Aufgabe 1:

* 1. Studium der Dokumentation

Bei dem Betrachten der VGA Signale durfte man nicht vergessen das die Signale eigentlich low aktiv sind.

Die Dokumentation des FPGAs betreffend der Anschlüsse des VGA Ausgangs war sehr klar und eindeutig. Es gibt jeweils Anschlüsse für horizontal und vertical Sync sowie vier Anschlüsse pro Farbe (RGB). Die 25.175 MHz Clock war auch sofort zu finden.

* 1. Logischer Ablauf

Unser Programm sollte wieder modular und klar strukturiert sein.

Als all umfassende Entity benutzen wir den „vgagenerator“ der die Schnittstelle zum FPGA bildet.

Darin wird die Clock zum Signalgenerator weitergeleitet der sich darum kümmert horizontal bzw. vertical sync zu generieren und dazu die aktuelle Spalte und Reihe ausgibt.

Die Information zur aktuellen Zelle wird dann an den Pixelgenerator weitergegeben der bei bedarf für die jeweilige Aufgabe ausgetauscht werden kann um der Aufgabenstellung gerecht zu werden.

Mit Hilfe von verschiedenen Hilfsgeneratoren wie „Rectangle“, „Gradient“ und „Whiteborder“ kann der Pixelgenerator dynamisch das Bild aufbauen.

Die vom Pixelgenerator ausgegebenen Werte werden dann mit hsync und vsync am FPGA angelegt.

* 1. Sync-Generator

Der Sync Generator hat als Eingang die Clock und gibt am Ende das hsync bzw vsync Signal mit der aktuellen Spalte und Reihe aus.

Dieser erreicht das mit Hilfe von zwei Componenten: zunächst der hsyncgenerator der das hsync Singal und die Reihe durch die Clock ausgibt dann der vsyncgenerator der mit dem hsync Signal die Spalte und vsync ausgibt.

Die jeweiligen Componenten benutzen die in der Vorlesung vorgestellte Zählweise, z.B. der hsync wartet erst 44 Ticks bevor er valid wird und erreicht dann nach weiteren 640 den Post validen Zustand bei dem das Zeilen Signal -1, also invalid wird und der hsync 0 bleibt bis nach 20 weiteren Ticks das hsync Signal auf 1 gesetzt und die Zeile weiter auf -1 bleibt. Am Ende werden die Zähler noch zurückgesetzt bevor das ganze wieder von vorne losgeht.

* 1. Pixel-Generator

Der Pixel-Generator ist unsere austauschbare Komponente die für jede Aufgabe angepasst wird. Für die Generierung der Formen bzw. Verläufe haben wir extra Hilfskomponenten geschrieben die miteinander kombiniert und vermischt werden können um das gewollte Bild zu generieren.

* 1. Simulation

Für die Simulierung des Pixel- bzw. Signalgenerators haben wir Testbenches geschrieben die wir zunächst unter geringerer Auflösung in GTKWave getestet haben um die Simulationszeit zu verkürzen. Die Testbenches haben wir mit Hilfe eines selbst geschriebenen Scripts einfach gemacht auszuführen. Das Testen hat uns dabei geholfen frühzeitig Fehler in unserer Implementierung zu identifizieren und gleich zu beheben.

* 1. Gesamtschaltung

Nach dem wir sicher waren das unser Pixel- und Signalgenerator funktionieren testeten wir die gesamte Schaltung. Diese lief dank der Testbenches für Pixel- und Signalgenerator auf anhieb. Wir testeten nur noch ob das ausgegebene Bild richtig war.

* 1. Testbild und Synthese

Nach Problem durch vergessen des Clock events oder vergessen einen Port als “in” zu bezeichnen bekamen wir dann erflogreich unser Testbild zu stande.

Aufgabe 2:

Um den Farbverlauf zu erreichen haben wir zuerst eine Gradient Komponente erstellt die dafür zuständig war verschiedene Farbverläufe zu generieren. Diese haben wir mit dem weißen Rand generator in die für diese Aufgabe neu angelegte Pixelgenerator eingebunden und die Farbwerte verodert um den Verlauf mit Rand zu erreichen.

Aufgabe 3:

Wie auch schon zuvor haben wir auch für das Rechteck einen Generator erstellt der als Eingabe die Zelle, Farbe sowie zwei Punkte die das Rechteck aufspannen und Buttons zum verschieben des Rechtecks. Ausgegeben wird dann wieder die Zelle sowie die Farbe des Pixels. Wie bei Aufgabe 2 wird das erstellte Rechteck mit dem Rest des Bilds verodert um die Shapes zusammenzufügen.

Aufgabe 4:

Die bereits in Aufgabe 3 implementierte Logik zum verschieben von Rechtecken kommt hier jetzt zum Einsatz. Mit Hilfe der vier blauen Tasten auf dem FPGA kann somit ein Eckpunkt durch das Bild geschoben werden