Tracking the game of Blackjack

by

Gabriel Ménard

Department of Electrical and Computer Engineering
McGill University
Montreal

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering

ABSTRACT

In this thesis we present a system that identifies the cards dealt during a game of Blackjack, as played at the game tables in a casino. In a first place, the tracking of a game of Blackjack is done to validate the work of the dealer by correlating results with an external bets and payouts tracking system to determine if the player actually won or lost. In a second place, this system is intended to detect card counting activities at the Blackjack table. Such practices are not illegal but given that it gives an advantage to the player, casinos prefer to decline the playing privilege to these players.

To perform tracking of the game, the system utilizes a standard resolution video feed from a camera above the game table. This is an important aspect to facilitate the implementation of the system since such video feeds are readily available in typical casino installations. The field of view encompasses all seven players' dealing spots as well as the dealer's dealing spot. The system recognizes the cards being dealt and reports the results to an operator.

The system is implemented using a FPGA device and the latter acts as the video processing engine that interfaces the video camera. The hardware is specifically crafted to adapt the devised algorithm.

RÉSUMÉ

Ce mémoire présente un système qui permet d'identifier les cartes distribuées lors d'une partie de Blackjack telle que jouée à un casino. En premier lieu, le suivi d'une partie de Blackjack est fait pour valider le travail du croupier en vérifiant les résultats à l'aide d'un système externe d'identification de mises qui permettra de déterminer si les joueurs ont effectivement gagné ou perdu. En second lieu, le système ouvre la porte à la détection de la pratique du comptage des cartes par les joueurs à la table de Blackjack. Bien qu'une telle pratique ne soit pas illégale, l'avantage qu'elle apporte aux joueurs fait en sorte que les casinos préfèrent retirer le privilège de jouer aux gens étant reconnus comme compteurs de cartes.

Pour faire le suivi d'une partie, le système utilise un signal vidéo de résolution standard en provenance d'une caméra située au-dessus de la table de jeu. Cet aspect est important pour faciliter l'implantation du système étant donné que de tels signaux vidéos sont déjà disponibles à partir des systèmes de surveillance des casinos. Le champ d'observation de la caméra permet de voir le jeu des sept joueurs ainsi que du croupier. Le système est en mesure d'identifier chacune des cartes distribuées et achemine les résultats à un opérateur.

Le système est implanté à partir de logique programmable qui agit en tant qu'unité de traitement de la vidéo en provenance de la caméra. Le matériel a spécifiquement été créé pour s'adapter à l'algorithme développé.

Contents

List of Figures

List of Tables

This work would not have been possible without the support and encouragement of my girlfriend, family and friends. I would also like to thank James Clark for letting me work under his supervision.

Chapter 1

Introduction

In the following sections, background information on the game of Blackjack is presented. First, the rules of the game are explained. Second, the problem to be solved is described. Griffin has presented a review of the theory of Blackjack and the following is inspired by his book, *Theory of Blackjack* (Griffin, 1999). In addition, the description that is done on how players can gain advantage over the casino by counting cards is a primary motivation to implement the presented system.

1.1 Blackjack rules

In a game of Blackjack, up to seven people play against a dealer. The dealer deals two cards to every player including himself. The players' cards are dealt face up while the dealer has one card face up and one card face down (hole card). Following the initial cards dealing, each player is sequentially given the opportunity to request new cards, one at the time. The action of requesting a new card is referred to as hitting. Not requesting a new card is referred to as standing. The goal of the game is to get as close as possible to 21 without exceeding this value, an event referred to as busting.

The cards 2 to 10 are worth their face value, the jack, queen and king (face cards) are worth 10 and the ace is worth 11 unless the card causes the player to bust, in which event the card is worth 1. Each player plays against the dealer and not between them. A player is a winner if he has the highest hand without busting. If a player and the dealer have the same hand value, it is a push and no one wins. It is also the case if both the player and the dealer have a Blackjack. In the event that a player and the dealer both bust, the player still loses. If the player or the dealer scores 21 on the first two card deal, this event is called a Blackjack.

Once each player has stood or busted, the dealer reveals his hole card and plays his hand. Differently from the players, the dealer obey specific rules to determine when he needs to stop hitting. These rules can be specific to each casino and table but the dealer will commonly hit as long as he has 16 or less regardless of the players' hand. Usually, the dealer must hit on a soft 17, which is a hand with at least one ace.

Before each round, each player is requested to make a bet. If the player loses or busts, he loses his bet. If the player wins, he receives an amount equal to his bet. If the player wins with a Blackjack, he receives 1.5 times his bet.

Under certain conditions, the flow of the game of Blackjack can be altered. A split can be requested by the player if his two cards have the same value. To split, the player must place an additional bet equal to his initial bet. Following this, the dealer deals a second card to the two newly created hands. The two hands are played separately and their outcome is independent. A double down can be requested by the player on his first two cards. To double down, the player must double his bet. Then, the dealer draws a third card which will be the last card the player will have. In the event that the dealer shows up an ace on the initial deal, the players are offered to take insurance. To take insurance, the player must place an additional bet worth half his initial bet. In the event that the dealer has a natural Blackjack (an ace and a 10-valued card), the insurance pays at 2:1 odds, otherwise the player loses his insurance

bet.

1.2 Motivations

Blackjack is more than a game, it is an industry and an important one since it is the most played table game in casinos (Griffin, 1999). Because the casino (or house) never stops playing, the law of large numbers can be applied to determine the house edge. The house edge is the house's long-term advantage and can be evaluated by a percentage: for every dollar gambled at a Blackjack table, the casino can expect a given percentage of profit in the long run. Vigilant players play the game using what is known as the basic strategy. The basic strategy is a set of rules that dictates the player's action on all given situations. For example, if the player has a 2 and a 7 and the dealer is showing a 5, the basic strategy instructs the player to double. Table 1.1 pictures the basic strategy. This strategy, which is the result of computer simulations, was first established by Edward O. Thorp in 1962 and described in his book *Beat the Dealer* (Thorp, 1966). Against a player using basic strategy, the house edge fluctuates between 0.4% and 1% depending on the rule variations (Griffin, 1999). This edge is small but the casino can assume profit at the game of Blackjack given this is the theoretical worst case scenario.

In an ideal world, the casino could rely on its edge to maintain its profit but cheating can drastically affect its income. The casino sees cheating as any activity that increases the player's advantage over the house edge. Cheating could be as simple as a player secretly increasing his bet when he is winning but can also be very subtle. Here we highlight different cheating methods and explain how a game tracking system can help the casinos identify suspicious players.

	Dealer Upcard									
Player Hand	2	3	4	5	6	7	8	9	10	A
7	H	H	H	H	H	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H
9	H	D	D	D	D	H	H	H	H	H
10	D	D	D	D	D	D	D	D	H	H
11	D	D	D	D	D	D	D	D	D	H
12	H	H	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
16	S	S	S	S	S	H	H	H	H	H
17	S	S	S	S	S	S	S	S	S	S
A,2	H	H	H	D	D	H	H	H	H	H
A,3	H	H	H	D	D	H	H	H	H	H
A,4	H	H	D	D	D	H	H	H	H	H
A,5	H	H	D	D	D	H	H	H	H	H
A,6	H	D	D	D	D	H	H	H	H	H
A,7	S	D	D	D	D	S	S	H	H	H
A,8	S	S	S	S	S	S	S	S	S	S
A,9	S	S	S	S	S	S	S	S	S	S
2,2	P	P	P	P	P	P	H	H	H	H
3,3	P	P	P	P	P	P	H	H	H	H
4,4	H	H	H	P	P	H	H	H	H	H
5,5	D	D	D	D	D	D	D	D	H	H
6,6	P	P	P	P	P	H	H	H	H	H
7,7	P	P	P	P	P	P	H	H	H	H
8,8	P	P	P	P	P	P	P	P	P	P
9,9	P	P	P	P	P	S	P	P	S	S
T,T	S	S	S	S	S	S	S	S	S	S
A,A	P	P	P	P	P	P	P	P	P	P

Legend

H: Hit
S: Stand
P: Split
D: Double

Table 1.1: Blackjack basic strategy

1.2.1 Card counting

Card counting is a method where the player uses information given to him by the flow of the game to increase his knowledge on how to play the game. In card counting, the player keeps track of all the cards previously dealt from the deck to dictate his betting scheme and modify the basic strategy to his advantage. The underlying principle is that a high count, a high ratio of high cards (10's and aces) to low cards (2 to 9), is advantageous to the player (Griffin, 1999). The player can therefore wager more when the deck is high in high cards and wage less when the deck is high in low cards. At the same time, the richness of high or low cards in the deck can help the player not to bust by modifying the basic strategy.

All the above is done mentally by the player and is therefore very difficult to identify visually by the pit bosses[1]. The best means to identify card counting is for the casino to keep track of the count at every table and identify players betting according to this. Since most dealers and surveillance employees are not qualified to do so and are susceptible to fatigue, an automated tracking system is a plausible solution to the problem.

1.2.2 Pastposting

Pastposting is an industry specific term to describe the cheating method by which a player replaces smaller denomination chips with chips of larger denomination. This cheating method only implies the player's dexterity and skill of getting unnoticed by the dealer, pit bosses and the surveillance cameras.

[1] A pit boss is a casino authority figure responsible of supervising a group of game tables

1.2.3 Accomplice dealer

If the dealer is accomplice to one or more players, many cheating methods can be employed. In a first case, the dealer has the ability to pay more when a player wins and to avoid taking a player's bet when the latter loses. In the second case, the dealer can signal the value of his hole card to an accomplice player which significantly increases the player's odds. Finally, the dealer has the possibility, if he has the necessary skills and dexterity, to distribute advantageous cards to an accomplice player. In addition, even if the dealer is not cheating, lack of concentration can result in errors at the expense of the casino.

1.3 Contributions and Overview

From the description above of the possible cheats, one realizes there are two aspects to the discovery by an automated system. In a first time, the cards dealt must be identified and understood as being the hand of each player including the dealer. This part of the system enables the possibility to keep track of the count and to determine which players actually won or lost. In a second time, the bets must be identified and correlated with the latter game data.

The work presented in this thesis addresses the first problem where each card is visually identified by a camera located above the game table. The cards identification process must isolate the cards dealt and recognize their value. The isolation of the cards is especially challenging because cards are placed one on top of each other in a diagonal manner.

The bet identification system is assumed to be taken care of by existing technology. The best existing solution for bet analysis is accomplished by RFID detection where each casino chip is equipped with a radio frequency identifiable tag that is read by

the game table (Shuffle Master, 1997). Other cheating methods like card marking are not addressed by the current thesis.

Chapter 2

Literature Review

The existing literature that covers the present topic is somehow limited. A single work addresses the concept itself but others can be considered to have solved portions of it.

Wesley Cooper from the University of Dublin presented a Blackjack tracking system (Cooper, 2004). The system uses a camera placed above the game table to provide a video feed to a personal computer that processes the data by software. To achieve real-time processing of the data, the system uses a 2.4 GHz processor analyzing 360 pixels x 290 pixels frames at a rate of 8.3 frames per second. With the latter specification, the system was able to achieve 4% error rate on test footage. The error rate degraded to 45%-50% when using a 360 pixels x 240 pixels frame resolution. Identification of the cards was based on features extraction of the the cards' dot patterns. Recognition of multiple overlapping cards was addressed by a contour tracing algorithm. The greatest limitation of the system is its inability to be scaled easily. Given the intensive required processing power, one personal computer per Blackjack table is required. Because modern casinos have dozens of Blackjack tables, integrating such a system is impractical. In counterpart, the system facilitates the

positioning of cameras by applying image normalization. This means that the camera does not need to be directly above the gaming table to provide an orthogonal view of the game. However, distancing the camera affects the resolution of the video feed, an aspect that strongly dictates the quality of the results.

Seth Eatinger, Ben Graf, Micheal Victory and Ray Wagner from Rice University presented a playing card recognition project for the game of poker (Eatinger *et al.*, 2000). The research project implemented a card identification algorithm in Matlab based on very high resolution card images (not an actual video feed). Cards were identified using template matching, essentially by matching the card's corner (number and suit). The multiple overlapping cards problem was not addressed because cards are juxtaposed and not overlapped in a game of poker. The error rate of the system ranged from 12% to 45% depending on conditions. The system is not suitable for real-time applications because analysis of the image content takes up to one minute. The error rate is higher but the analysis task is also more difficult than the game of Blackjack. That is mainly because card suits must be identified and the face cards must be discriminated one from another (not all valued to 10).

Three commercial products also address the tracking of the game of Blackjack. Because they are not in the public domain, their actual implementation cannot be reviewed but the methods used are described in order to identify key technologies.

Tangam Gaming manufactures an automated table tracking product (Tangam Gaming, 2007). The system uses a video feed from a camera above gaming tables and a software to track the card games. The system interprets the video to recognize and record actual game events happening at the table. The key feature of this system is that it uses only the provided video feed to track the card game. This is the approach demonstrated in the current thesis and such a technique is not used by the other products described below.

Shuffler Master manufactures an intelligent card shoe for the game of Baccarat (Shuffle Master, 2007). The system has the ability to scan and identify every card dealt from the shoe but does not provide any information on the flow of the game. The system can therefore only be applied to games such as Baccarat where the flow of the game can be easily tracked or where inputs from the dealer or an additional system could be used to interpret the flow of the game. Furthermore, if such a card shoe was to be used in the game of Blackjack, it could create currently inexistent security issues. That is because the dealer's hole card (card not shown to the players), which is only known to the dealer would become accessible to the outside world by the means of dishonest operators or hacking, a situation that creates a problematic greater than the issue it tries to solve.

Bally Technologies manufactures a specially-designed Blackjack tabletop that incorporates many features to monitor players' actions (Bally Technologies, 2007). The system uses specially encoded playing cards, using invisible ink and bar codes. It also incorporates 14 small cameras built into the dealer's chip tray. These cameras can read every card in play by reading the invisible ink printed on them. Such a system is highly intrusive to the playing environment and requires major equipment change and investment (US$20,000 per tabletop). In addition, regular Blackjack players criticize the invasive nature of this type of system and tend to avoid such tables.

Chapter 3

System Implementation

In this chapter, we describe the approach that was elaborated to solve the problem of tracking cards at the game of Blackjack. Firstly, the framework for addressing the given image and video processing problem is explained. Secondly, the algorithm that enables the tracking of cards using the previously mentioned framework is described. Finally, details on hardware implementation of the algorithm are given.

3.1 Image and Video Processing Framework

3.1.1 System Overview

In the current work, the sole data source used for the tracking the cards is the video feed from the surveillance camera located directly above the Blackjack table. The greatest advantage of this approach is the simplicity of integration of the system. In modern casinos, having a camera at each game table is a de facto standard (Rehrmann *et al.*, 1999) and therefore makes the system non-intrusive and easy to implement.

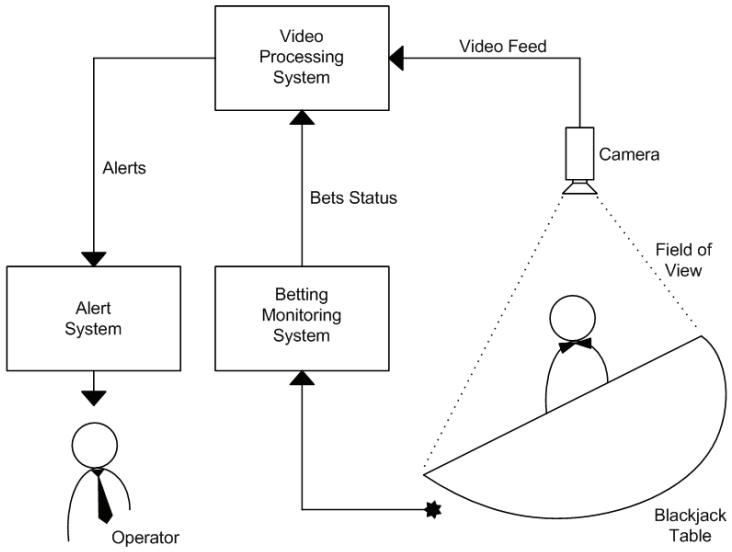

Figure 3.1: Card tracking system overview

The video feed is provided to a video processing system that applies the devised algorithm. When abnormalities are detected by the video processing system, the latter reports these abnormalities to an alert system that is accessible by an operator.

3.1.2 Video Processing

Because of the nature of the game of Blackjack, simplifying assumptions can be made of the extracted video feed. First, the color information is irrelevant because the suit of the cards is not considered in Blackjack. The color of the table mat could however be used to facilitate the isolation of cards but given that the contrast between the cards (white) and the mat (usually green) is high, color information can still be discarded. The system therefore utilizes a black and white video feed to proceed to card tracking. Second, the face cards, that all have a value of 10, are not independently discriminated. This means that the system will treat as a ten any jack, queen or king.

The field of view of the camera is illustrated in Figure 3.2. The table has seven player spots in addition to the dealer's spot. Pixel coordinates for the complete frame originate at the top left corner. A typical video feed is constituted of 720 pixels wide by 480 pixels high frames at a rate of 30 frames per second. Each player spots are given parameterizable attributes that are specific to each camera and game table. A first set of parameters that must be defined is the coordinates of the four corners of each player's first card spot. A second parameter is the width and height of a playing card on the table as lying at the center the game table. It is assumed that the geometry of a playing card may vary when its position is changed within the field of view and the latter parameter is used as a reasonable estimate. These parameters are valid as long as the camera and the table remain in the same position. This implies that a dedicated non-moving camera is located above the table.

Typical activity (i.e. movement) over this field of view is restrained to the dealer's arm and hand when he is dealing the cards. The task of the dealer is following a particular order and this fact simplifies the processing of the video feed. The dealer always deals sequentially one card from player 1 to player 7, skipping empty spots (no players), and finally deals a card to himself. He proceeds the exact same way a second time except for his second card that is face down. Empty spots are recognized by the external betting system since no bets are placed and this information is used by the card tracking system. It is assumed that activity over the camera field of view is only caused by the dealer with the exception of the players having liberty to move beyond the first card spots. The dashed line in Figure 3.2 pictures the regions where activity from the dealer and the players is respectively expected. Obviously, the player activity has no effect on the card tracking algorithm and is simply ignored.

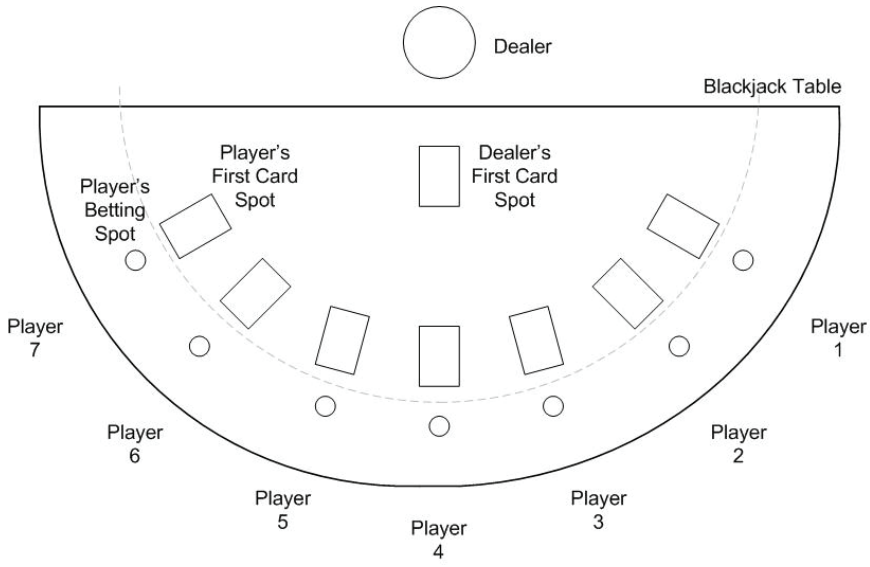

Figure 3.2: Camera Field of View

3.1.3 System Implementation

For such a real life application as tracking the game of Blackjack, devising an algorithm is not sufficient to fully solve the problem and the system implementation is therefore an equally important part of the challenge. As described above, a camera serves as the input to the system. Although it is not refuted that a fully analog system could accomplish the task, the current work concentrates on a digital implementation and for this reason, the video signal is assumed to be digital.

The main task to be accomplished is the video processing of this data and three options are considered: 1) Software implementation using conventional computing devices (i.e.: Von Neumann computer architecture); 2) Software implementation targeted at digital signal processor (DSP); and 3) Custom hardware implementation.

As the literature review stated, software implementation is possible using readily available computing devices such as personal computers. However, the performance was demonstrated to be marginal. This is explained by the fact that the central processing unit (CPU) is designed to accomplish any and all generic tasks. The problem

of image processing is a computationally expensive one and general processors are not optimized to accomplish this. In addition, scalability of the system to many Blackjack tables is questionable because one complete computer would be required per table. On the other hand, software implementation using a digital signal processor would provide improved processing capabilities because its multiply and accumulate unit (MAC) is specifically well suited for image processing tasks such as filter and transform computations. However, this kind of processor lacks the ability to handle control and other general tasks. For that reason, control peripherals such as a microcontoller unit (MCU) or programmable logic would need to be added to the system to accept the video source and to manage the alert system. At this regards, a custom hardware implementation where it is possible to incorporate a custom digital signal processing unit and a control unit appears as a unified and attractive solution. The digital signal processing can be custom-made and be specifically adapted to the current task (e.g. specific data path). In addition, computationally expensive operation can be directly implemented in hardware with custom logic to offload the digital signal processor. This has the overall advantage of having an extremely efficient hardware.

In terms of hardware implementation, two possibilities arise for this digital design: develop an Application-Specific Integrated Circuit (ASIC) or make use of a Field-Programmable Gate Array (FPGA). In terms of developing the Register Transfer Level (RTL) description, both ASIC and FPGA use a similar process. In the end, the hardware will take the form of Hardware Description Language (HDL), either Verilog or VHDL. Even more, the simulation of the HDL is also identical for both technologies. However, when it comes to prototype and to finalize the design, differences appear. The ASIC, as its name implies, is a custom integrated circuit and when its HDL is transposed into a particular process (0.18 micrometer, 0.13 micrometer, 90 nanometer, etc.), it is a final and definitive design. A single change in the HDL and the costly process of creating the ASIC must be completely redone. This means that for creating a single working prototype, tremendous sums of money and time

must be invested. For that reason, an ASIC is mostly targeted at very high-volume products. On the other hand, a FPGA is a programmable digital integrated circuit for which implementing a design is a matter of using ready-made logic cells. The HDL is compiled for the targeted device in terms of these logic cells and the FPGA can be programmed as many times as required. When a final working design is reached, the same integrated circuit can be used for the end application. Furthermore, even when the FPGA is integrated into the system, it can be re-programmed with a newer version of the hardware to improve the system. This can be done as long as the surrounding peripherals (such as the video input and alert system in the current work) can keep up with the FPGA ameliorations. For these reasons, this thesis considers the implementation of the system in a field programmable gate array device rather than an application-specific integrated circuit.

Chapter 4

Card Tracking Algorithm

Here we describe the card tracking algorithm.

4.1 The Flow of the Game

Tracking the game of Blackjack is facilitated by the logical flow that every game follows. In a first time, bets are accepted and this defines which player spot(s) must be tracked. A first card is expected for every player and the dealer, then a second a card. Note that the dealer's second card is hidden underneath his first card. Following the distribution of the two initial cards, each player must decide repeatedly if he desires another card until he is satisfied with his score or until he busts. The dealer proceeds in a similar fashion once every player has stood or busted. The goal of the card tracking algorithm is to identify every card being put face up on the table as they are dealt and that is done according to this flow in illustrated Figure 4.1.

At this regard, specific observable areas can be defined before hand to reduce the amount of information needed to be processed by the system. The first card being

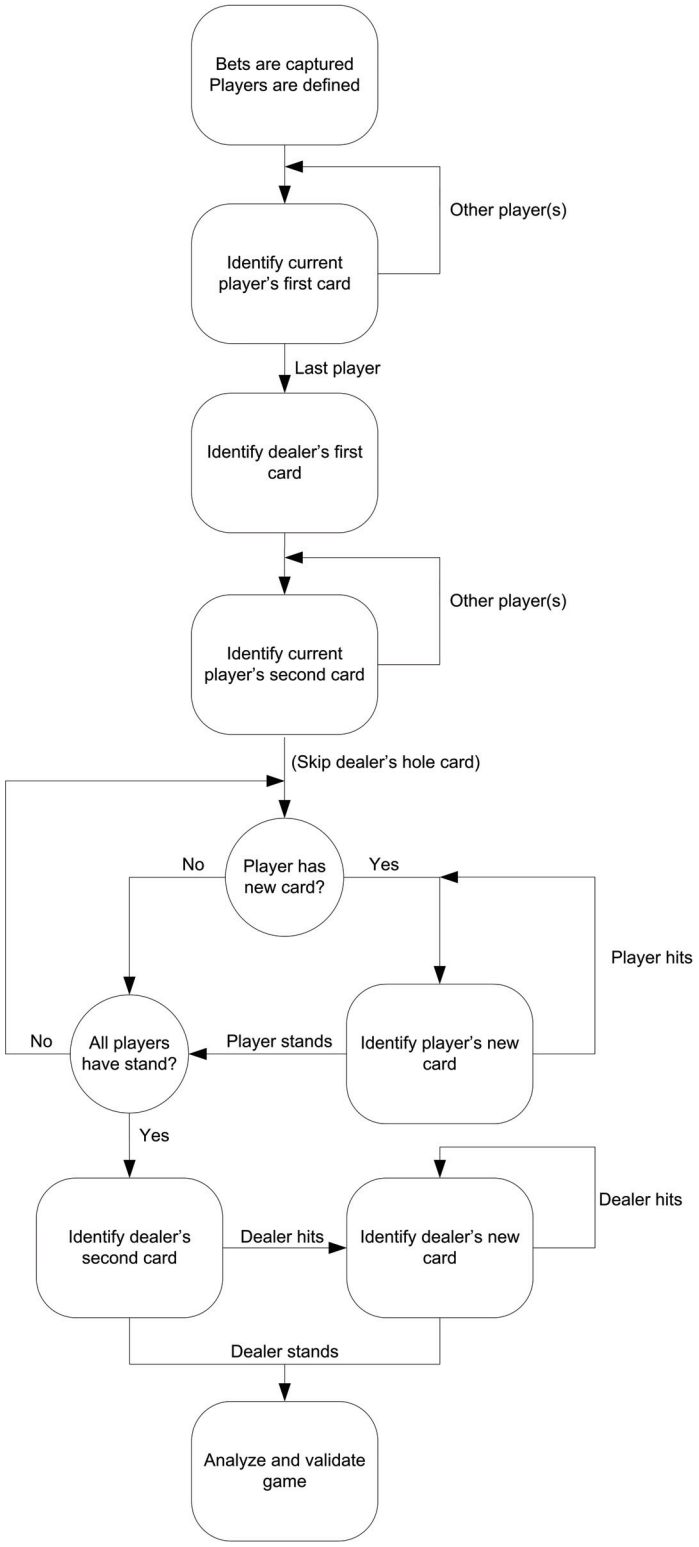

Figure 4.1: Flow of a Game of Blackjack

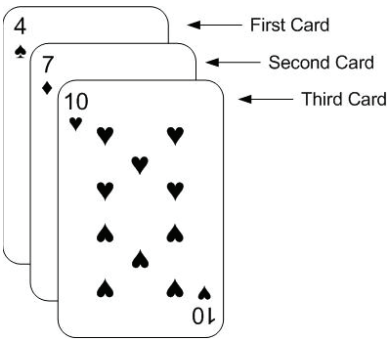

Figure 4.2: Cards Overlapping

dealt to every player is placed within a predefined rectangular region. The second and following cards are overlapped over the preceding card in an oblique manner as pictured in Figure 4.2.

4.2 Identification of the Players' First Card

Because color information is irrelevant to the identification of cards, and that is especially true because of the high contrast between the cards and the playing surface, each frame of the video feed is thresholded to an 8-bit black and white image. This first step reduces greatly the computational complexity. Figure 4.3 illustrates an original frame (left), its grayscale correspondence (middle) and the final thresholded black and white image (right). Note that the threshold value from 0 to 255 is specific to the environment conditions (especially lighting) and must be calibrated from one game table to another.

As mentioned above, identification of the first card of each player is done inside a predefined observable area for each player's spot. The area corresponds to the rectangular shape of the cards but is larger to facilitate the distribution done by the dealer. However, common Blackjack playing surfaces as shown in Figure 4.4 do not include such marking to identify the area where the first card should be dealt. Note

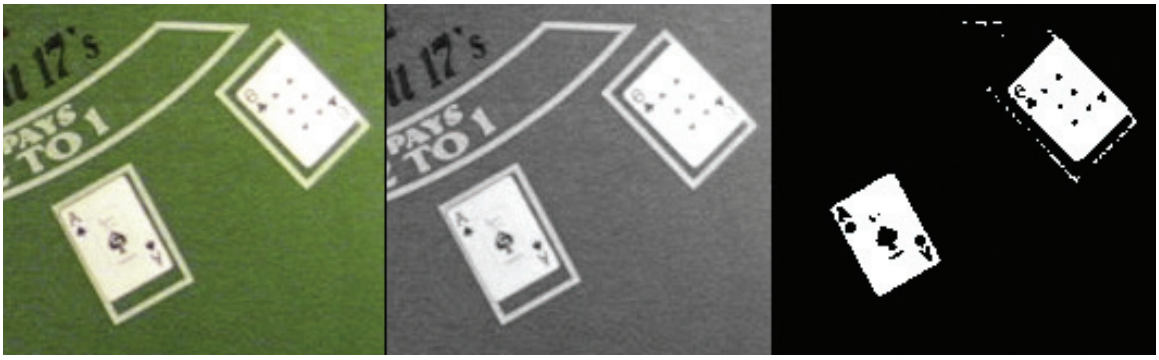

Figure 4.3: Frame Thresholding

there are some rectangular spots on the table but they are normally used for the bets. This problem is trivial and can be solved either by having dealers sufficiently trained to deal the cards constantly in this area and/or by adding barely noticeable marks to existing playing surfaces. To reduce the computational complexity and avoid rotation of the oblique observable areas of players 1, 2, 3, 5, 6 and 7, the latter areas are enlarged to be at 90 degrees with respect to the field of view as pictured in Figure 4.5. In this work, the rectangular spots marked on the table are used to deal the first card to better illustrate the flow of the game.

Given that observable areas are now determined, a numerical analysis method is needed to perform the detection of the first card for each player, that is when motion appears in the observable area and then stops. The frame when the motion completely stops is the instant that the card has been put on the table. Following the work in *Statistical analysis of a change detector based on image modeling of difference picture* (Park and Hush, 1990) on motion detection, the sum of the squared-difference appears as a computationally efficient yet effective method for the current problem. The sum of the squared-difference is defined as

$$\sum_{j=1}^{n}\sum_{i=1}^{m}(A_{i,j}-B_{i,j})^2$$

for two consecutive frames A and B of dimension m x n. In addition, the sum of

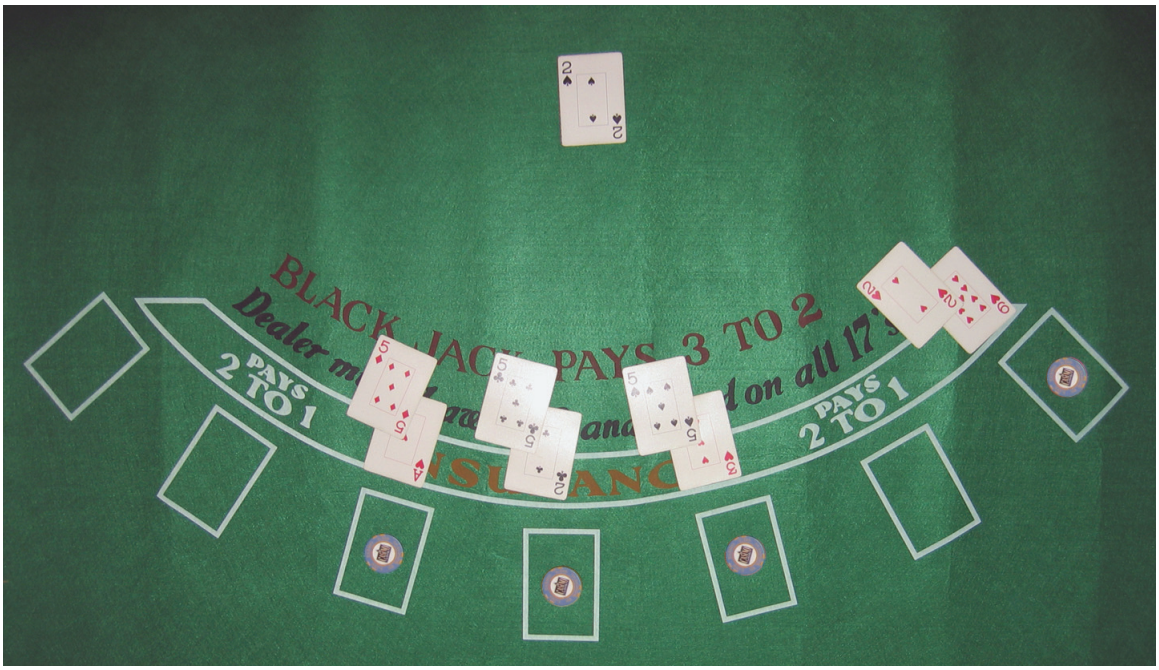

Figure 4.4: Typical Blackjack Playing Surface

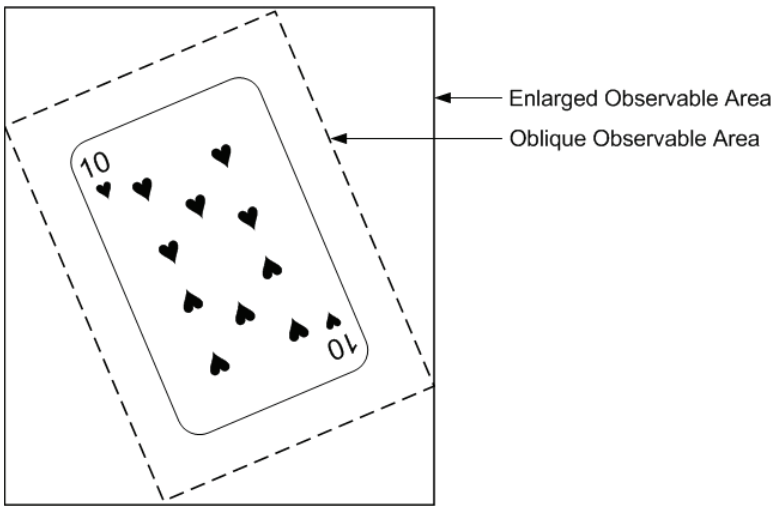

Figure 4.5: Oblique Observable Areas

the squared-difference is applied only to the pixels defined by the original observable area. That is because the enlarged observable area does not carry any information about the detection of the card. The result of this operation on every consecutive frame is a numerical value that expresses the level of difference between two frames. When the sum of squared-difference exceeds a certain threshold, motion is detected (a card enters the observable area) and when the sum of squared-difference returns to a static value, the card has been put on the table.

However, it is computationally intense to realize the sum of squared-difference in real-time (Lowe, 1999) over the complete observable area in the wait of card. For this reason, a simpler method is used to determine when a card first appears in the observable area. Instead of analyzing the complete observable area, a single line as pictured by Figure 4.6 is used. This line is referred to as the detection line. As long as the sum of absolute difference between two consecutive detection lines is below a certain threshold, there is no motion detected. Note that the sum of absolute difference is used instead of the sum of squared-difference to reduce the computational complexity. Once motion is detected, the sum of squared-difference method is applied on the following frames to determine with precision the frame when the first card is put in the observable area. It is not possible to use the detection line method to determine when the motion stops because elements such as the dealer's fingers can still be in the observable area while the detection line "sees" a static card. This situation is illustrated in Figure 4.7 where the detection line perceives the motion in A) but fails to notice the difference between B) and C). To overcome this, motion is assumed to be stopped and the card fully dropped on the table once a constant delay is reached after motion is initially detected.

Having identified the frame when the card lies in the observable area, the latter is used to identify the value of the card.

The first step to identify the card's value is to trim the card in the image being

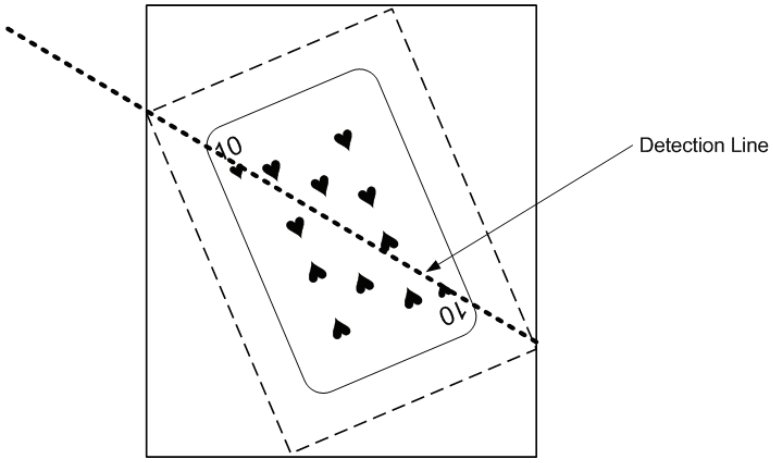

Figure 4.6: Detection Line

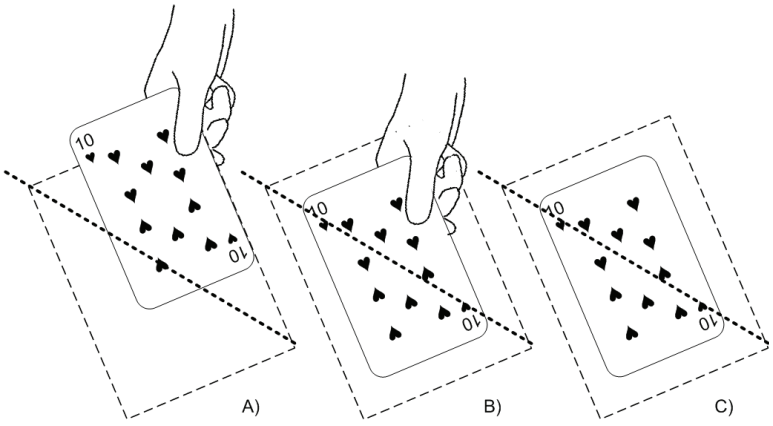

Figure 4.7: Dealer's hand not perceived by detection line

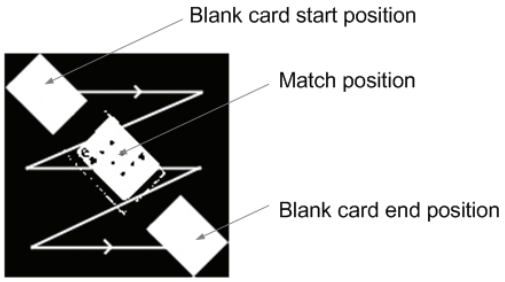

Figure 4.8: Blank card matching

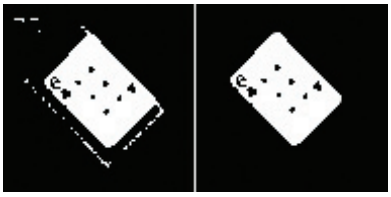

Figure 4.9: Isolated card following blank card matching

analyzed. This means that all the surrounding artifacts must be blanked to white to match the card's main color. To accomplish this, a blank (white) card is generated based on the fact that the card dimensions are known. The blank card is rotated to match the player's spot angle, again a known constant, because the card in the image is most likely to be at approximatively that angle. Then, by applying the sum of squared-difference for every position of the blank card over the image of the first card, a match determining the position of the card is obtained when the sum of squared-difference reaches its lowest value. This process is illustrated in Figure 4.8. Note that the enlarged observable area is larger than in fact to better illustrate the blank card sweep process. This matching is possible based on the fact that the card is mainly white like the blank card and the surrounding region is mainly black. Having matched the blank card to the card in the image, it is trivial to crop the image information inside that region, as shown in Figure 4.9, using the blank card coordinates.

The current card now isolated, the next step is to identify the value that of card. For the cards valued from ace to 10, this is done by counting the number of dots that appear at the center of the cards. The face cards are a special case that is

Figure 4.10: Portion of the card to isolate

explained later. One could argue that identification of the card's value could be done by optical character recognition (Trier *et al.*, 1996) or template matching (Prasad and Iyengar, 1996) but both these methods are computationally expensive and rely on high-resolution images of the cards for satisfactory results. As it will be demonstrated, the proposed method is simple and can operate with low-resolution images. This is especially true because the suit of cards and value of figure cards are irrelevant.

In order to count the dots in the image or to identify a figure card, it is necessary to first remove the card's border. Figure 4.10 illustrates the portion of the card that needs to be isolated. Because the card's edges are known using the blank card coordinates, it is possible to trim the dots or figure by applying a ratio based on the known dimensions of the cards. The trim ratio is constant for every card on a given camera/table setup but must be calibrated for each game table.

The center of the card is now isolated and we proceed to the determination of the value of the card. This is done by applying two tests as illustrated in Figure 4.11. First, a black to white ratio is established. Figure 4.12 shows the black to white ratio for every card in a typical deck of card. The black to white ratio allows to discriminate figure cards because these have more black pixels than white ones. Because all figure cards have a value of ten, no further analysis is required. Second, the black pixel regions (dots) are isolated and counted. Trivially, the number of dots

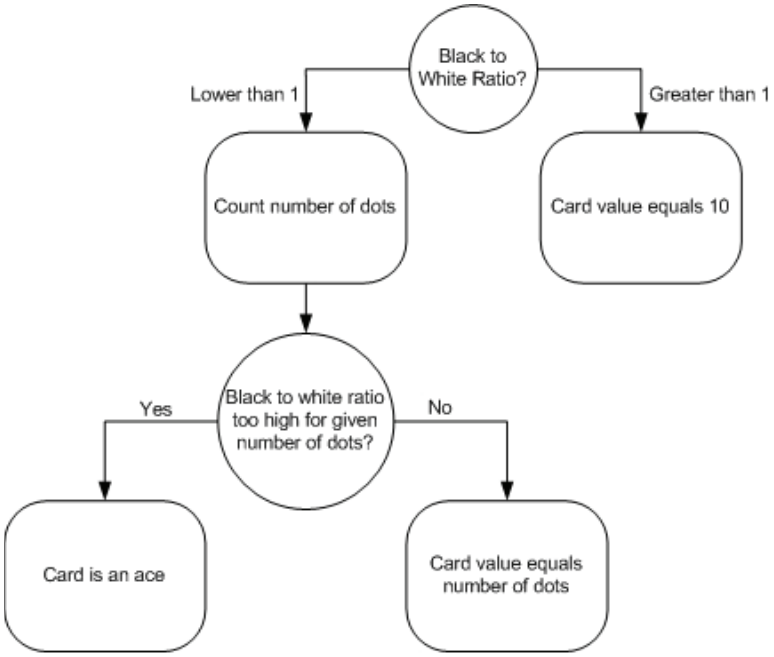

Figure 4.11: Card value determination process

is equal to the value of the card.

However, a special case may arise in some decks of cards. As Figure 4.13 shows, the ace of spades can contain more than a single black pixel region because the brand of the deck may be added to this card. To overcome this issue, the black and white ratio is used in addition to the given number of black pixels regions appearing on the ace. Because the black and white ratio is lower than a figure card but higher than a card with few dots, this case is easily discriminated as 4.12 shows.

The procedure described above to determine the first card is repeated for every player.

4.2.1 Identification of the Dealer's First Card

The method to identify the dealer's first card is the same as for the players. However, unlike the players' observable areas where motion only appears when a card is dealt

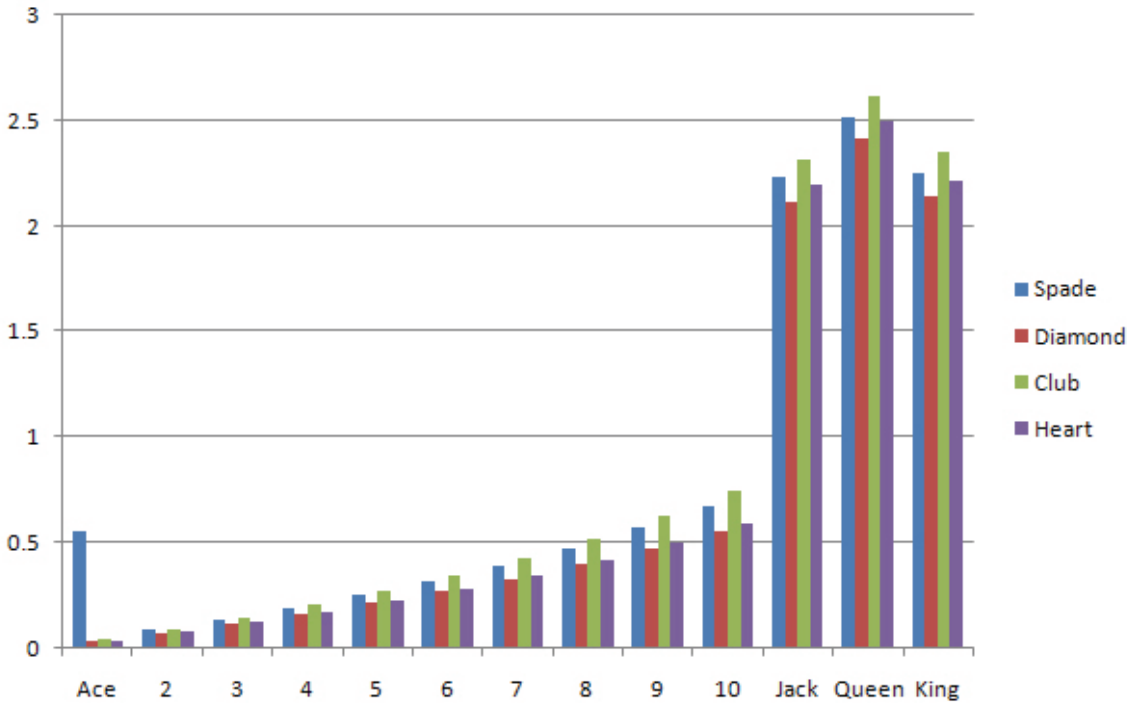

Figure 4.12: Normalized black to white ratio

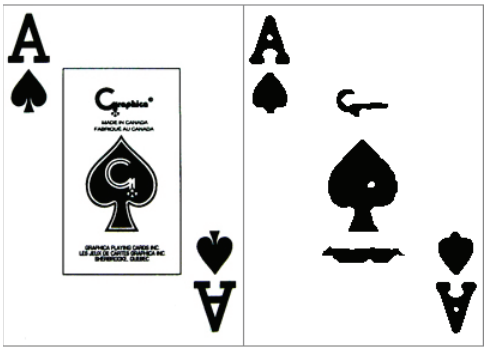

Figure 4.13: Actual ace of spade (left) and thresholded version (right)

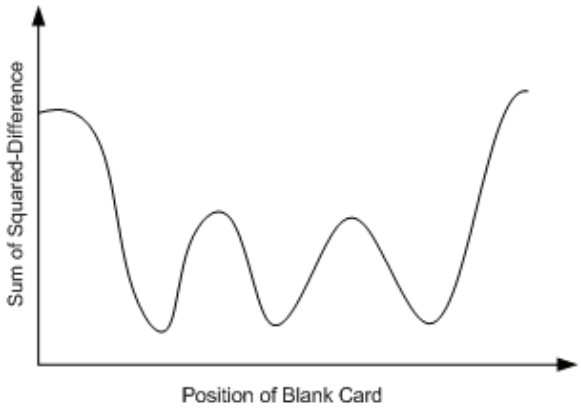

Figure 4.14: Sum of squared-difference of the blank card over an observable area with three cards

to the player, the dealer's observable area is constantly moving as the dealer draws cards. Again, the logical flow of the game of Blackjack helps to overcome this problem. First, the dealer's observable area is not probed until all players received their first card. Second, given that motion appears and stops in the dealer's observable area, the sum of squared-difference can be applied between the key frame (before motion starts) and the capture frame (when motion stops). That way, if a card is placed in the observable area, the sum of squared-difference will increase significantly.

4.3 Identification of the Players' Second and Subsequent Cards

The method to identify the second and subsequent cards dealt to a player is essentially the same as for the first card. However, the blank card matching method must be enhanced. Figure 4.14 shows that more than one sum of squared-difference extrema can appear. In the given example, there are three sum of squared-difference minimums because three cards were in the observable area. The latest dealt card, the one to identify, is always the first minimum value given that blank card is swept as pictured in Figure 4.8.

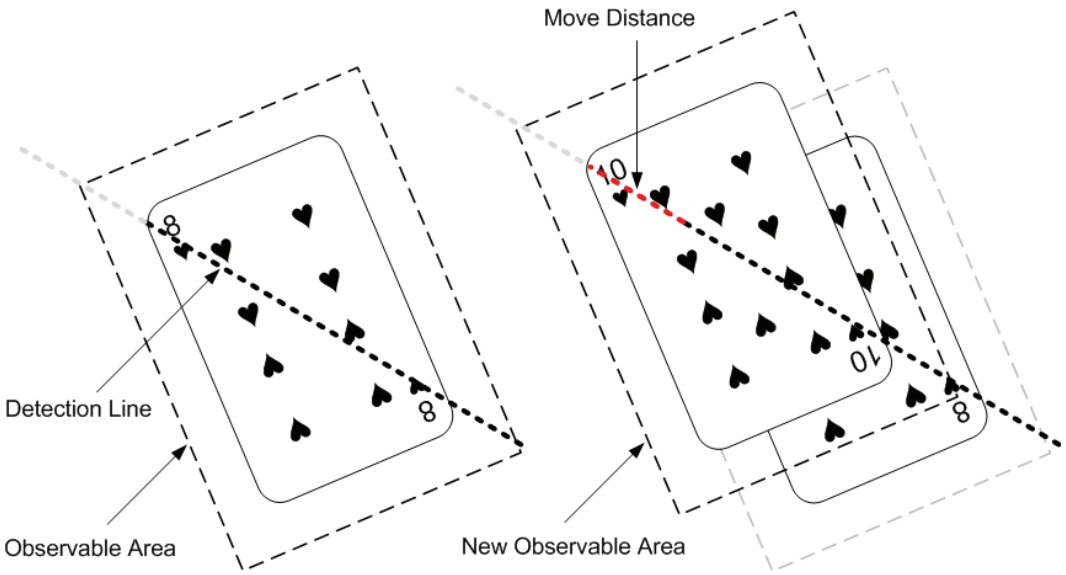

Figure 4.15: Displacement of the observable area along the detection line

Theoretically, a player can be dealt as many as eleven cards. Although this situation is highly improbable, the system must be able to handle such a situation. In addition, to be useful, the observable area around the cards must be kept as small as possible. This implies that the observable area must be moved when cards are added. To do this, the detection line is used. As figure 4.15 illustrates, because the detection line is laid in the direction that the cards are dealt, the observable area is moved along this line. The distance by which the observable area must be moved corresponds to the length difference of the detection line between two consecutive cards (red dashed line).

4.4 Identification of the Dealer's Second and Subsequent Cards

When the dealer's second card is dealt, it is put face down under the first card. This means that at the moment it is initially dealt, no action must be taken by the

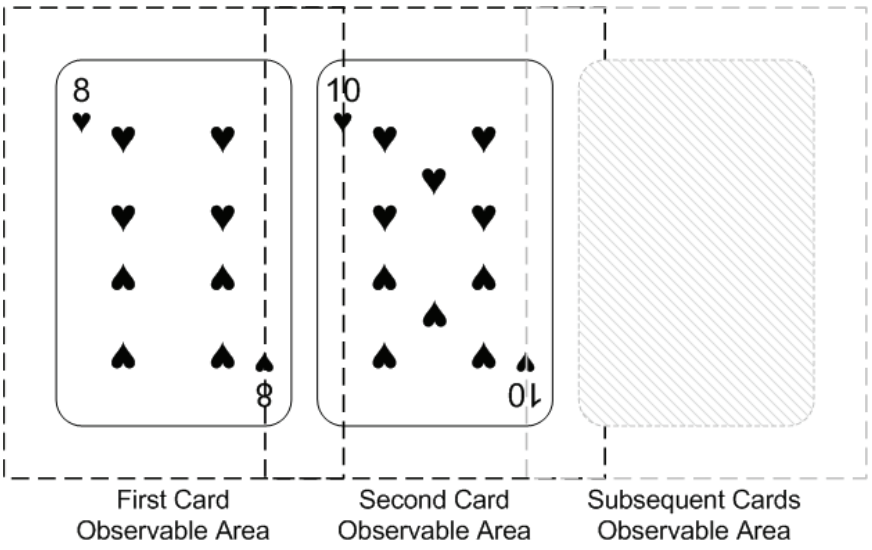

Figure 4.16: Dealer's cards and observables areas

card tracking system. Since the first card has already been identified, the first card's observable area is not probed. The detection of the second and subsequent cards is easier than the players' cards because all the cards are put side by side on the table as illustrated in Figure 4.16. Therefore, each card the dealer deals to himself can have its own observable area. The dealer's second second card will only appear in its observable area when every player has stood or busted. The remainder of the card identification process is the same as described above except that no rotations are necessary because the dealer's observable areas make a right angle with the field of view. This makes the identification of the dealer's cards the least computationally expensive part of the process.

Chapter 5

Hardware Implementation of the System

The hardware implementation of the card tracking system is achieved in a Digilent FPGA development board (Digilent Inc., 2007). The development board is powered by a Xilinx Virtex-II Pro XC2VP30 FPGA device that provides 30,816 logic cells. An analog video decoding board is coupled to the FPGA development board to provide video data to the FPGA device. The video decoding board is powered by an Analog Devices AD7183B integrated circuit.

Three major components of the system are implemented in the FPGA device: a video frame grabber, a video processor and a console. The video frame grabber receives the video data from the camera and outputs video frames in a buffer that is read by the video processor. In addition, the video frame grabber provides the content of the buffer to an external video DAC to provide the ability to monitor the video from the camera. The video processor analyzes each frame from the video buffer and applies the algorithm described in the preceding chapter. Results of the real-time algorithm computation are reported to a console. Communication to the console of

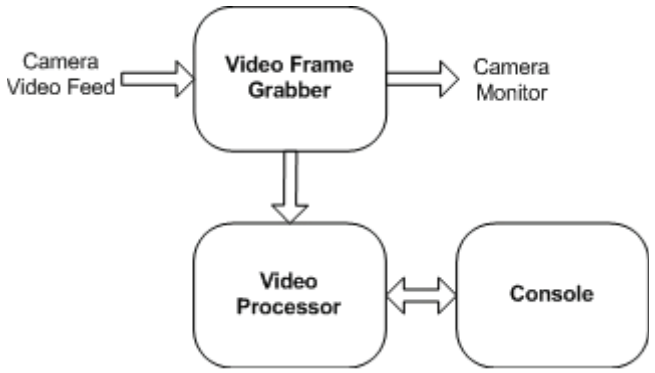

Figure 5.1: Hardware Implementation of System Overview

the system is possible using a RS-232 serial link from a PC host. The communication link accesses the system's console to monitor alerts and tracking status.

Figure 5.1 shows the general block diagram of the hardware implementation.

5.1 Video Frame Grabber

The task of the video frame grabber is to provide video data to the video processor. To do this, the video frame grabber accepts an analog video feed that is converted to ITU-R BT.601/656 digital video format by an Analog Devices ADV7183B video decoder. The video field and line timing signals are extracted from the ITU-R BT.601/656 digital video format. A conversion from YCbCr 4:2:2 to YCbCr 4:4:4 is done prior to have the digital video ultimately formatted to RGB colour space. Decoded RGB video data is buffered in the FPGA's BRAM to serve as a source for the video processor. Data is stored line by line in a double-depth line buffer, meaning that while a line is being stored the previous line is being read. Once a complete frame is acquired, the data in transfered to a frame buffer. The video data is ultimately provided to the video processor and is also presented to a Fairchild FMS3818 video DAC from the BRAM for display on a VGA monitor. In this regard, video timing signals are generated and provided to the video DAC monitor and VGA monitor for proper image

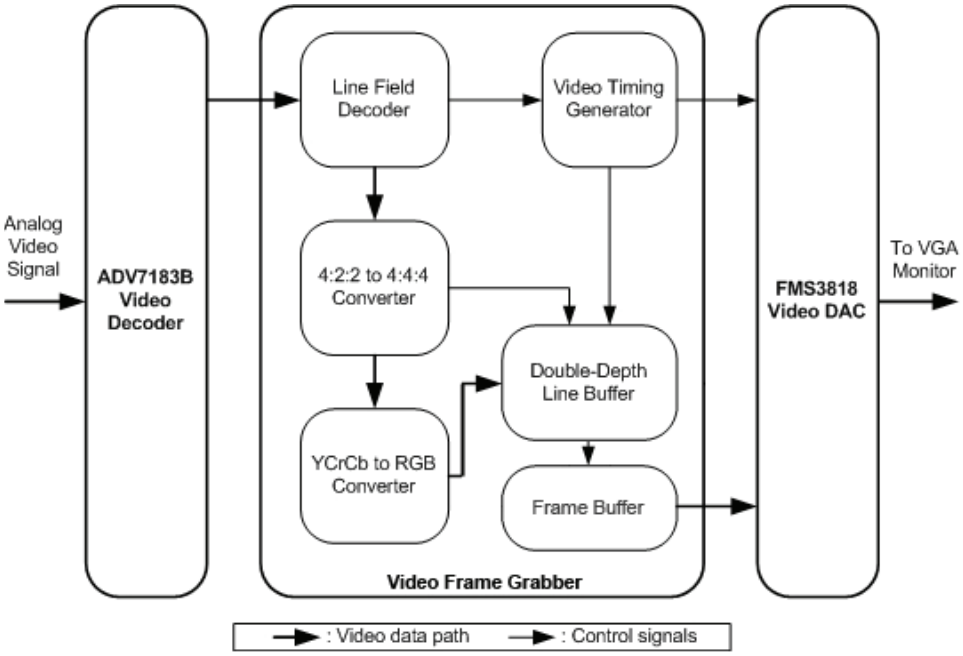

Figure 5.2: Video Data Grabber Block Diagram

synchronization.

Figure 5.2 shows the general block diagram of the video data grabber (detailed schematic shown in appendix). Globally, the system takes a NTSC, PAL or SECAM analog video input signal and outputs a video signal for a VGA monitor.

5.1.1 Background

In the process of taking an analog video signal and recovering each line in digital buffer, many video standards and concepts are involved. This section surveys these concepts in order to facilitate the understanding of the current implementation.

ITU-R BT.601 ITU-R BT.601, previously known as CCIR 601, is a standard for encoding interlaced analog video signal in a digital format. It includes both methods of encoding 525 lines at 60 Hz (NTSC) and 625 lines at 50 Hz (PAL). In the cur-

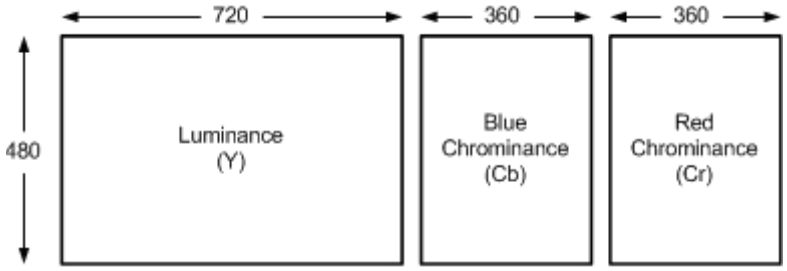

Figure 5.3: YCbCr 4:2:2

rent implementation we only consider a NTSC signal. Each line has luminance and chrominance samples. Luminance is the weighted sum of the linear RGB components of a colour video signal, proportional to intensity. Chrominance defines the difference between a colour and a chosen reference colour of the same luminous intensity. The principal interest in transmitting image information in luminance and chrominance rather than RGB (red, green and blue) colour space was to keep backward compatibility with monochrome receivers that were incompatible with RGB signals.

4:2:2 and 4:4:4 Colour Ratios In ITU-R BT.601, the colour encoding system is known as YCrCb 4:2:2, that being the ratio of Y:Cb:Cr where Y is the luminance data, Cb the blue chrominance data and Cr the red chrominance data. In the 4:2:2 format, there is twice as much luminance data than blue and red chrominance data. In the 4:4:4 format, there is no loss in the blue and red chrominances and all components have an equal number of pixels.

Frame Size In ITU-R BT.601, a frame is defined by 720 pixels per line and 480 lines per frame. Figure 5.3 illustrates the frame size and YCbCr correlation.

Serial Digital Interface The serial digital interface (SDI), standardized in ITU-R BT.656 (and SMPTE-259M) is a digitized video interface used for broadcast grade video. This standard is used for transmission of uncompressed, unencrypted digital

Sample n				*Sample n+1*			
Cb	Y	Cr	Y'	Cb	Y	Cr	Y'

Figure 5.4: Sample datastream

television signals within television facilities. It is designed for operation over short distances because of its high bitrates. The most common bitrates, although other bitrates exist, is 270 Mb/s and is the one we use in the current design. Figure 5.4 shows how the datastream is arranged.

As described above, the video is encoded in 4:2:2 format, that is when the luminance is encoded at full bandwidth (13.5 MHz) and the two chrominance channels are subsampled horizontally and encoded at half bandwidth (6.75 MHz). The Y, Cr, and Cb samples are acquired at the same instance in time and the Y' sample is acquired at the time halfway between two adjacent Y samples.

There are two word formats for the video data: the 10-bit studio format and the 8-bit consumer format. For the 10-bit studio format, video payload may use any 10-bit word in the range 4 to 1019 inclusively. The values 0-3 and 1020-1023 are reserved and may not appear anywhere in the payload. These reserved words serve for synchronization packets and for additional data headers. In the 8-bit consumer format, the two least significant bits are ignored but the reserved words remain the same. The presented design only supports the 8-bit consumer format although the data path has been kept at 10-bit wide for further support of the studio format. More details regarding the Serial Digital Interface are given in the implementation section.

5.1.2 Implementation

This section describes the implementation of each block constituting the design. Each block description is accompanied with a specific block diagram that highlights relevant signals.

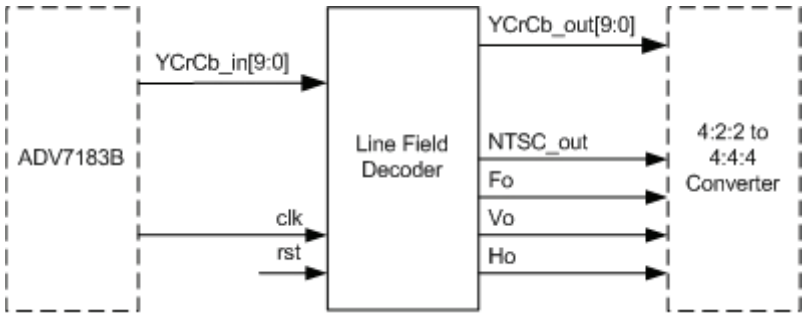

Figure 5.5: Line Field Decoder Block Diagram

Line Field Decoder The purpose of the Line Field Decoder block is to retrieve all the video field and line timing embedded in the ITU-R BT.656 standard. This is done by decoding non-video related data values.

Figure 5.5 is the corresponding block diagram. The behavioural description of the internal circuitry follows.

In the video data, the reserved 8-bit values of 0x00 and 0xFF are used to signal a Timing Reference Symbol (TRS). When the TRS sequence 0xFF 0x00 0x00 occurs, and this occurs at every line, a special word referenced as XY always follows. This sequence is the field ID pattern. Decoding the XY word in relation with the various counts, such as line count, completely specify the NTSC timing. The bit 4 of XY indicates a start of active video (SAV) if high and an end of active video (EAV) if low. Active pixels follow the SAV while the horizontal blanking follows EAV. The bits 5 and 6 are respectively the V and F bits. The V bit denotes if a field blanking is present (V bit is high) or not (V bit is low). The F bit denotes if an odd field (F bit is low) or an even field (F bit is high) is occurring. Figure 5.6 illustrates a waveform relating the TRS, XY, SAV and EAV terms.

In terms of implementation, the Line Field Decoder block has therefore the function, for each line, to find the field ID pattern 0xFF 0x00 0x00 0xXY and then to latch XY. The XY word will therefore indicate what field is currently active and whether or not field blanking is active. In addition, if the XY word denotes a SAV, then we

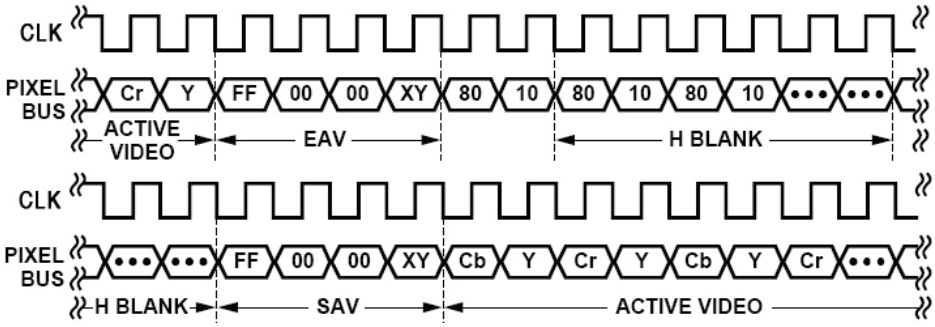

Figure 5.6: Video Input Waveform

know we are at pixel 0. The Fo, Vo and Ho signals are therefore generated that way.

To identify the format as NTSC or as PAL and therefore generate the NTSC_out signal, we look for a falling edge on the V bit and then count the number of SAV terms until the next V bit rising edge. NTSC has 19 lines during this period and PAL has 24.

The output YCrCb_out corresponds to the input YCrCb_in delayed by five clock cycles, which corresponds to the pipeline delay of the block. That is to have the video data synchronized with the Fo, Vo and Ho signals generated by the Line Field Decoder.

4:2:2 to 4:4:4 Converter The 4:2:2 to 4:4:4 Converter creates the missing Cr and Cb components. This is done by duplicating the Cr and Cb components.

Figure 5.7 is the corresponding block diagram. The behavioural description of the internal circuitry follows.

To efficiently duplicate the values of Cr and Cb, a pipeline for each component (Y, Cr and Cb) is created and video data is input to its corresponding pipeline. The Y-pipeline has a depth of four and the Cr- and Cb-pipeline have a depth of three. This allows for the propagation of a complete cycle of Cb Y Cr Y'. The Y, Cr and

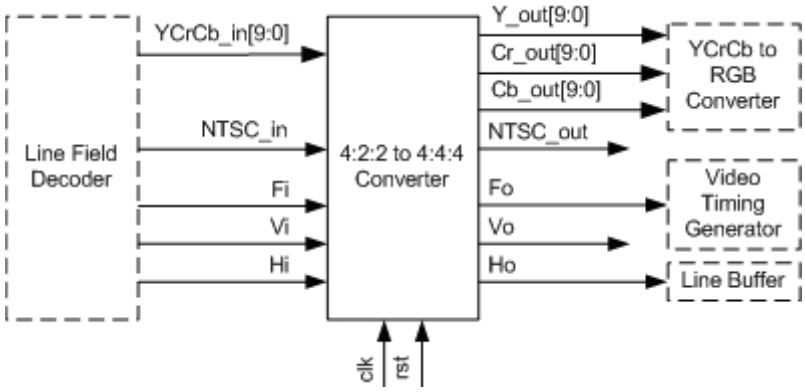

Figure 5.7: 4:2:2 to 4:4:4 Converter Block Diagram

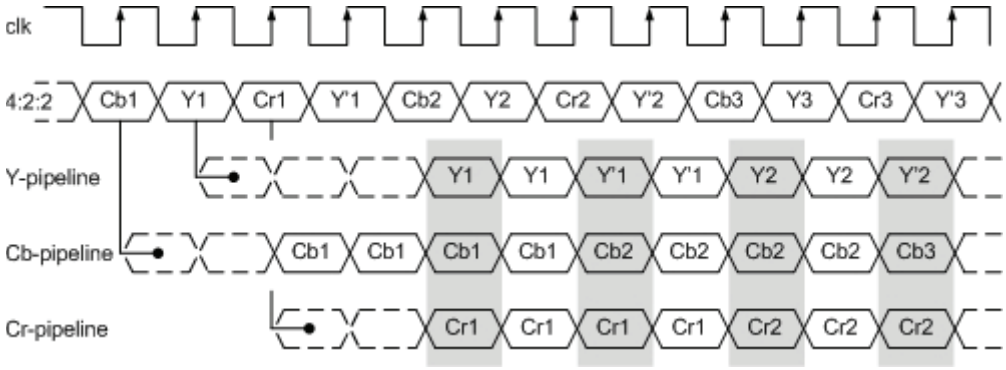

Figure 5.8: 4:2:2 to 4:4:4 Converter Pipelines

Cb outputs of the 4:2:2 to 4:4:4 are the last stage of each pipeline. This results in the correct interpolation of Y, Cr and Cb components at each clock cycle. Note that the next block operates at half the frequency of the 4:2:2 to 4:4:4 Converter and therefore the samples appearing twice are dropped. Figure 5.8 illustrates the pipelines constituting the 4:2:2 to 4:4:4 Converter block.

The outputs NTSC_out, Fo, Vo and Ho correspond to the inputs NTSC_in, Fi, Vi and Hi delayed by four clock cycles, which match to the pipeline delay of the block. That is to have the video data synchronized with the Fo, Vo and Ho signals generated by the 4:2:2 to 4:4:4 Converter.

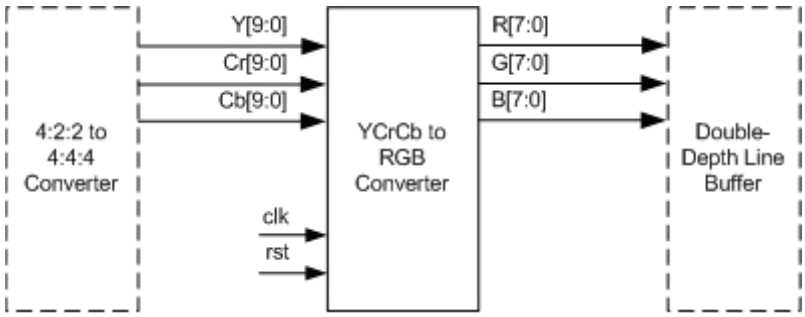

Figure 5.9: YCrCb to RGB Converter Block Diagram

YCrCb to RGB Converter The YCrCb to RGB Converter transforms the Y, Cr and Cb video components into RGB colour space. This is done by applying weighted sums to the Y, Cr and Cb components. For the 8-bit consumer format, the weighted sums are the following:

R = 1.164(Y - 16) + 2.017(Cr - 128)

G = 1.164(Y - 16) - 0.813(Cb - 128) - 0.392(Cr - 128)

B = 1.164(Y - 16) + 1.596(Cb - 128)

For the 10-bit studio format, the one we considered in the current design, the weighted sums are the following:

R = 1.164(Y - 64) + 2.017(Cr - 512)

G = 1.164(Y - 64) - 0.813(Cb - 512) - 0.392(Cr - 512)

B = 1.164(Y - 64) + 1.596(Cb - 512)

Figure 5.9 is the corresponding block diagram. The behavioural description of the internal circuitry follows.

In the implementation, the range of valid values for the Y, Cr and Cb components were respectively limited to [64, 940], [64, 960] and [64, 960]. The weighted sums were

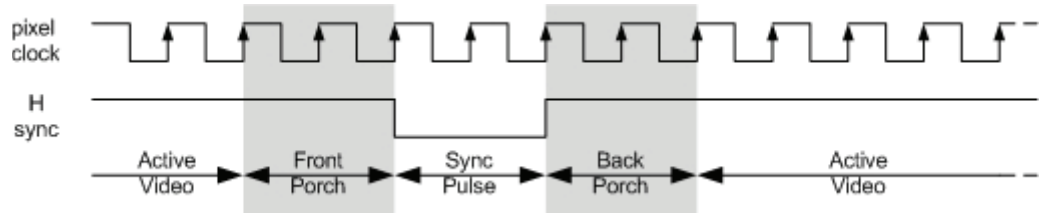

Figure 5.10: Horizontal Synchronization

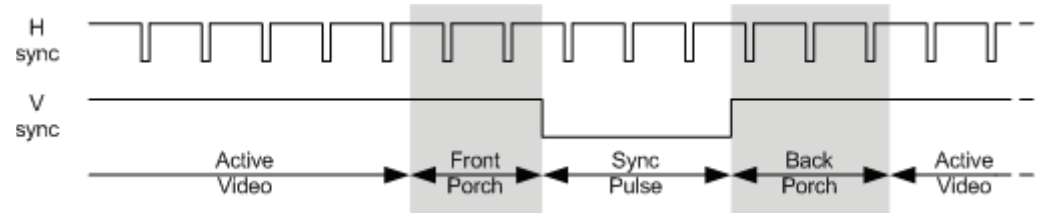

Figure 5.11: Vertical Synchronization

computed using fixed-point representation.

Video Timing Generator In order to display the video data contained in the Double-Depth Line Buffer onto a VGA monitor, the Video Timing Generator needs to generate the horizontal and vertical synchronization pulses. These signals are derived from the field signal (Fo), indicating the beginning of an odd field. Here we assume interlaced video and therefore make abstraction of the vertical blank signal Vo. As their names imply, the horizontal synchronization pulse indicates the beginning of a new line while the vertical synchronization pulse indicates the beginning of a new frame. Not all the pixels in a line are displayed as "active video" because of the horizontal blanking. The blanking period before and after the horizontal synchronization pulse are respectively referred to as the horizontal front and back porches. Similarly, not all the lines in a frame are displayed as "active video" because of the vertical blanking. The blanking period before and after the vertical synchronization pulse are respectively referred to as the vertical front and back porches.

The display format used in the current design is 720 pixels by 480 pixels at 60 Hz with a 27 MHz pixel clock. Figure 5.12 illustrates the resulting parameters.

Horizontal (in pixels)				*Vertical* (in lines)			
Active Video	Front Porch	Sync Pulse	Back Porch	Active Video	Front Porch	Sync Pulse	Back Porch
720	7	62	69	480	4	4	30
Total = 858				Total = 525			

Figure 5.12: Horizontal and Vertical Parameters

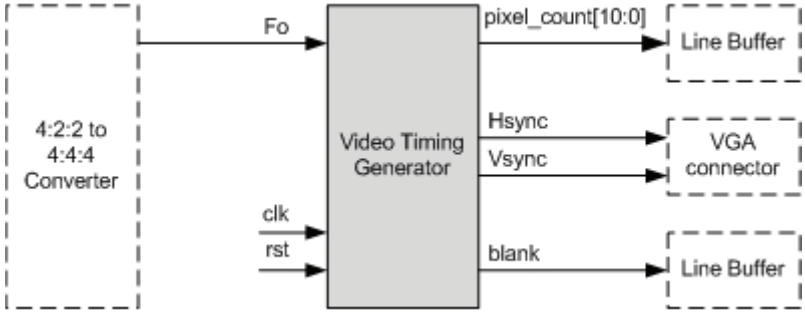

Figure 5.13: Video Timing Generator Block Diagram

The Video Timing Generator also produces a "pixel count" that indicates to the line buffer which pixel from the current line to feed into the video DAC. Finally, the horizontal and vertical blanking signals are generated and fed to the video DAC in order to have the DAC output driven to blanking level during blanking periods.

Figure 5.13 is the corresponding block diagram. The behavioural description of the internal circuitry follows.

On a falling edge of Fo, the Video Timing Generator resets a pixel counter and a line counter on which all the outputs are based. The pixel counter is incremented by the pixel clock and is self-reset when the count reaches the total number of pixels in a line (858 with the given display size). Similarly, the line counter is incremented each time the pixel counter reaches its maximum value and is self-reset when the count reaches the total number of lines in a frame (525 with the given display size). The horizontal synchronization pulse and blanking signal are asserted according to the current pixel count in relation with the horizontal parameters (see Figure 5.10 and Figure 5.12). Similarly, the vertical synchronization pulse and blanking signal are asserted according to the current line count in relation with the vertical parameters

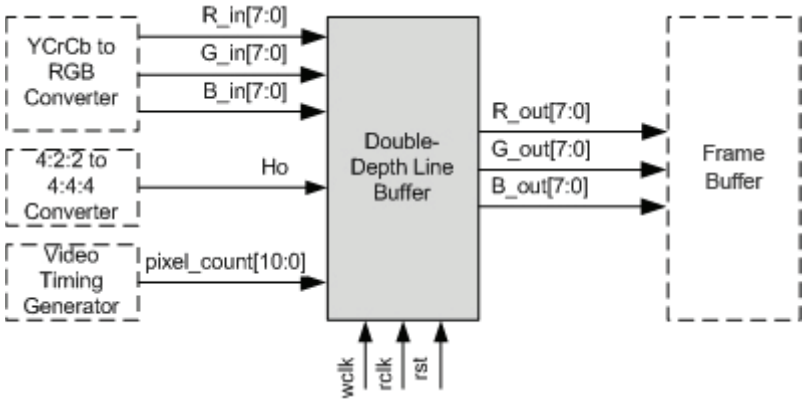

Figure 5.14: Double-Depth Line Buffer Block Diagram

(see Figure 5.11 and Figure 5.12).

Double-Depth Line Buffer and Frame Buffer The Double-Depth Line Buffer stores the current line of RGB video data while previous line is presented to the Frame Buffer. The R, G and B output values are fed into a triple 8-bit video DAC in order to generate the suitable analog levels for these three components. The R, G and B outputs of the video DAC are fed directly to a VGA monitor by the means of a VGA connector.

Figure 5.14 is the corresponding block diagram. The behavioral description of the internal circuitry follows.

Each individual line buffer is implemented using BRAM. The toggling between one line and the other as the source and destination for read and written data occurs at each rising edge of the Ho signal. The read and write operation have two distinct, yet, in-phase clocks. The read address is the pixel_count value from the Video Timing Generator. The write address is an internal counter reset by Ho and incremented on each write clock cycle. In the current design, the data is read at 27 MHz while the data is written at 13.5 MHz.

Each line registered in the line buffer is transfered to the frame buffer. The frame

buffer, implemented using DRAM, is filled using two distinct pointers. A first pointer, the line index pointer, circulates from 0 to 479 and addresses each line to be stored. The line index pointer is added to the second pointer, the frame index pointer, which is incremented once each frame is completely stored. Given that the development board is equipped with 512MB of DRAM and that each frame occupies a little less than 1MB (480px * 720px * 3B/px = 1013kB), a maximum of 512 frames can be contained at any moment in the memory.

5.2 Video Processor

The task of the video processor is to take the video feed provided by the frame grabber, analyze its content, and return the players' and dealer's hands to the console after each game.

The video processor takes full advantage of the hardware parallelism allowed by the FPGA implementation (Sonka *et al.*, 1999). The video processor is implemented into three elements that operate independently: a finite state machine, an activity detection engine and a card identification engine. First, the finite state machine controls the flow of operation and ultimately returns players' and dealer's hands after each game. Second, the activity detection engine signals the finite state machine if motion is detected in one of the seven player spots. Finally, the card identification engine applies card tracking specific algorithms and returns the card value to the finite state machine following analysis of the provided frames.

5.2.1 Finite State Machine

Based on the fact that the game follows a predetermined flow, the video processor is governed by a finite state machine. The finite state machine follows the same states

as the ones described in the preceding chapter and are illustrated in Figure 5.15. For each state, the card identification engine applies a specific portion of the card tracking algorithm and returns a card value. Once the finite state machine reaches its final state, the hands of the dealer and all the players have been acquired and this information is returned to the console for validation of the current game.

The finite state machine is implemented using the Mealy model (Sipser, 2006). The system will stay in the initial state as long as no activity is signaled in the first player's spot, a task accomplished in parallel by the activity detection engine. Once the finite state machine sees the end of a burst of activity, meaning the card has been placed in the player's spot, it will enter the first card identification state. Before proceeding to the next player in the same fashion, a card value must have been returned by the card identification engine. Once all the players' and the dealer's first cards have been identified, the second card identification state is entered and a card value is again expected from the card identification engine for each player. Note that the dealer's second and subsequent cards identification only comes at the end of the game when all players have stand or busted.

Once all players' two first cards have been identified, the finite state machine proceeds to the subsequent identification states. This is done player by player until each player stands or busts. The condition that determines if a player has stand is the detection of movement in the next player's spot, again signaled by the activity detection engine. Once activity is detected in the next player's spot, the system assumes the current player has stand and moves to the next player. Note that if the player busts, a condition similar to a stand in terms of game flow, the finite state machine also steps to the next player. This condition is self-detected when the running count of the identified cards is above 21.

Exiting the state of cards identification of last player's constitutes a special case because the next player is the dealer. In order to step to the dealer's cards iden-

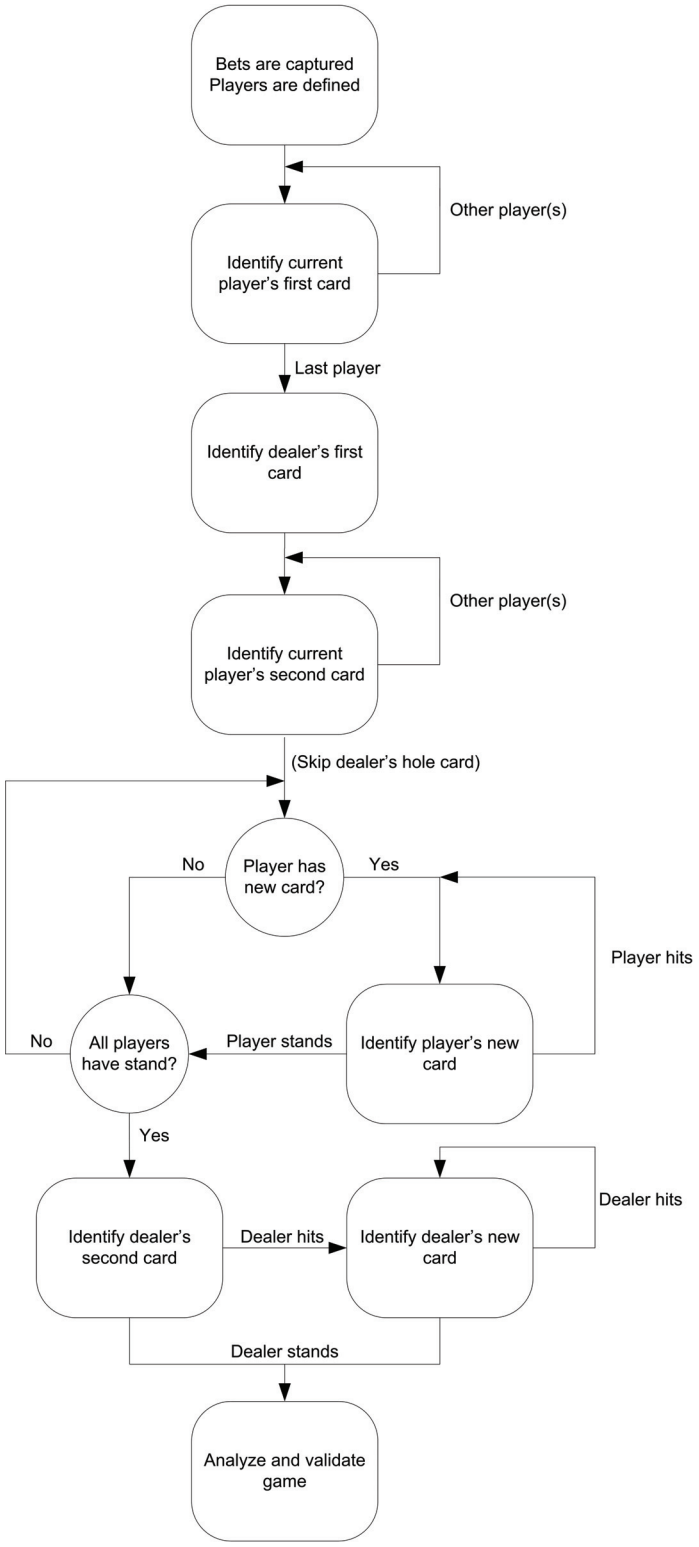

Figure 5.15: Flow of a Game of Blackjack

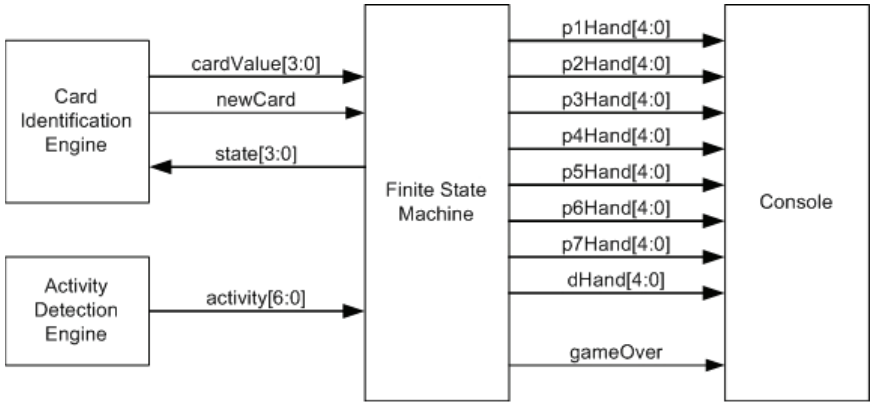

Figure 5.16: Finite State Machine Signals

tification states, the actual activity detection cannot be used for the dealer's spot since the dealer constantly moves his arm above this region. Instead, assuming that under normal conditions every player hits in less than five seconds, a five seconds time of non-activity on the current spot is used to signal that the last player has stand. This function is embedded in the activity detection engine. Following this condition, the finite state machine steps to the identification of the dealer's cards. In this final state of cards identification, card values are expected until the predefined dealer stand condition is achieved (usually 17 or more).

At this point, the hands of the dealer and every player have been collected and this information is returned to the console. Figure 5.16 illustrates the signals affecting the finite state machine.

The card identification engine is governed by the *state[3:0]* bus that dictates which card identification algorithm to apply given the current state of the finite state machine. Card values are returned to the finite state machine by the *cardValue[3:0]* bus. When a new card is identified, the *newCard* signal is asserted for one clock period.

The activity detection engine provides one activity line per player spot creating the *activity[7:0]* bus. Corresponding lines are asserted when activity is detected on the corresponding player's or dealer's spot.

The console is provided with eight 5-bit hand value buses: *p*Hand[4:0]* for player one to seven and *dHand[4:0]* for the dealer. Returned values represent the summation of all cards dealt to the player. These values can theoretically range from one to 31 but only the values from two to 30 can be found real-life scenarios. A returned value of zero is a special case that indicates that the spot is not being used. When a game has been completed and all the hands are available for validation, the *gameOver* signal is asserted for one clock period.

5.2.2 Activity Detection Engine

The detection of activity in each player's and dealer's spot is the most important condition that enables the finite state machine to step throughout the various states described above. If activity is detected in the player's or dealer's spot, the corresponding activity line will be asserted as long as activity is present.

Activity Detection The regions where activity is tracked are the detection lines described in Chapter 4 and their coordinates are defined in the user configurable register map. At each time a frame is written to the DRAM, each of the eight detection lines are copied to dedicated block RAM buffers (BRAM) for independent analysis. There is in fact two BRAM's per player's or dealer's spot, one containing the current detection line and the other one containing the key detection line, that is the initial detection line captured at the very beginning of the state, a moment known to show the player's spot without any activity. Because 24-bit RGB color is more information than necessary for motion detection (color information is unimportant to motion detection as long as the image is clear) and that converting each line to gray-scale or black-and-white would be a waste of processing power and/or hardware, only the 8-bit green channel is copied to the BRAM. At this regard, Figure 5.17 illustrates a typical game table viewed by either 8-bit red, green or blue channels. In order to

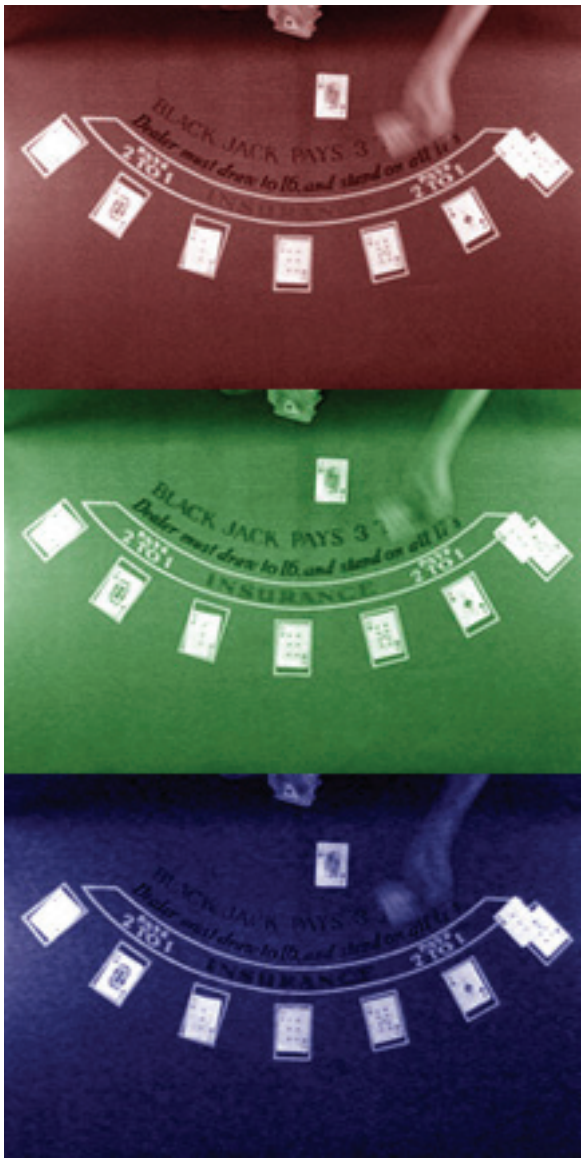

Figure 5.17: From top to bottom: 8-bit red frame, 8-bit green frame and 8-bit blue frame of typical game table

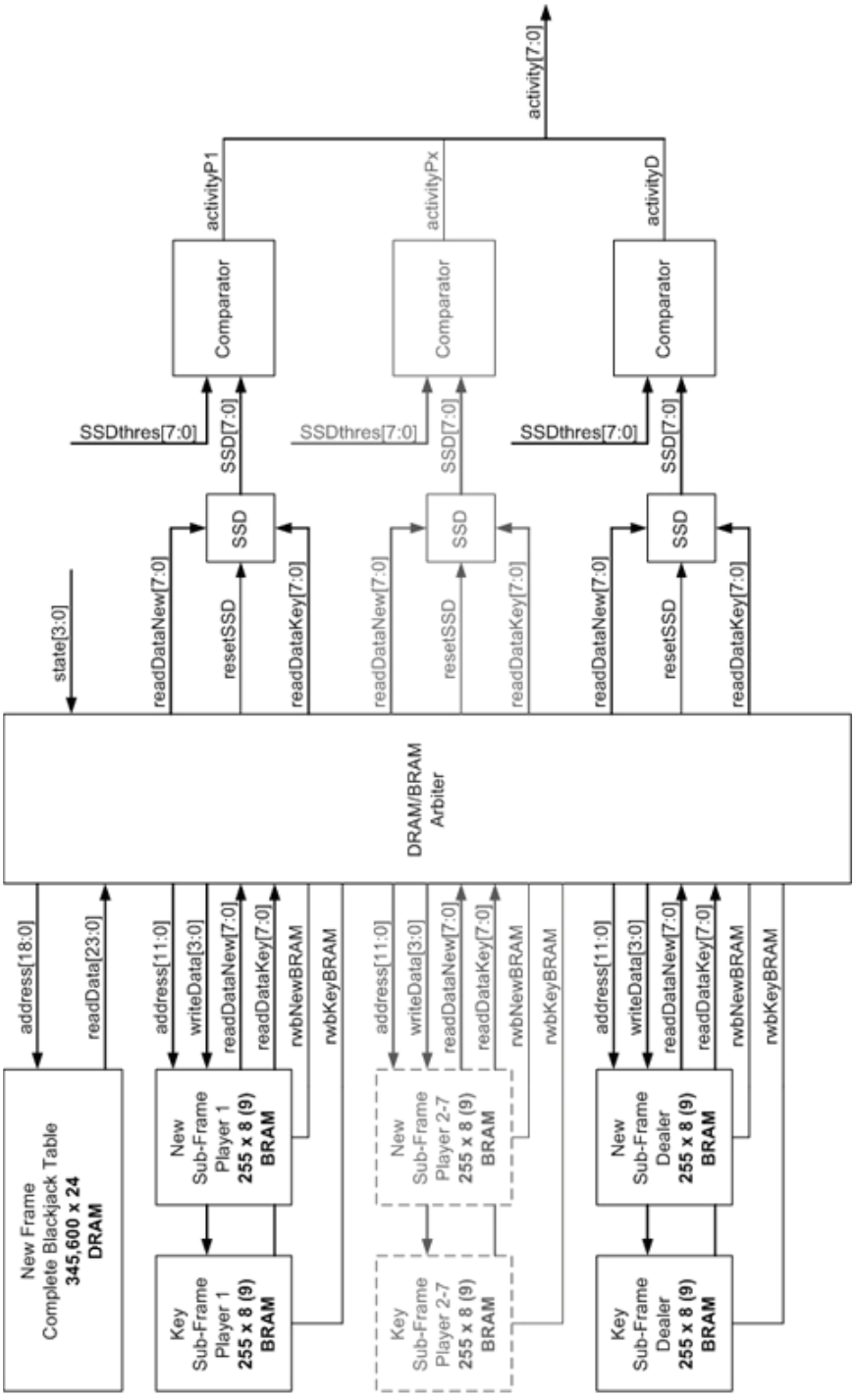

Figure 5.18: DRAM/BRAM arbiter with SSD circuitry

keep the hardware consistent for the 8 players and dealer, the activity line BRAM's are 255 pixels each. The actual detection lines' length is set in the configurable register map to the longest line possible without having the players' detection lines interfere with the dealer's spot. The detection line BRAM's are therefore organized as 250 x 9 bits where one bit is wasted by pixel. This restriction is imposed by the predefined Xilinx FPGA block sizes: 16K x 1 bit, 8K x 2 bits, 4K x 4 bits, 2K x 9 bits, 1K x 18 bits or 512 x 36 bits.

Figure 5.18 illustrates the DRAM/BRAM arbiter and the motion detection circuitry, that is the sum of square differences (SSD) block. Each time the finite state machine enters a new state, which is signaled by the *state[3:0]* bus, the eight key detection line BRAM's are updated. Each time a new frame is written into the DRAM, the new detection line BRAM's are updated and the SSD is computed on each set of key and new detection lines. The DRAM-BRAM arbiter then outputs the content of the key and new detection line BRAM's, pixel by pixel, to the SSD block that performs the subtraction and accumulates the square of the difference. The sum is reset each time a new detection line is written. When the SSD value exceeds the threshold set by the *SSDthres[7:0]* value in the configurable register map, the activity line for the corresponding player's or dealer's spot is asserted. The activity line is asserted for a predefined delay, about 2 seconds, long enough for the dealer to drop the card and take his hand out of the spot.

Displacement of Observable Regions As described in section 4.3, in order to identify the next card, the observable region coordinates must be re-centered around this new card. The direction of the displacement is made along the detection line. The distance by which the observable region must be displaced is defined by the number of pixels with high SSD values between the key detection line and last captured detection line. Figure 5.19 illustrates the key and last captured detection lines (green line transposed to red box) for the first and second cards. The red portion of the

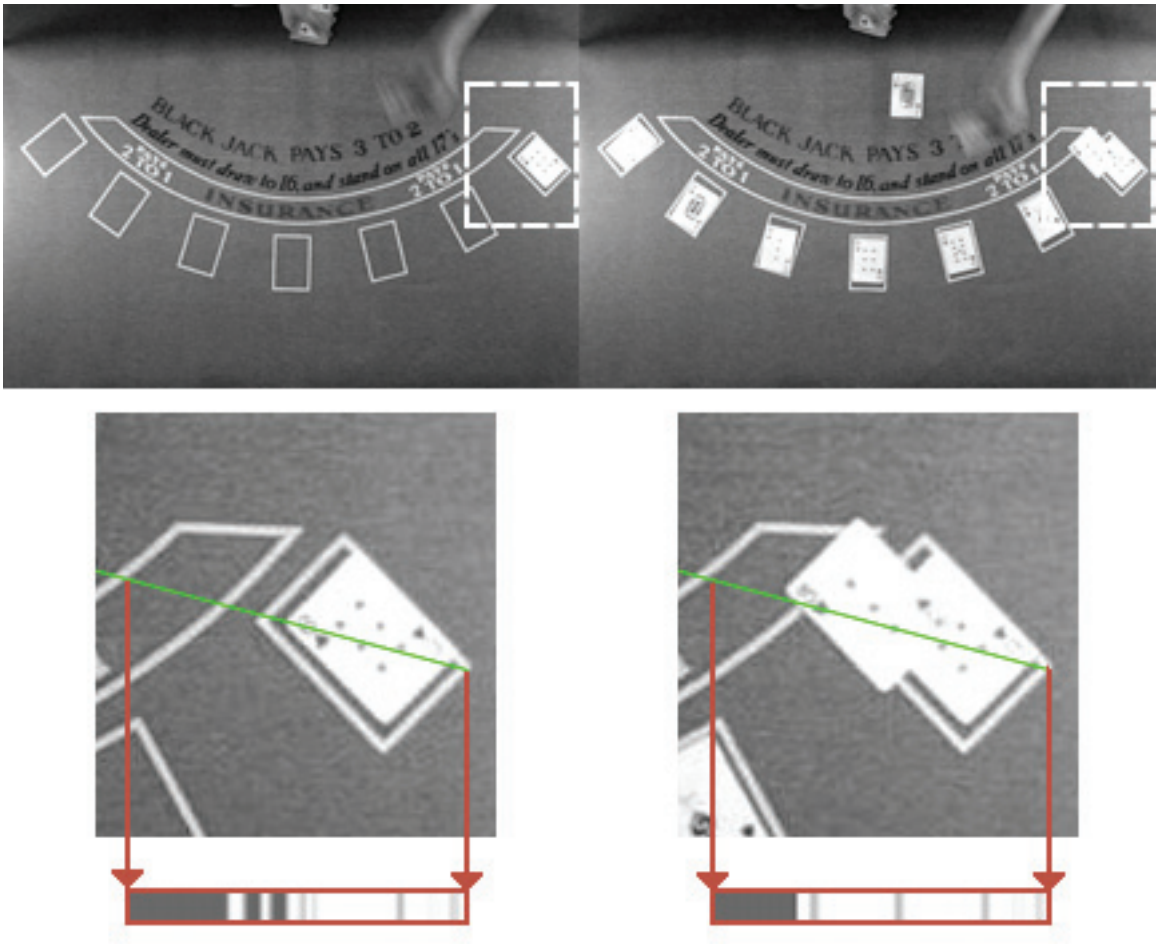

Figure 5.19: Example of key and last captured detection lines

detection line in Figure 5.20 illustrates how the SSD increases as the second card (shown in ghost) overlaps the first card.

The task of the hardware is therefore to identify how many pixels exceed a given threshold, set in the configurable register map, before the SSD value definitively returns to a low SSD value. With that number of pixels, the observable regions are then read from the register map, translated and then written back in preparation of the next card identification. The SSD tracker block accomplishes this and is repeated for each player and dealer.

Following a falling edge of the given activity line, indicating that the detection

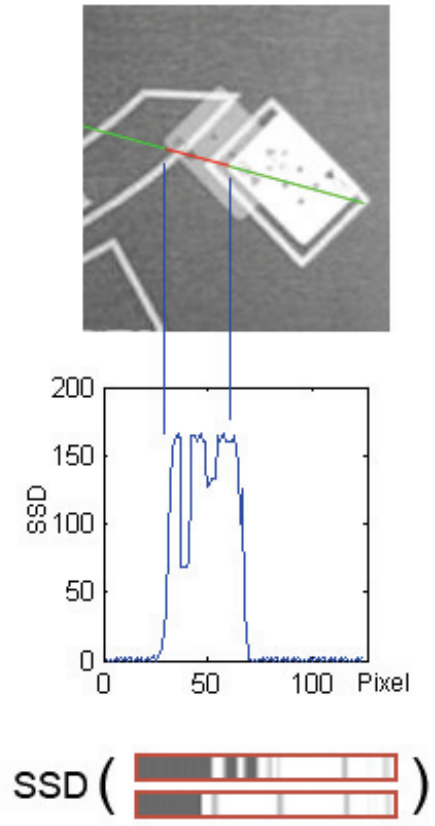

Figure 5.20: Correlation between detection lines and SSD values

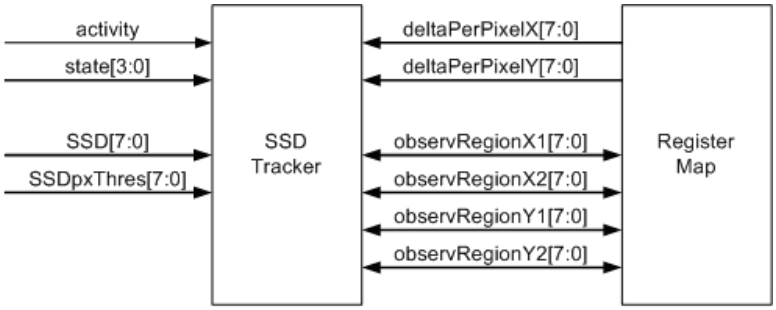

Figure 5.21: SSD tracker and observable regions update circuit

line BRAM's contain a suitable set of detection lines for comparison, the SSD tracker starts monitoring the SSD values. When the SSD values exceed the threshold set in the register map, *SSDpxThres[7:0]*, the SSD tracker starts counting the number of pixels computed by the SSD block. If the SSD value returns below the threshold, the pixel count is retained at that instant. However, if the SSD value rises and falls again, meaning that the latter occurrence was a false trigger and that pixel was still part of the second card, that new pixel count will be retained instead.

Figure 5.21 shows the interaction between the SSD tracker and the register map. When the pixel count is determined by the SSD tracker, an X and Y translation is applied based on the *deltaPerPixelX[7:0]* and *deltaPerPixelY[7:0]* values set in the register map. The operation done is the pixel count multiplied by the delta per pixel in X and Y added to the current observable regions coordinates yields the new observable regions coordinates. The translation geometry relative to the pixel count and the X and Y deltas is illustrated in Figure 5.22.

5.2.3 Card Identification Engine

Card Identification Sub-frame In order to proceed to the identification of each card of every player and the dealer, a single frame per card to be identified is used. When the finite state machine enters an identification state, which only happens

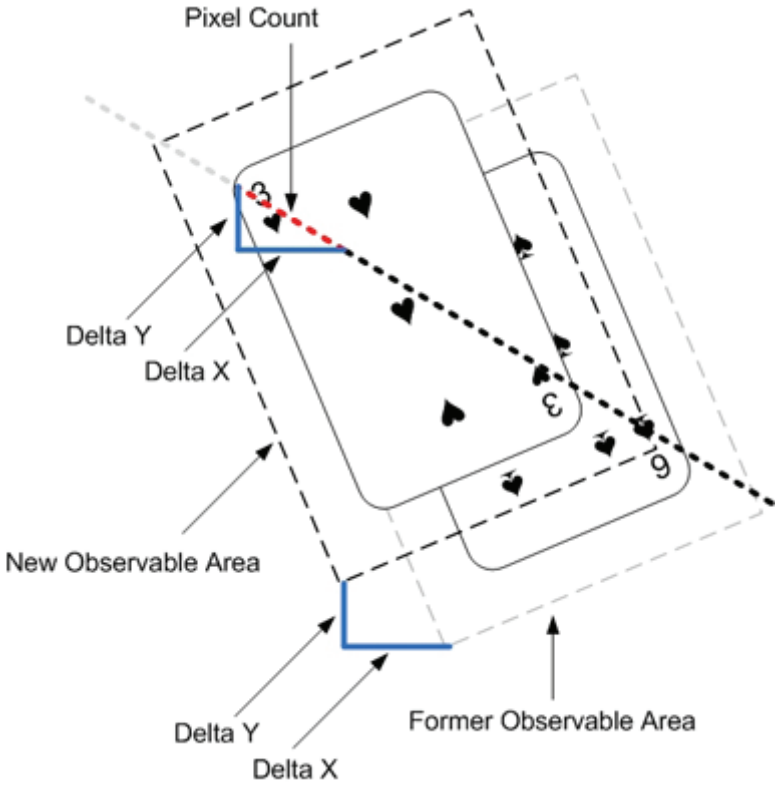

Figure 5.22: Displacement of the observable region based on the pixel counts

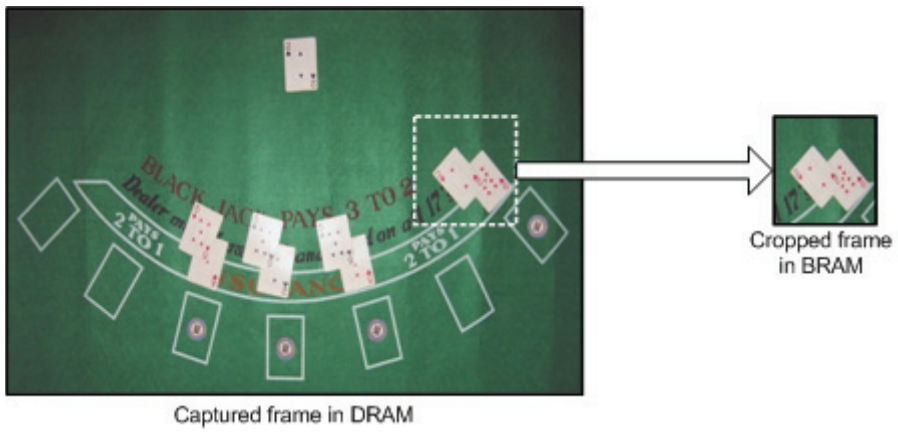

Figure 5.23: Example of DRAM frame cropped to BRAM

when the card has been dropped and is ready for identification, the card identification engine captures the frame at that exact moment. This frame, residing in the DRAM, is copied to a dedicated block RAM buffer (BRAM) in the card identification engine. The block ram is organized in 36-bit blocks where 12 bits are wasted per RGB pixel. As described for the activity detection engine, this restriction is imposed by the predefined block sizes. Because block RAM is a limited resource in the FPGA device (2,448 Kb or 136 x 18 Kb blocks), only the portion of the frame covering the current player's spot or dealer's spot (described as the enlarger observable area in the previous chapter) is copied. These areas are predefined parameters specific to each Blackjack table and are part of a register map configurable by the console. The current player's or dealer's spot to be copied is an information provided by the finite state machine by the *state* bus. Figure 5.23 illustrates the content of the DRAM and the BRAM on a specific example, here the identification of the second card of player one. Note that in order to keep the hardware consistent and reusable for every player's and dealer's spots, the area copied to the BRAM buffer is always the same size, that is 60 pixels by 60 pixels.

RGB to Black and White Conversion Having extracted the sub-frame containing the card to identify, the first video processing operation done is the conversion to

a black and white image. The first step to accomplish this is to convert every red, green and blue color quanta to a single gray value ranging from 0 to 255. This is done according to the NTSC standard (International Telecommunication Union, 1998) for which:

$$Gray = 0.2989 * Red + 0.5870 * Green + 0.1140 * Blue$$

Since floating point operations are computationally expensive (Sonka *et al.*, 1999), the red, green and blue coefficients are quantized to 8-bits which yield:

$$Gray = (2^{-2}+2^{-5}+2^{-6})*Red+(2^{-1}+2^{-4}+2^{-6}+2^{-7})*Green+(2^{-4}+2^{-5}+2^{-6})*Blue$$

$$Gray = 0.296875 * Red + 0.5859375 * Green + 0.109375 * Blue$$

Then to render a black and white image, all the gray pixels need to be thresholded. This is done by comparing the pixels to a predefined threshold value specific to each Blackjack table and stored in the register map, again configurable by the console. If the gray value is below the given threshold, the pixel is set to 1 and if the gray value is above the given threshold, the pixel is set to 0.

In terms of hardware, this RGB to gray-scale converter is implemented using a dedicated circuit illustrated in Figure 5.24. Having quantized the coefficients to powers of two, every division is done by taping the appropriate bits at the output of the BRAM and by padding the most significant bit as many times as needed at the left. This is the equivalent of accomplishing the division by right-shifting the bits. These 'divided' values, each corresponding to a factorized element of the quantized coefficients, are then fed into a 10-input, 8-bit adder to reproduce the equation above. The output of the adder is compared to the threshold value from the register map to yield the new black and white pixel. In order to process each pixel from the RGB BRAM, a counter is incremented and drives the address line of the RGB RAM. Every

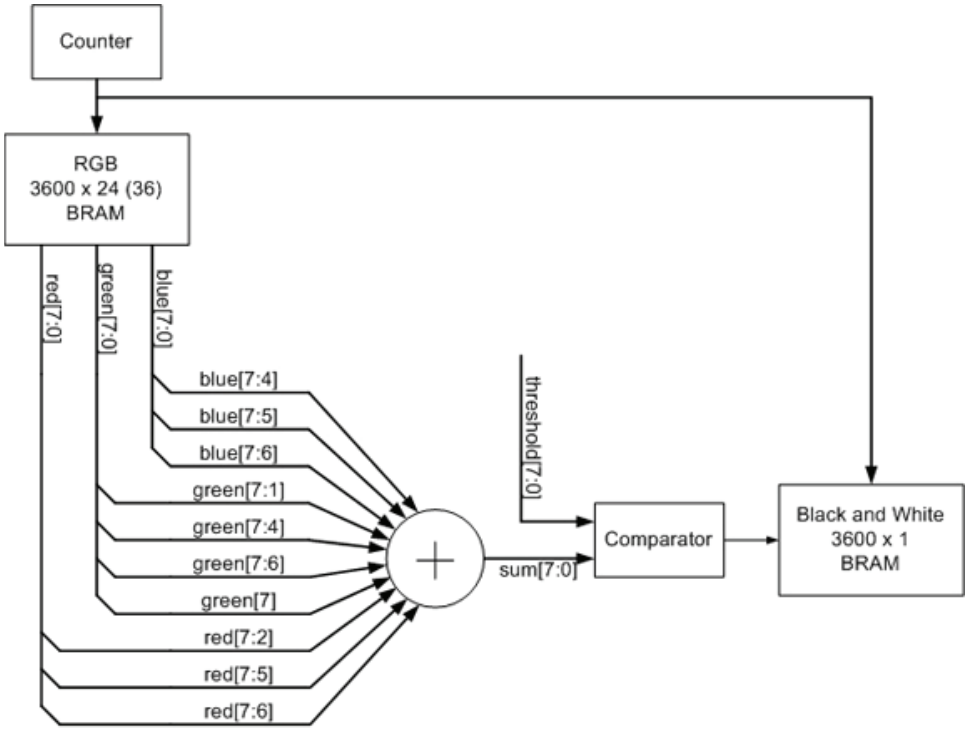

Figure 5.24: RGB to black and white conversion circuit

new pixel created is placed in another dedicated, but smaller, BRAM. This BRAM is organized as a single 16K x 1 bit block. The same counter is used to drive the address line of the black-and-white BRAM.

Card Isolation At this point, as illustrated by the first row of Figure 5.26, the sub-frame with the card to identify resides in the black-and-white BRAM. To isolate this card and ultimately proceed to its identification, the system generates a blank card and tries to register (match) the position of the card to identify. This generated card, referred to as the blank card, is simply a white rectangle based on the dimensions of the playing cards as observed by the camera. The blank card, as illustrated by the second row of Figure 5.26, is generated at the angle of the player's spot to improve the registration since the card to match is more likely to be at that angle. Note that these dimensions and rotation angle are configurable parameters in the register map.

The blank card is generated in another BRAM using a predefined coordinates look-up table. Given a playing card's short edge length (in pixels; the long edge is a fixed multiple of the short edge) and the rotation angle (in degrees), the position of the four corners of the *to-be-generated* blank card are fetched from the loop-up table. Using these coordinates, the rectangle generator block fills up the blank card BRAM with the image of the blank card. The blank card BRAM is also a 60x60 pixels memory to simplify the registration computation even though the blank card will always be much smaller than the sub-frame image. The unused pixels in the blank card BRAM are set to black.

To perform the card position registration, as described in the previous chapter, a sum of square differences (SSD) algorithm is applied between the blank card and the sub-frame. The blank card is initially generated by the rectangle generator in the top left corner and is shifted by one pixel right and one pixel down at the time, in a zig-zag manner, computing the SSD at each position. Once the SSD as been applied to all positions, the position with the lowest SSD is determined to be the registered position of the card. Note that the card to identify may not be the only one in the sub-frame but it will always be above any other card and, most importantly, the topmost card in the image. This is the reason why the first minimum SSD is considered the correct registration of the card to identify.

As Figure 5.25 shows, the rectangle generator block also serves the purpose of controlling the flow of data since it is already driving the *data* and *address* lines of the blank card BRAM. It circulates the addresses for the blank card BRAM and the black-and-white BRAM in order to register the position with the lowest SSD value from the SSD block. Once this is done, the rectangle generator accomplishes the task of copying a trimmed version of the card contained in the black-and-white BRAM to another BRAM, the trimmed card BRAM. The trimmed version of the card, as illustrated by the last row of Figure 5.26, only contains the interior region of the card, that is the dots or the figure of the card, and is going to be used to identify

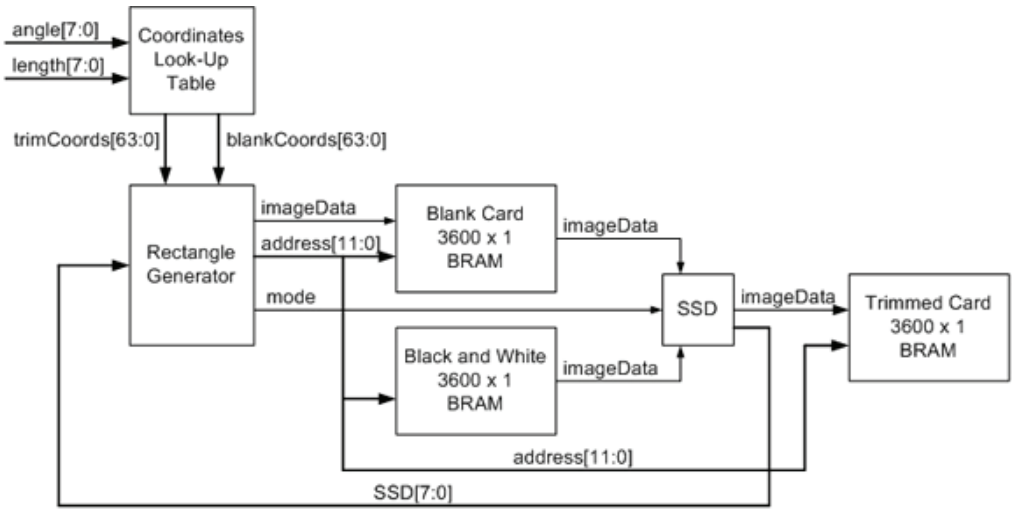

Figure 5.25: SSD Engine

the value the card. In order to do this, the blank card BRAM is loaded with a white rectangle generated at the registered position of the card but uses the trimmed subset of coordinates from the look-up table, that is the *trimCoords[63:0]* bus. The trimmed subset of coordinates corresponds to the four corners of the interior region of the card as illustrated by Figure 4.10. That being done, the rectangle generator switches the mode of operation of the SSD block into a logical AND, circulates the addresses of the blank card BRAM and blank-and-white BRAM and writes the output of the logic AND to the trimmed card BRAM. This operation has the overall effect of masking the blank-and-white sub-frame image with a rectangle window that only exposes the interior region of the card and therefore leaves only the dots or the figure of the card in the trimmed card BRAM. The interior card region mask is illustrated by the third row of Figure 5.26.

Figure 5.26 illustrates an example of the card isolation process for the seven players.

Card Value Determination The determination of the value of the card is done by counting the number of dots. In the case of the face cards, the black and white ratio

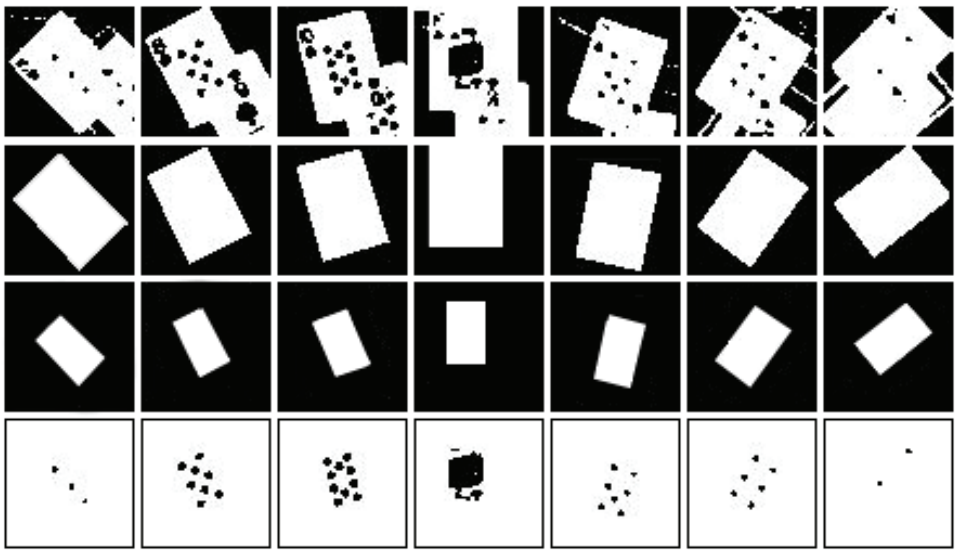

Figure 5.26: Example of cards isolation for the seven players on their second card (From left to right: players one to seven; From Top to bottom: Black and white sub-frame, blank card matching and trimmed card)

prevails no matters the number of dots. In order to avoid that noise or artifacts affect the perceived number of dots, a majority morphological operation is applied to the trimmed card BRAM. This operation sets a pixel to black only if five or more pixels of the 3-by-3 neighbors are also black. To accomplish this, a morphological engine, illustrated by Figure 5.27, reads the trimmed card BRAM and fills another BRAM of the same size, the clean dots BRAM. From each pixel to be written, the morphological engine reads also the surrounding 8 pixels and determines if the conditions turn the pixel black or white. The content of the clean dots BRAM is then ready to be processed in order to determine the black and white ratio and the number of dots.

To determine the black and white ratio, the morphological engine reads all the pixels from the clean dots BRAM and sums up all the black pixels. The ratio is established by comparing this value with the total number of pixels in the interior region of the card, a predefined value provided by the register map. If there are more black pixels than white pixels, the card is automatically identified as a face card and the identification process stops there. If there are more white pixels than black pixels, the number of dots must be taken into account.

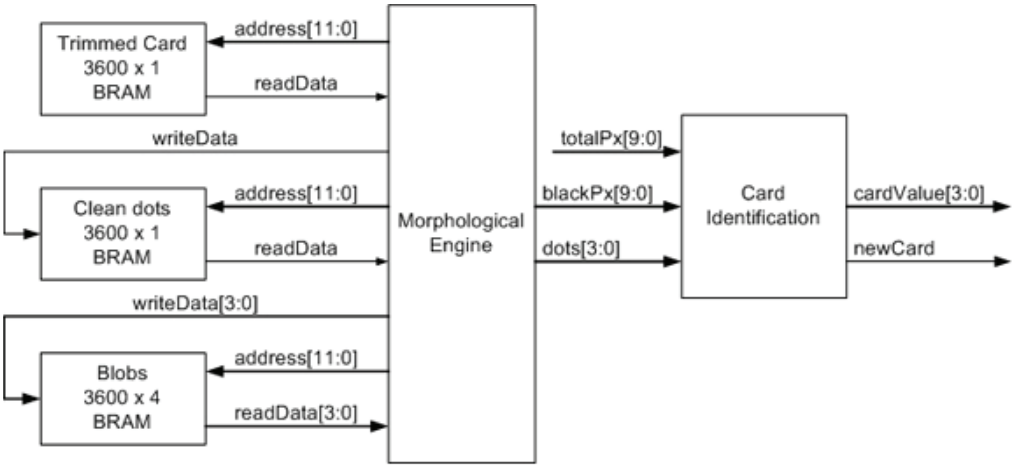

Figure 5.27: Morphological Engine

The number of dots is determined by applying a segmentation algorithm to the black regions contained in the clean dots BRAM. The segmentation algorithm has the goal to assign a unique number to each group of pixels that touches each other and that therefore forms a blob. Because a single bit, that is the black or white pixel, is insufficient to distinguish many groups of pixels, a copy of the clean dots BRAM is made into a 4-bit wide BRAM, the blobs BRAM. Each black pixel is initially copied as 0xF and white pixels are set as zeros. A pixel labeled as 0xF indicates an unidentified blob. With this set up, the morphological engine traverses all the pixels and sets the new label according to their 3x3 neighbors:

- A pixel that is part of an unidentified blob, that is if all 3x3 neighbors are also unidentified, is assigned the current count, which starts at 0x1.

- An unidentified pixel that has a single identified 3x3 neighbor is assigned the same blob label.

- An unidentified pixel that has two or more identified 3x3 neighbor is assigned one of the previously assigned blob label. These equivalent blob labels are noted and once the image is fully traversed, a new pass is done to assign a unique label to equivalent blobs.

The number of unique blobs is the number of dots and this value is used in relation with the black and white ratio to determine the card value. The card value is produced by the card identification block according to the *blackPx[9:0]*, *whitePx[9:0]* and *dots[3:0]* buses value. A new card is signaled to the finite state machine by pulsing the *newCard* signal for one clock cycle. To determine the card value, the card identification block uses the following conditions:

- If there are more black pixels than white pixels, the card is a face card and its value is ten.

- If there there are less than five dots but there are less than ten times or more white pixels than black pixels, the card is the ace of spade and its value is one.

- If the two conditions above are false, the number of dots equals the card value.

5.2.4 Console

The console is the interface between the card tracking system and the user. The console serves two purposes: (1) configure parameters into the system by writing to the register map and (2) monitor the results of each game. The hardware link is established using a RS-232 serial interface and is accessible via any host equipped with such hardware requirements.

Chapter 6

System Validation and Results

6.1 System Validation Methodology

The presented Blackjack tracking system was tested using video footage from a camera placed above a Blackjack table similar to those found in casinos. In the footage, 100 games of Blackjack were played following simple Blackjack rules, that is without splitting, doubling or insurance. Players played according to basic strategy rules and the dealer followed usual house rules, that is to hit as long he/she has 16 or less or on a soft 17 (hand with at least one ace). Different test cases were applied to test the system under different game flows and conditions. These test cases and their results are presented below. In addition, the resource utilization and the constraints on the selected FPGA device are described.

6.1.1 Test Cases

The test cases used are the following:

Figure 6.1: Sample frame from test footage

- Case 1: 30 games with 7 players following appropriate dealing conditions

- Case 2: 30 games with 6 players or less following appropriate dealing conditions (five games with each possible player count with different permutations)

- Case 3: 10 games with 7 players where cards were dropped with an exaggerated spread between each card

- Case 4: 10 games with 7 players where cards were dropped with an exaggerated angle with respect to the player's spot

- Case 5: 10 games with 7 players and lighting conditions varying from dim to bright

- Case 6: 10 games with the dealer and the players creating unsolicited activity

in the player's spot

6.1.2 Playing Conditions for Dealer and Players

The dealer plays an important role in dealer cards in such a manner that the system will more easily track the game being played. Appropriate dealing conditions are defined as follow:

- Condition 1: The dealer places the first card of each player as much as possible in the middle of the player's spot

- Condition 2: The dealer places each card in the player's spot at the angle as close as possible to 90 degrees

- Condition 3: The second and following cards are placed diagonally exposing about 1/5 of the previous card (see figure 4.2)

- Condition 4: The dealer does not create activity in the player's spot unless to drop a card.

The players are not supposed to interact with the video processing sensitive regions, that is the region surrounding the cards. Current casinos already impose such a rule by preventing players to touch cards (or approach them) in a game of Blackjack. The main reason being to avoid players to have the possibility to mark cards. Still, players can create activity in the player's spot before cards are dealt to them, creating false first card detection. Therefore, the only condition for players is to avoid to place their arms and/or hands beyond the dashed line on Figure 3.2.

6.2 Results

The following lists the results obtained for the test cases mentioned above. For each test case, the percentage of error is presented in two ways: card-wise and hand-wise. The card percentage of error corresponds to the number of correctly identified cards over the total number of cards dealt. The hand percentage of error corresponds to the number of correctly identified hands over the total number of hands dealt (the dealer's hand is considered under this percentage). For every test case, the key elements that failed are highlighted. A single misidentified card will automatically cause a misinterpreted hand, unless the faulty interpretation yields the same result. For that reason, the hand percentage of error is always greater than the card percentage of error.

6.2.1 Test Case 1: 30 games with 7 players; Appropriate dealing conditions

With seven people playing 30 games under appropriate dealing conditions, the card and hand percentages of error were respectively 0.52% and 1.67%. That means that out of 768 cards drawn, four cards were misidentified causing four hands out of 240 to be misinterpreted.

This is the absolute best that the system can achieve because all the games were played optimally given the video processor limitations, which are described in the following test cases. The four cards were misidentified because the dealer put down the card too slowly and his hand was captured with the card to be identified.

6.2.2 Test Case 2: 30 games with 6 players or less; Appropriate dealing conditions

With one to six people each playing 5 games under appropriate dealing conditions, the card and hand percentages of error were respectively 0.75% and 2.22%. That means that out of 401 cards drawn, three cards were misidentified causing three hands out of 135 to be misinterpreted.

The goal of this test case was to determine if interleaving different numbers of players would affect the activity detection by the insertion of irregular delays. The activity detection itself was not affected and cards were identified with similar reliability that with seven players.

6.2.3 Test Case 3: 10 games with 7 players; Exaggerated spread between each card

With seven people playing 10 games where cards were dropped with exaggerated distance between each card, the card and hand percentages of error were respectively 4.95% and 8.75%. That means that out of 222 cards drawn, eleven cards were misidentified causing seven hands out of 80 to be misinterpreted.

Spreading the cards has the effect of putting the cards down closer to the edge of the observable regions. As long as the card is fully captured in the sub-frame, this case does not raise issues. However, putting the card down on the edge of the observable region or beyond automatically results in misidentification of the card because the image of the latter is cropped. Once a card fails, the following cards will also fail because the displacement of the observable region relies on the previous card. This situation therefore highly increases the number of misidentified cards and hands.

6.2.4 Test Case 4: 10 games with 7 players; Exaggerated card angle with respect to the player's spot

With seven people playing 10 games where cards were dropped with exaggerated angle between each card and the player's spot (not at 90 degrees), the card and hand percentages of error were respectively 3.04% and 7.5%. That means that out of 230 cards drawn, seven cards were misidentified causing six hands out of 80 to be misinterpreted.

Because the cards are at an angle, they tend to be off the detection line and the values registered for the displacement of the observable region are inaccurate. Again, once a card fails, the following cards will also fail because the displacement of the observable region relies on the previous card. This however only raise issues in extreme cases, that is with the angle off by 30 degrees or more.

6.2.5 Test Case 5: 10 games with 7 players; Lighting conditions varying from dim to bright

With seven people playing 10 games where lighting conditions varied from dim to bright, the card and hand percentages of error were respectively 43.7% and 50%. That means that out of 268 cards drawn, 117 cards were misidentified causing 40 hands out of 80 to be misinterpreted.

In this test case, it was important to have seven people playing to observe how light condition can vary from one end of the table to the other. Usually if the lighting is globally too bright or too dim, the camera is able to compensate. However, with lighting conditions varying across the table, some regions are too bright while others are too dim. This situation is even more critical because playing cards are highly reflective and as soon as the light is too bright or too dim, the cards are still extracted

but it is impossible to identify the dots. With too much light, the card is viewed as a white rectangle and no dots are seen. With not enough light, the card is viewed as a black rectangle and again, no dots are seen. The leftmost and rightmost playing spots of the table are mostly affected by this condition.

6.2.6 Test Case 6: 10 games with 7 players; Unsolicited activity in the player's spot

When the system expects activity and the latter is created by unsolicited means (i.e. dealer's or player's hand), the system automatically captures false sub-frames. The card cannot be identified from this sub-frame simply because no cards were present at the moment of the capture. In addition, because the key sub-frame is wrong, all the subsequent cards of the hand will also be misidentified. Therefore, under this test case, as soon as unsolicited activity was generated, the card and hand were misinterpreted.

6.2.7 Limitations

The results above highlight the limitations of the system.

First, the card must absolutely be placed in the observable region for proper identification. It would be possible to enlarge the observable regions but this would imply that more artifacts would also be included in the sub-frame, therefore creating more possibilities for error. In addition, this would require more hardware resources to process the additional data.

Then, the pace at which the dealer drops the card in the observable region is very sensitive. If the card is dropped too slowly, the hand of the dealer will be

Xilinx 2VP30FF896-7			
Resource	Used	Total	Utilization %
Slices	8544	13696	62.4%
Slice Flip-Flops	17288	27392	63.1%
4-Input LUTs	15698	27392	57.3%
Bonded IOBs	56	556	10%
BRAMs	31	136	22.8%
18x18 Multipliers	22	136	16.2%
GCLKs	2	16	12%

Table 6.1: Resource usage of FPGA device

Xilinx 2VP30FF896-7			
	Min	Max	Unit
Clock period	9.594		ns
Clock frequency		104.233	MHz
Setup time	2.698		ns
Hold time	5.236		ns

Table 6.2: Constraints of FPGA device

captured. In addition, similar problems arise when unsolicited activity is generated. These limitations however tend to disappear as the dealer is sufficiently trained on the system.

Finally, the lighting conditions are the most critical limitations of the system. The camera and table must therefore be carefully adjusted to avoid situations with unfavorable lighting.

6.3 Resource Usage and Constraints

The design was implemented in a Xilinx Virtex-II Pro XC2VP30 FPGA device that provides 30,816 logic cells. Table 6.1 and 6.2 illustrates the resource usage and constraints.

Chapter 7

Conclusion

This thesis demonstrated a system that attempts to track the game of Blackjack using a video feed from above a Blackjack table.

First, background information on the game of Blackjack, the motivations and contributions of such a system were exposed. Background information on the game of Blackjack included the rules of the game and how the game is played in casinos. The motivation to develop a Blackjack tracking system included the fight against cheating and validation of the dealer's work. The contribution of the proposed system is the identification of every card dealt to the players and the dealer. The correlation of the identified hands with the players' bet is not addressed.

Second, literature on the subject was reviewed. Two groups of researchers presented work that addresses the identification of playing cards. One specific thesis by Wesley Cooper attempts to solve the same problem of tracking every card in a game of Blackjack. In addition, given the limited literature on the subject, three newly released commercial products that identify cards in a casino environment were reviewed.

Then, the framework proposed to track a game of Blackjack in a casino was explained and the developed card tracking algorithm was exposed. The framework pictured a FPGA device used to process a video feed coming from a camera above a Blackjack table. The algorithm highlighted the flow of the game and how specifically each step of the game is tracked resulting in the identification of every card dealt.

Finally, the hardware implementation of the system was described and specific algorithm computing details were given. The hardware shown to be composed of three main elements: a video frame grabber to capture video data from the camera, a video processor to apply the card tracking algorithm and return card values, and a console used to output the results of each game and enables the operator to input parameters into the system.

Under optimal conditions, the system had a percentage of error as low as 0.5%. However, limitations prevent the system to be immune to human errors and such cases were responsible for percentage of error as high as 4.95%. Furthermore, it was shown that lighting conditions must be adapted to the system to avoid very high percentage of error.

7.1 Futher Improvements

Further improvements are suggested to the current implementation of the system.

In a first time, the card identification should account for lighting conditions. Contrast should be automatically adjusted and compensated given the lighting conditions.

In a second time, the system should account for advanced Blackjack rules such as splitting, doubling and insurance. The current implementation was demonstrated to track cards in the player's spots but accounting for splitting implies that much more

manipulation is done with the cards and the video processing challenge associated with this is much higher.

Finally, the results of the game as well as the in-game information should be displayed to the operator by the means of an on-screen display. Since the frame grabber already returns the content of the frame buffer to a VGA output, this improvement is possible by "ORing" data in the frame buffer. Having the relevant information returned to the operator by the means of the on-screen display, the console should then only serve the purpose of configuring the register map and interfacing the external betting system. In addition, given that RS-232 is considered a legacy communication protocol, it should be replaced by a more modern protocol such as a universal serial bus (USB) to enhance compatibility with host systems.

Appendix A

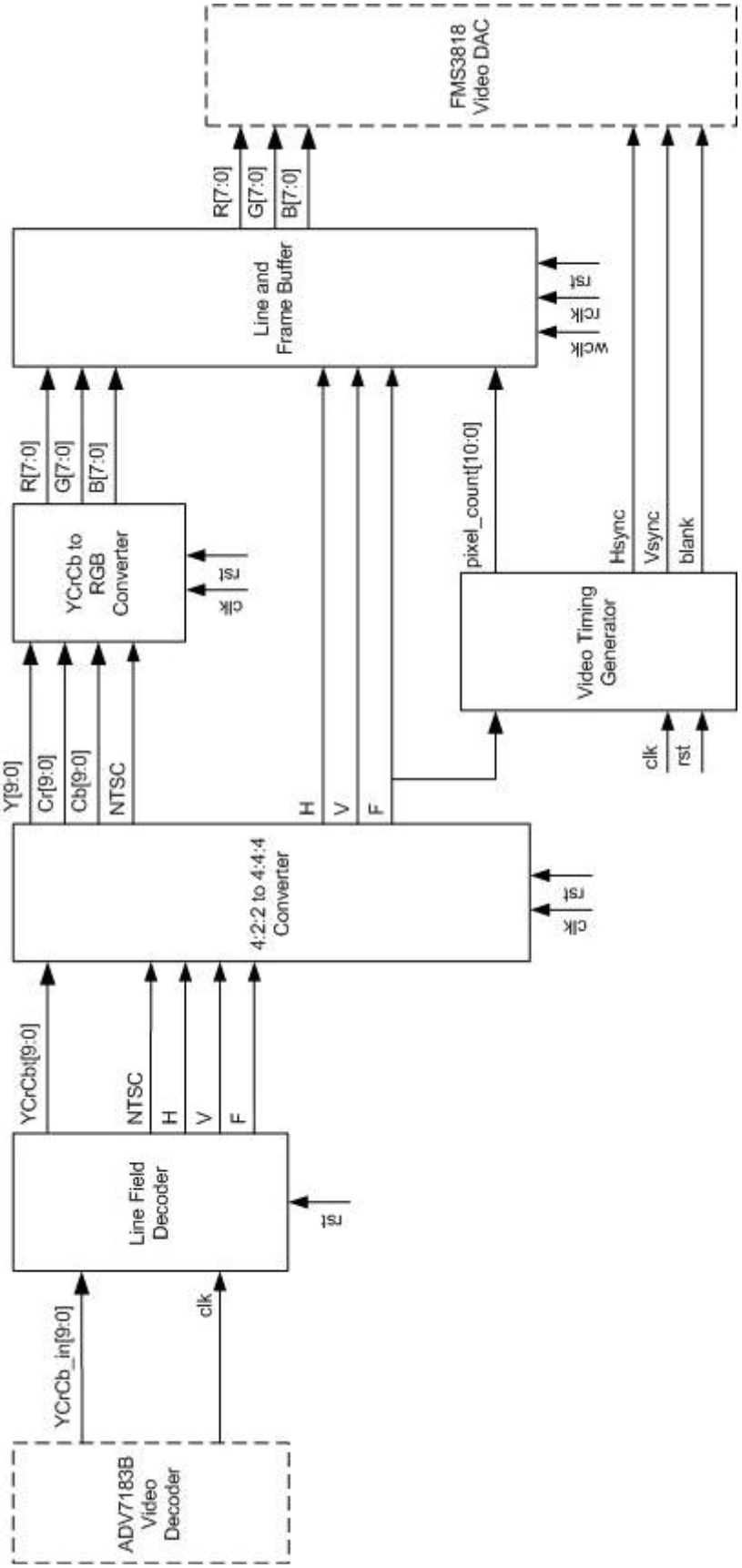

Figure A.1: Detailed Video Data Grabber Block Diagram

Bibliography

Bally Technologies (2007). Tms.21. www.ballytech.com.

Cooper, W. (2004). Blackjack tracking system. Master's thesis, University of Dublin, Trinity College.

Digilent Inc. (2007). Digital design engineering's source (website). www.digilentinc.com.

Eatinger, S., Graf, B., Victory, M., and Wagner, R. (2000). Playing Card Recognition Project, Rice University. http://www.owlnet.rice.edu/ elec301/Projects99/nphaze/research.html.

Griffin, P. A. (1999). *The Theory of Blackjack*. Huntington Press, 6th edition.

International Telecommunication Union (1998). Recommendation itu-r bt.470-7, conventional analog television systems.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999*, 2:1150–1157.

Park, D. and Hush, D. (1990). Statistical analysis of a change detector based on image modeling of difference picture. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 4:2185–2188.

Prasad, L. and Iyengar, S. (1996). High performance algorithms for object recognition problem bymultiresolution template matching. *Seventh International Conference on Tools with Artificial Intelligence*, Pattern Recognition 29:641–662.

Rehrmann, V., Lakmann, R., and Priese, L. (1999). A parallel system for real-time traffic sign recognition. *Intelligent Vehicles Symposium 93*, pages 95–100.

Shuffle Master (1997). U.S. patent 5,651,548: Gaming chips with electronic circuits scanned by antennas in gaming chip placement areas for tracking the movement of gaming chips within a casino apparatus and method.

Shuffle Master (2007). Is-b1 intelligent shoe (brochure). www.shufflemaster.com.

Sipser, M. (2006). *Introduction to the Theory of Computation*. Boston Mass: Thomson Course Technology, 2nd edition.

Sonka, M., Hlavac, V., and Boyle, R. (1999). *Image Processing, Analysis, and Machine vision*. PWS publishing, 2nd edition.

Tangam Gaming (2007). Automated table tracking (brochure). www.tangamgaming.com.

Thorp, E. O. (1966). *Beat the Dealer*. Vintage Books, 1st edition.

Trier, O., Jain, A., and Taxt, T. (1996). Feature extraction methods for character recognition - a survey. Pattern Recognition 29:641–662.