, render, and filter, locating the dependency relation of each macroblock prior to sorting. Azevedo et al. [30] proposed a 3-D-Wave method to integrate frame and macroblock levels in order to locate decodable macroblocks across all frames, which solves scalability and data dependency. However, the algorithm applied in a super multicore algorithm remains difficult to implement.

Although many papers have addressed the parallel decoding problem, few studies have focused on the power consumption for DVB-H TV program. To this question, this paper presents a novel architecture to obtain a front-end buffer control mechanism and a parallel decoding model to increase the speed of decoding TV program and reduce the power consumption according to picture complexity.

3. System Architecture

Figure 7 shows the system architecture proposed in this study. After accessing the system, DVB-H streaming data are saved to an external memory; the main processor unit (MPU) initializes and starts the digital signal processor (DSP), and then, the streaming data is moved to the internal memory of DSP to decode. The system waits for completely compressed data before entropy decoding (ED), and the H.264 baseline profile is similar to common compression specifications, as per context-adaptive variable-length coding (CAVLC). Inverse quantization and inverse
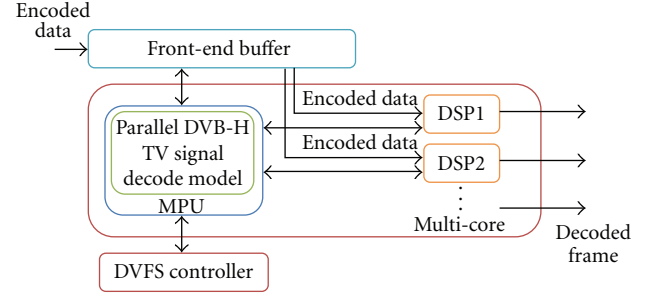
transform (IQ/IT) are done on compressed pictures, then intraprediction or interprediction is made according to the video format. Finally, a deblocking Filter (DF) is carried out on the imaged pictures to eliminate boundary effects and improve image quality. After decoding, if the picture is referred to, then it is saved to internal memory. Otherwise, the direct memory access unit (DMA) will move the data to an external memory, and the MPU will transfer the picture data to a frame buffer for displaying. As to H.264 specifications, it adopts the concept of a network abstraction layer (NAL) to adapt to a progressive streaming application. The NAL Unit is the streaming unit, which includes a header and the contents of compressed video data or decoded auxiliary information (e.g., resolution, display time, and video information). The MPU locates the compressed video data and decodes it according to the auxiliary information. The streaming data are parallel sorted. Dynamic voltage and frequency scaling (DVFS) system decoding prediction is performed according to sorted video dependency and video formats [31–33]. In most cases, the DSP system codes/decodes videos through a heterogeneous multicore platform. Therefore, this paper focuses on parallel decoding of a single MPU, coupled with a multi-DSP-core platform. The MPU controls parallel planning, DVFS prediction, and settings of system. With the front-end processing design, parallel processing of the DVFS system can be performed without changing the DSP decoding process. The results of the multicore platform could also be applied to a parallel decoding design on another platform.

3.1. Front-End Buffer Control Mechanism

3.1.1. Group of Picture Unit of DVB-H Video Content. DVB-H TV programming is composed of H.264 formatted images, and each H.264 stream consists of numerous group of picture (GOP). Each GOP consists of I-frames, B-frames, and P-frames, where I-frame is used for DCT-based compressed digital video frame, and B-frame and P-frame are used as backup frames to enhance compression ratio. Due to picture interdependency, which comes from the motion vectors of the I, P, and B frames in GOP and compensation coding, when the DVB-H TV signal accesses a system to decode, parallel decoding of DVB-H TV signal is performed according to GOP features. Since H.264 GOP size is subjected to the complexity of a picture in its content, this study
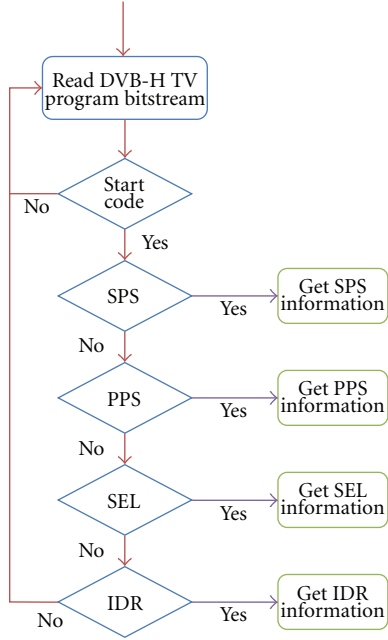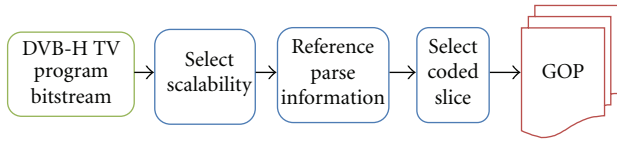
FIGURE 8: DVB-H TV Signal Reference Parse.

FIGURE 9: GOP Divide Workflow.

regulates the size of the front-end buffer according to the information provided by each GOP header in order to save power. First, an entire DVB-H bitstream is received from a receiver, then the entire bitstream is analyzed according to its streaming manner, and 00 00 00 01 is the start code of the NAL unit, and then moves to next byte and determines the NAL unit type. If the NAL unit type is SPS, PPS, or SEI, it collects the sequence, start position, data size, instantaneous decoder refresh (IDR) sequence, percentage of IDR sequence over entire coded video sequence, and the start position. The entire process is called reference parse, as shown in Figure 8. According to the parse result, the size of each GOP and its number of frames contained can be known. Then, the IDR feature is used to select the proper number of intrakey pictures, and each intrakey picture is converted into an IDR picture, as required. This study adds IDR pictures for original reference and changes its slice header syntax element, in case of decompression failure or incomplete state as each GOP unit changes its tunable combination (Figure 9).

3.1.2. Front-End Buffer Configuration. One ideal parallel processing can increase the speed of system processing; however, data dependency often occurs when coding/decoding multimedia data. This study designs a simple parallel decoding architecture that could solve data dependency, as shown in Figure 10. The architecture utilizes a buffer mechanism to
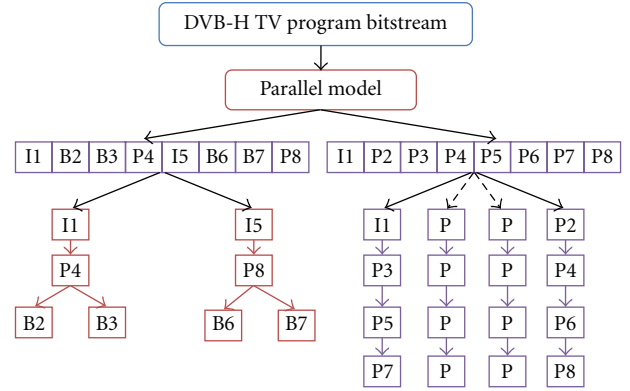

FIGURE 10: Parallel Decoding Architecture.

store stream data and then waits for the next independent picture before decoding together. A buffer must be set for GOP to be decoded, and the buffer size is also a focus of smooth TV programming, as excessive buffers may occupy too much system memory, and insufficient buffers may have too few fragments leading to image delay.

However, in a prolonged dependent video format, this may cause latency and buffer size problems. It is because prolonged latency is unacceptable to the streaming process, and buffer size is subject to hardware size, which influences power consumption. Thus, this study designs a dynamic allocation mechanism and defines a front-end buffer (FEB), which is used to store streaming data for parallel processing. Since FEB is not overflowed, independent parallel decoding is adopted when the system locates the next independent picture. However, when streaming data overflows FEB and no independent picture appears, dependent parallel decoding is used to meet data streaming features, and the frame and macroblock level for the integrated parallel decoding concept are applied to synchronize decoding. This decoding method is performed by the transfer of a synchronous signal.

The steps for choosing FEB to satisfy a streaming system are defined as follows: Suppose that T_{acp} denotes a time coefficient acceptable to the end user, then, the time spent on the entire computing process is

$$T = \frac{\mathrm{FEB}}{S} + T_{\mathrm{proc}} + D, \tag{1}$$

where, T_{proc} denotes the DVB-H data streaming speed, and D denotes the FEB derived latency. To comply with a streaming system, the following equations must be satisfied:

$$T_{\mathrm{acp}} = \frac{\mathrm{FEB}_{\mathrm{acp}}}{S} + T_{\mathrm{proc}} + D > T, \tag{2}$$

$$\mathrm{FEB}_{\mathrm{acp}} = \frac{\left(T_{\mathrm{acp}} - T_{\mathrm{proc}} - D\right)}{S}. \tag{3}$$

As the system is a frame-based case, suppose that T_{proc} denotes the processing time required for a picture, hence, $\mathrm{FEB}_{\mathrm{acp}}$ is the preset size.
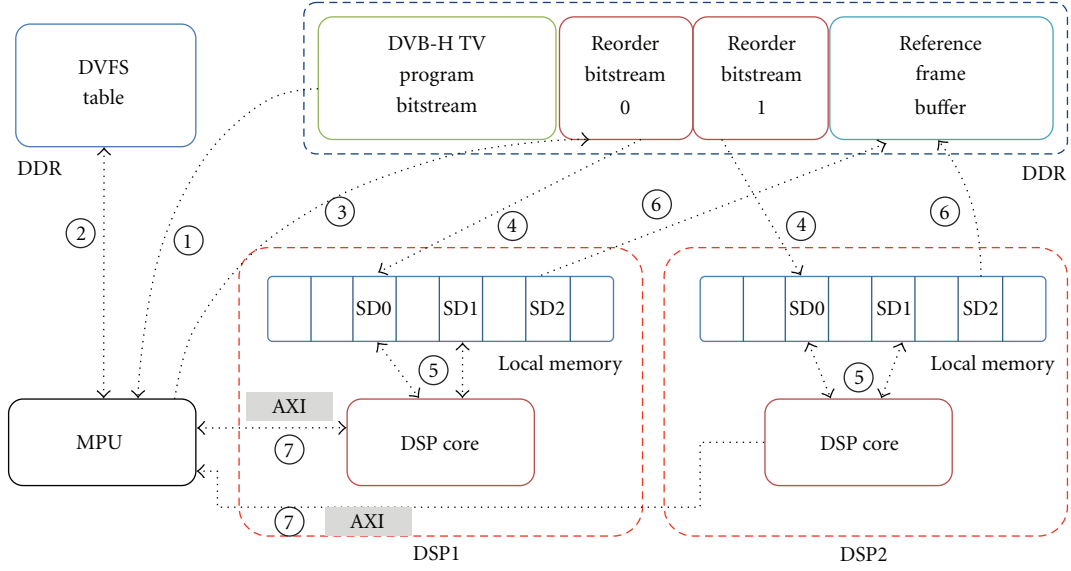
FIGURE 11: Data Parallel Architecture.

3.2. Parallel DVB-H TV Signal Decoding Model. As to a multimedia decoding system, the two most important methods for reducing energy consumption are (1) elimination of time slack, (2) prediction of processing quantity required for the next picture and presetting the voltage/frequency. To achieve these two objectives, this study uses a simple, yet practical, concept of using front end and back end buffers to construct the entire DVFS architecture, realize the parallel mechanism, and eliminate the time slack. The constant decoding time of each picture is defined as a deadline. However, rather than changing a deadline schedule, a power approximation method is used to predict system voltage and frequency and corrects the system voltage and frequency according to the weight of each task in the decoding program.

The power consumed by a processor during the CMOS manufacturing process is defined as

$$P = C_{\text{eff}} * V_{\text{dd}}^2 * f, \tag{4}$$

where, C_{eff} denotes an effective switch capacitance, V_{dd} denotes working voltage, and F denotes working frequency, and the frequency versus voltage relation can be expressed by the following:

$$f = K \frac{(V_{\text{dd}} - V_t)^a}{V_{\text{dd}}}, \tag{5}$$

where, K is a constant, V_t denotes the threshold voltage, and $a = 1.2 \sim 2$ denotes the electron coefficient [25]. The time spent in executing a task is called the workload and is defined as T_{Proc}, which can be calculated from the following:

$$T_{\text{Proc}} = \frac{C}{f}, \tag{6}$$

where, C is the number of cycles required for this task during system computing, and by substituting (2) for (3), we can obtain

$$T_{\text{Proc}} = \frac{C}{F} = C \frac{V_{\text{dd}}}{K * (V_{\text{dd}} - V_t)^a}, \tag{7}$$

according to the energy formula

$$E = P * T_{\text{Proc}}. \tag{8}$$

The architecture of data parallel implementation is shown in Figure 11. In this architecture, MPU analyzes the complexity of TV bit streams firstly, sets voltage and frequency for each TV bit stream according to the materials of DVFS Table, and then transfers TV bit streams to DSP core for decoding. After decoding TV bit streams, the decoded data is delivered to Reference Frame Buffer for playing TV programs. We can reduce the power consumption via DVFS adjustments. Different from a program-oriented parallel architecture, which evenly allocates one identical picture to various DSPs to decode, the data-oriented parallel architecture sends a picture to different DSP for separate decoding. Therefore, when a primary data stream arrives, the MPU analyzes its information, sorts, reallocates the pictures, then sends it to the DSP for decoding. The DVFS mechanism adopts a frame-based direct mechanism in order to dynamically tune the DSP voltage and frequency, as per the ratio of picture data size/decoding seconds. The difference is dynamic tuning performed during the workload period. The tuning equations are expressed by (7) and (8). Data-oriented parallel processing has another problem, when simultaneously decoded pictures are dependent, and then there is data synchronization problem. This study adopts a picture coordinate synchronizing method, where a macroblock coordinate of the picture being decoded is
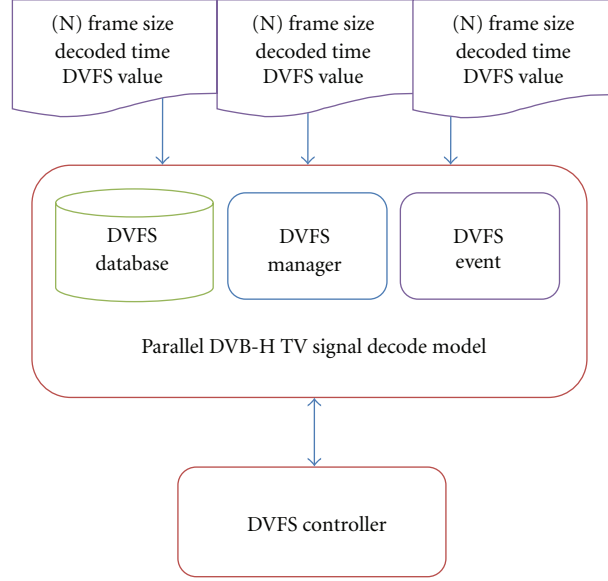
FIGURE 12: Prediction Mechanism for DVFS.

TABLE 1: Testing Sample.

Video						
Name	Mobile	Foreman	Highway	Silent	Football	Speaker
Type	Motion	Silent	Motion	Silent	Motion	Silent
Resolution	QCIF (176×144)	QCIF (176×144)	QVGA (320×240)	QVGA (320×240)	VGA (640×480)	VGA (640×480)
Frame type	1I3P6B (IPBBPBB)	1I3P6B (IPBBPBB)	1I3P6B (IPBBPBB)	1I3P6B (IPBBPBB)	1I3P6B (IPBBPBB)	1I3P6B (IPBBPBB)
Frame number	300	300	300	300	300	300

$(X_{\mathrm{cur}}, Y_{\mathrm{cur}})$, and the coordinate of its reference picture is $((X_{\mathrm{ref}}, Y_{\mathrm{ref}})$, then, their relation formula is as follows:

$$Y_{\mathrm{ref}} \geq Y_{\mathrm{cur}} + N_{\mathrm{mac}} \mid Y_{\mathrm{ref}}, \quad Y_{\mathrm{cur}} \leq Y_{\mathrm{Max}},$$
$$X_{\mathrm{ref}} \geq X_{\mathrm{cur}} + N_{\mathrm{mac}} \mid X_{\mathrm{ref}}, \quad X_{\mathrm{cur}} \leq X_{\mathrm{Max}}, \tag{9}$$

where, X_{Max} and Y_{Max} denote the numbers of boundaries of the divided picture, and N_{sub} denotes the number of macroblock referenced by the macroblock.

3.2.1. Prediction of System Cycle Number C. In this paper, an offline come online mechanism is adopted in order to lower the prediction error rate. The method is shown in Figure 12.

DVFS is divided into two parts in order to implement the entire design architecture, and the offline mechanism is implemented on the MPU. On the DSP, the online mechanism is realized through the dynamic tuning of voltage and frequency the decoding process. Prior to a system decoding process, the DVFS Model determines the initial voltage and frequency, set according to the picture format, previously decoded picture size, and decoding time. In the decoding process, the DSP end decides the dynamic tuning of voltage and frequency, set according to time spent and the dependency data of each function in the decoding process.

Without knowing the exact cycles consumed by the next decoding picture, this study first predicts through offline statistics and builds a database in the MPU part, which records time, type, voltage, and frequency required for analyzing a video. Based on the analysis data of the previous picture of the same type, this study defines two formats: (1) the averaged cycles, C_{avg}; (2) the cycles of a reference picture, C_{prev} and video content variation rate, α

$$\alpha = \frac{\left| C_j - C_{j-1} \right|}{C_j}. \tag{10}$$

The predicted cycles vary with the type of picture in the variable format. In a more static neighboring video, prediction by the previous format would be better. In a dynamic video, the average prediction would be more accurate. Therefore, different prediction effects are designed for each picture format. As shown in (11), as I frame is an
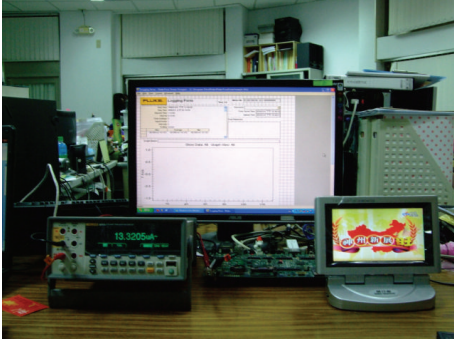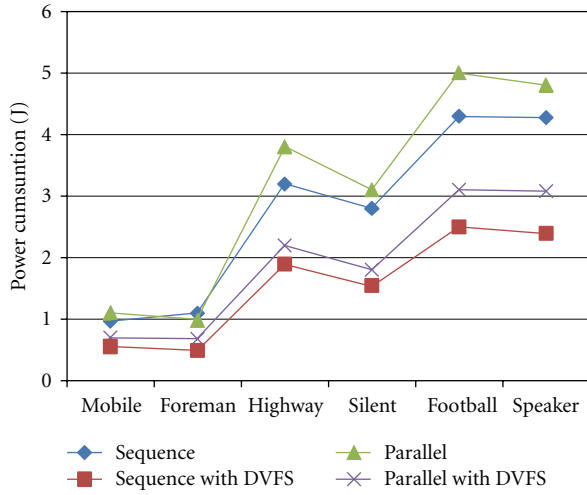
FIGURE 13: Experimental Environment.

FIGURE 14: Power Cumsuntion.

FIGURE 15: Deadline Miss.

FIGURE 16: Error Rate.

independent picture without a reference picture, the average prediction mode is adopted. In B frame, the picture refers to both a previous and a next picture, thus, the cycles of the B frame are estimated from the average of the reference pictures. As to P frame, the average forecast or reference picture forecast is adopted according to α:

$$
C_{est} = \begin{cases} C_{avg}, & \text{for I frame} \\ C_{avg}, & \text{for P frame \&\& } \alpha \geqq 15\% \\ C_{ref}, & \text{for P frame \&\& } \alpha < 15\% \\ \dfrac{\sum_N C_{ref}}{N}, & \text{for B frame.} \end{cases} \tag{11}
$$

4. Experiment and Result

The experimental environment is shown in Figure 13. This study uses Fluke 8846A to measure the power consumed by each DSP during decoding and displays the data on a computer through FlukeView software. First, for dynamic or static variable resolution TV program in Table 1, the common DVB-H TV signal decoding and parallel DVB-H TV signal decoding methods are used to measure the consumed power. The results are shown in Figure 14, where
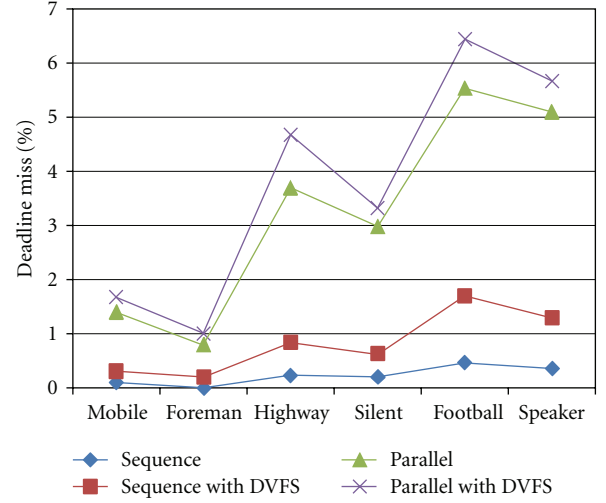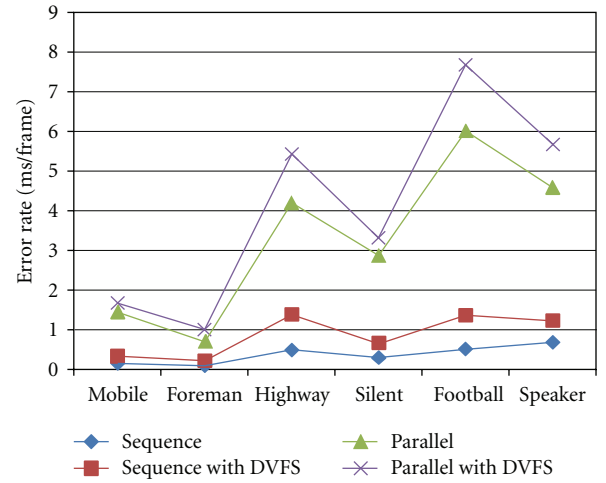
39% of the total power loss is saved. This parallel DVFS architecture shows that a Data parallel architecture can perform better for power control of a dynamic picture. After decoding, this study lists the decoding seconds, data size, and error occurrence of each picture, and calculates its error rate according to following formula:

$$
err_{rate} = \frac{\sqrt{\sum (T_{est} - T_{dec})^2}}{Num_{frame}}, \tag{12}
$$

$$
err_{miss} = \frac{Num_{miss}}{Num_{frame}} * 100\%, \tag{13}
$$

where, T_{est} denotes the estimated decoding time, T_{dec} denotes the actual decoding time, Num_{frame} denotes the number of pictures, and Num_{miss} denotes the number of pictures with missed deadlines. The statistical results are shown as Figures 14 and 15. When decoding a dynamic or high resolution picture, the use of the data parallel architecture has a high error rate, presumably because the data parallel

architecture does not forecast at regular periods. Upon a missed deadline, continuous deadline misses will occur due to data dependency.

5. Conclusions

This study proposed a power-aware DVB-H mobile TV system on a heterogeneous multicore platform and established a front-end buffer control mechanism and a parallel DVB-H TV signal decoding model. Applying a parallel architecture could increase the speed for the decoding of a DVB-H TV program. Dependent on picture complexity, the DVFS system can be used for dynamic tuning of the system voltage and frequency to lower power consumption. The experimental results confirmed that applying a DVFS system could save as much as 39% power, which could increase the service time of some mobile TV devices. If coupled with a receiving scheduler mechanism at the Receiver, even more energy could be saved. When decoding dynamic or high resolution pictures, a high error rate is occurred in the data parallel architecture, since the data parallel architecture is not forecasted at regular periods probably. Upon a missed deadline, continuous deadline misses will occur due to data dependency. Those issues are interesting challenges in the parallel video decoding architecture.

Acknowledgment

References

[1] S. Parkvall, E. Englund, M. Lundevall, and J. Torsner, "Evolving 3G mobile systems: broadband and broadcast services in WCDMA," *IEEE Communications Magazine*, vol. 44, no. 2, pp. 68–74, 2006.

[2] S. Buchinger, S. Kriglstein, and H. Hlavacs, "A comprehensive view on user studies: survey and open issues for mobile TV," in *Proceedings of the 7th European Conference on European Interactive Television Conference*, Leuven, Belgium, 2009.

[3] E. Kaasinen, M. Kulju, T. Kivinen, and V. Oksman, "User acceptance of mobile TV services," in *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '09)*, September 2009.

[4] H. Fuchs and N. Färber, "Optimizing channel change time in IPTV applications," in *Proceedings of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB '08)*, April 2008.

[5] S. Lee, J. Koo, and K. Chung, "Content-aware rate control to improve the energy efficiency in mobile IPTV services," *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 5, pp. 52–58, 2010.

[6] Y. Solomon, "The Economics of Mobile Broadcast TV," Mobile DTV Alliance Whitepaper, January 2007.

[7] J. Valerdi, A. González, and F. J. Garrido, "Automatic testing and measurement of QoE in IPTV using image and video comparison," in *Proceedings of 4th International Conference on Digital Telecommunications*, Colmar, France, 2009.

[8] S. Yaldiz, A. Demir, and S. Tasiran, "Stochastic modeling and optimization for energy management in multicore systems: a video decoding case study," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1264–1277, 2008.

[9] P. Francesco, P. Antonio, B. Davide, B. Luca, and M. Poletc, "Energy-E cient multiprocessor systems-on-chip for embedded computing: exploring programming models and their architectural support," *IEEE Transactions on Computers*, vol. 56, no. 5, pp. 606–621, 2007.

[10] M. Hefeeda and C.-H. Hsu, "On burst transmission scheduling in mobile TV broadcast networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 610–623, 2010.

[11] M. Rezaei, I. Bouazizi, and M. Gabbouj, "Joint video coding and statistical multiplexing for broadcasting over DVB-H channels," *IEEE Transactions on Multimedia*, vol. 10, no. 8, pp. 1455–1464, 2008.

[12] M. Kornfeld and G. May, "DVB-H and IP datacast—broadcast to handheld devices," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 161–170, 2007.

[13] ETSI EN 302 304, "Digital Video Broadcasting (DVB): transmission system for handheld terminals (DVB-H)," European Standard, v.1.1.1, 2004.

[14] ETSI TR 102 377 v1.3.1, "Digital Video Broadcasting (DVB); DVB-H Implementation Guidelines," May 2007.

[15] ETSI TS 102 472 v1.2.1, "IP Datacast over DVB-H: Content Delivery Protocols," December 2006.

[16] "Digital Video Broadcasting - Handheld (DVB-H) home page," 2008, http://www.dvb-h.org/.

[17] D. Gómez-Barquero and A. Bria, "Forward error correction for file delivery in DVB-H," in *Proceedings of IEEE Vehicular Technology Conference*, pp. 2951–2955, Dublin, Ireland, April 2007.

[18] M. Kornfeld and G. May, "DVB-H and IP datacast - Broadcast to handheld devices," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 161–170, 2007.

[19] C. Meenderinck, A. Azevedo, B. Juurlink, M. Alvarez Mesa, and A. Ramirez, "Parallel scalability of video decoders," *Journal of Signal Processing Systems*, pp. 1–22, 2008.

[20] A. Azevedo, C. Meenderinck, B. Juurlink, et al., "Parallel H.264 decoding on an embedded multicore processor," in *Proceedings of the 4th International Conference on High Performance and Embedded Architectures and Compilers*, pp. 404–418, January 2009.

[21] T. Olivares, F. J. Quiles, P. Cuenca, L. Orozco-Barbosa, and I. Ahmad, "Study of data distribution techniques for the implementation of an MPEG-2 video encoder," in *Proceedings of Parallel and Distributed Computing Systems*, pp. 537–542, November 1999.

[22] A. Bilas, J. Fritts, and J. P. Singh, "Real-time parallel MPEG-2 decoding in software," in *Proceedings of the 11th International Parallel Processing Symposium (IPPS '97)*, pp. 197–203, April 1997.

[23] D. Farin, N. Mache, and H. N. Peter, "SAMPEG, a scene adaptive parallel MPEG-2 software encoder," in *Visual Communications and Image Processing*, vol. 4310 of *Proceedings of SPIE*, pp. 272–283, San Jose, Calif, USA, January 2001.

[24] M. Flierl and B. Girod, "Generalized B pictures and the draft H.264/AVC video-compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 587–597, 2003.

[25] A. Rodriguez, A. Gonzalez, and M. P. Malumbres, "Hierarchical parallelization of an h.264/AVC video encoder," in *Proceedings of International Symposium on Parallel Computing in Electrical Engineering*, pp. 363–368, September 2006.

[26] Z. Zhao and P. Liang, "Data partition for wavefront parallelization of H.264 video encoder," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 2669–2672, May 2006.

[27] M. Roitzsch, "Slice-balancing H.264 video encoding for improved scalability of multicore decoding," in *Proceedings of the 7th ACM and IEEE international conference on Embedded software*, pp. 269–278, September 2006.

[28] E. B. van der Tol, E. G. T. Jaspers, and R. H. Gelderblom, "Mapping of H.264 decoding on a multiprocessor architecture," in *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, pp. 40–49, November 2008.

[29] J. Chong, N. Satish, B. Catanzaro, K. Ravindran, and K. Keutzer, "Efficient parallelization of H.264 decoding with macro block level scheduling," in *Proceedings of IEEE International Conference onMultimedia and Expo (ICME '07)*, pp. 1874–1877, July 2007.

[30] A. Azevedo, C. Meenderinck, B. Juurlink, A. Terechko, et al., "Parallel H.264 decoding on an embedded multicore processor," in *Proceedings of the 4th International Conference on High Performance and Embedded Architectures and Compilers*, pp. 404–418, January 2009.

[31] W. Lee, K. Patel, and M. Pedram, "GOP-level dynamic thermal management in MPEG-2 decoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 6, pp. 662–672, 2008.

[32] I. Yeo, H. K. Lee, E. J. Kim, and K. H. Yum, "Effective dynamic thermal management for MPEG-4 decoding," in *Proceedings of IEEE International Conference on Computer Design (ICCD '07)*, pp. 623–628, October 2007.

[33] M. Mehendale, "Socs for portable video applications: architecture level considerations," in *Proceedings of IEEE Electronic Design Processes Workshop*, pp. 213–217, April 2007.